FOI-R--1180--SE
February 2004
ISSN 1650-1942
User report

Pontus Svenson, Christian Mårtenson, Hedvig Sidenbladh, Michael Malm

Swarm Intelligence for logistics: Background

Command and Control Systems
SE-581 11 Linköping

SWEDISH DEFENCE RESEARCH AGENCY

Command and Control Systems
P.O. Box 1165
SE-581 11 Linköping

FOI-R--1180--SE

February 2004

ISSN 1650-1942

User report

Pontus Svenson, Christian Mårtenson, Hedvig Sidenbladh, Michael Malm

Swarm Intelligence for logistics: Background

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 Linköping	FOI-R--1180--SE	User report
	Research area code 4. C4ISR	
	Month year February 2004	**Project no.** E7849
	Customers code 5. Commissioned Research	
	Sub area code 41 C4I	

Author/s (editor/s)		Project manager
Pontus Svenson	ponsve@foi.se	Pontus Svenson
Christian Mårtenson	cmart@foi.se	**Approved by**
Hedvig Sidenbladh	hedvig@foi.se	
Michael Malm	micmal@foi.se	**Sponsoring agency** FMV
		Scientifically and technically responsible Pontus Svenson

Report title

Swarm Intelligence for logistics: Background

Abstract (not more than 200 words)

In this report, we describe the result of a literature survey on swarm intelligence, focusing on possible applications for logistics in network-based defense. Some background on swarming and its uses for understanding biological and sociological phenomena is given, and the two major variants of swarming methods for optimization are thoroughly described. One of these, ant colony optimization, is based on the behavior of foraging ants, while the other, particle swarm optimization, mimics the behavior of a crowd wherein each participant is influenced by their neighbors. For both of these methods, we describe the original work presenting them and some interesting applications and refinements. Both methods have been extended significantly by several different authors from their original versions.

We also describe previously published ideas on military applications of swarming, and give a brief introduction to some problems related to supply-chain management for the future network-based defense.

The main result of this study is the bibliography. Here we collect a large number of papers on applying swarm intelligence, with emphasis on applications relevant for logistics.

Keywords

Swarm intelligence, ant colony optimization, particle swarm optimization, heuristical methods, logistics, spare parts optimization, supply-chain management

Further bibliographic information	Language English
http://www.foi.se/fusion	
ISSN 1650-1942	**Pages** 43 p.
	Price acc. to pricelist

Utgivare
Totalförsvarets Forskningsinstitut - FOI
Ledningssystem
Box 1165
581 11 Linköping

Rapportnummer, ISRN
FOI-R--1180--SE

Klassificering
Användarrapport

Forskningsområde
4. Spaning och ledning

Månad, år
Februari 2004

Projektnummer
E7849

Verksamhetsgren
5. Uppdragsfinansierad verksamhet

Delområde
41 Ledning med samband och telekom och IT-system

Författare/redaktör
Pontus Svenson ponsve@foi.se
Christian Mårtenson cmart@foi.se
Hedvig Sidenbladh hedvig@foi.se
Michael Malm micmal@foi.se

Projektledare
Pontus Svenson

Godkänd av

Uppdragsgivare/kundbeteckning
FMV

Tekniskt och/eller vetenskapligt ansvarig
Pontus Svenson

Rapportens titel (i översättning)
Swarm Intelligence för logistik: bakgrund

Sammanfattning (högst 200 ord)

Den här rapporten är resultatet av en litteraturstudie om swarm intelligence. Studien har främst inriktats på logistiknära tillämpningar med användning inom det nätverksbaserade försvaret. Vi beskriver överskådligt bakgrunden till swarming och hur det använts inom biologi och sociologi. Två huvudvarianter av swarm-metoder för optimering beskrivs i detalj. Den första, ant colony optimization, baseras på hur myror letar efter mat, medan den andra, particle swarm optimization, baseras på hur individer i folkmassor låter sitt beteende påverkas av sin omgivning. Vi beskriver originalversionerna samt några intressanta tillämpningar och utvidgningar för respektive metod.

Vi nämner också tidigare tidigare framfördra idéer om militära tillämpningar av swarming, och ger en kort introduktion till vissa logistikproblem som är relevanta för det framtida nätverksbaserade försvaret.

Studiens huvudresultat är bibliografin. Här har vi samlat en stor mängd artiklar som tillämpar swarm intelligence på olika problem, främst logistikrelaterade sådana.

Nyckelord

Övriga bibliografiska uppgifter
http://www.foi.se/fusion

Språk Engelska

ISSN 1650-1942

Antal sidor: 43 s.

Distribution enligt missiv

Pris: Enligt prislista

Contents

Chapter 1

Introduction

1.1 Outline and summary

In this report we describe a preliminary study exploring the possibilities of improving military logistics processes by applying so called swarm intelligence methods. The study was financed by the Swedish Defence Materiel Administration. The purpose of the study was to do a thorough literature survey in order to be able to decide whether to try to use swarming to solve logistics-related optimization problems. We have found a large number of papers that deal with problems similar to those occurring in network-based defense logistics, and hence we are confident that a further study will reveal many useful applications of swarming to military logistics.

In this introductory chapter, we give an introduction to logistics and supply chain management in section 1.2. Section 1.3 then briefly presents one specific important logistics problem, spare parts optimization. This is the problem which a possible follow-up study should focus on.

In section 1.4, the reader is given a first glimpse of swarming. Here we discuss the origin of swarm intelligence methods, and present some applications of it. For completeness, section 1.5 briefly describes some of the previous applications of swarming for military purposes. We believe that the concept of swarming will prove an essential part of any successful implementation of a network-based defense.

In chapter 2, we turn to *ant colony optimization*, the first and arguably most successful optimization method based on swarming. We introduce the algorithm by focussing on a specific problem, the Travelling Salesman Problem, which is described in section 2.2. The basic ant colony optimization method is then described in sections 2.3 and 2.4, while section 2.5 describes the Max-Min improvement of it. Other refinements are described in sections 2.6 and 2.7. The theoretical description of ant colony optimization is finished in section 2.8, where we present results relating to the guaranteed convergence of it. Applications of ant colony optimization for logistics, routing and scheduling are described in sections 2.9.1, 2.9.2 and 2.9.3. The chapter concludes with a section on miscellaneous applications.

Particle swarm optimization is described in chapter 3. This chapter starts by introducing the algorithm and comparing it to evolutionary computing methods in section 3.1. Extensions are described in section 3.2. These extensions are very relevant for military logistics, since they deal with dynamical optimization, constraints, and how to optimize several functions at the same time. Section 3.3 describes applications of particle swarm optimization.

Chapter 4, finally, presents our conclusions and gives some possible directions for future work.

The main result of this study is the bibliography. Here we collect a large number of papers on various aspects of swarm optimization. We also include appendices explaining the concept of NP-completeness and hard optimization problems and giving a list of swarm-related web-sites. This list is certainly not as complete as the bibliography.

1.2 Supply-chain management in network-based defense

The Swedish Armed Forces is going trough a fundamental transformation from a defense against invasion of mainland Sweden to an international active objective defense force. Defending Sweden and contributing to international security by participation in a full spectrum of operations together with UN, NATO and EU.

Today the spectrum of operations from war to military operations other than war (MOOTW) is almost infinite which requires a flexible force structure that is adaptable to world wide operations, against opponents that are not traditional military forces but more often clans, para-military forces, criminal groups, terrorists etc in a changing environment with or without traditional frontlines.

A flexible force structure is created with a modular based force concept. A module is the smallest entity in the system that can be put together with other modules in order to create a unit. An example of a module can be a medical expert, or an airplane with its crew, or a number of people with specific training, equipment and readiness. In the future one of the challenges would be to create a required objective force unit out of a number of modules, ready to deploy, in a very short time. The number of modules in the future Swedish Armed Forces is probably enormous.

The logistical concept for future operations, domestic or international, has to be able to support the modular based force concept "from factory to foxhole". The logistical concept has to be able to deal with every aspect of the supply and value chain with domestic and international suppliers, different supply concepts together with others in joint operations or alone.

In order to fulfill its task the future logistical concept and system has to be able to organize and facilitate the flow of information, people and equipment (material, spare parts, water, fuel etc) in a way that ensures that the right equipment, people and information end up at the right place at the right time in the right way all the way from the very first idea all the way through the supply chain to the soldier/operator in the field. Traditionally Sweden have had a few main suppliers (SAAB, Volvo, Hägglund etc) and a force structure with well defined pre fixed units (platoons, companies, battalions, brigades etc) and objectives only within its own borders which was not an easy task to make work but still much less complex, fragmented and unknown than what we have today and what we will have in the future. Many of the domestic Swedish suppliers are no more Swedish and with a more international security and military doctrine more foreign suppliers have become an option. As we said earlier we will not in the future have pre fixed units that belong primarily to a geographical area and a specific task within our national borders but a broad range of capabilities and modules ready to be configured into objective force units.

We have therefore to find different ways to manage the future supply chain and find different logistical concepts in order to manage future (and current) operations.

A set of challenges for logistics in a transformed army has been formulated in (RAND, 2003). In particular, distribution-based logistics, which aims at reducing stockpiles, is a natural candidate for solving using swarm methods. This problem can be seen as an extension of the spare parts optimization problem described below.

1.3 Spare parts optimization

Spare parts optimization can be seen as a prototype problem for supply management in network-based defense. The variant of it which we will describe here is based on the one presented and analyzed in (Alfredsson, 1997).

Consider a military expedition that needs to have a certain amount of aircraft airborne at all times. This will require several air-bases and a supply-chain that provides these with necessary spare parts (and fuel, ammunition and other expendables). Consider a setting where there are A bases. Each base a has a certain number N_a of aircraft. During missions, these aircraft will suffer damage and parts of them will need to be replaced and/or repaired. If there are K resources/parts that can fail on the airplanes[1], each base needs to have supplies of K different resources. Our problem can now be formulated quite succinctly as "determine the amount of supply of each type at each base so that the number of grounded aircraft is minimized while limiting transportation costs". In the spare parts optimization problem, the demand for part k at site a is referred to as λ_{ak}. This quantity is determined from the failure rates of the parts (Alfredsson, 1997).

Sometimes a faulty part of an aircraft can be repaired directly, in other cases it needs to be replaced and repaired. Such repairs can sometimes be performed at the base, but for complicated parts it is necessary to send the part to a central depot and have it repaired there. Such transports cost, and it is thus necessary to limit them. Repairs also take some amount of time, and it is thus necessary for each base to have a large-enough supply of each part so that it can keep as many of its aircraft airborne as possible. In (Alfredsson, 1997), T and R denote the average time to ship a part to a base and the time needed for repair, respectively, while the current stock level at a base is denoted S. Thus, T_{ak} would denote the time to ship part k to base a. Items that have been ordered or are undergoing repair are said to be in the pipeline; the number of items in this and the expected time an item remains there are called X_t and L. Given this information, it is possible to calculate the mean waiting time for a spare (Alfredsson, 1997), $W = B/\lambda$, where B, the expected number of backorders, can be found from the distribution of X_t. The mean waiting time for a spare can now be calculated to be

$$MWTS = \frac{\sum_a \sum_k B_{ak}}{\sum_a \sum_k \lambda_{ak}}. \tag{1.1}$$

[1]For simplicity, we will assume that all aircrafts are identical. Extending the formulation for different types of aircraft is trivial.

This gives us the meantime to repair aircraft as

$$MTTR = \frac{\sum_a \sum_k \lambda_{ak} MTTR_{ak}}{\sum_a \sum_k \lambda_{ak}}, \tag{1.2}$$

which finally leads to an expression for the number of operational aircraft

$$NOR = \sum_a \sum_k \lambda_{ak} MTTR_{ak} + B_{ak}. \tag{1.3}$$

Equation 1.3 is the objective function for optimization. For more details on the approximations and calculations needed to get to this equation, see (Alfredsson, 1997). We also need to take account of budget constraints that limit the number of parts that we can store at each site, which leads to a more difficult optimization problem.

The very simple spare parts optimization problem that we have formulated here can nevertheless be seen as a prototype problem for supply-chain management. We can make several extensions of it to make it more realistic. First, the assumption of one central depot is not necessarily valid. We can have several manufacturing sites and several warehouse sites. Note that each air-base could be seen as a warehouse; it could send some of its surplus to another base if the supply is too low there. Second, it is trivial to consider any kind of resource instead of parts for aircraft.

An interesting extension of this is supply-chain management for international operations other than war. Here, we have several different types of resources that are needed. The needs for different types of resources at different places can change very rapidly. Some requirements might also be more important than other. Since the needs might change quickly, an adaptive method for finding the correct flow of resources is needed.

Let there be N different sites, including both factories, warehouses and areas of operations. If there are K different types of resources (food, medicine, police, etc), we need to keep track of KN different variables representing supplies at each time. Let $X(i, a, t)$ be the amount of resource-type a on site i at time t. The need for resources is not the same as the amount of supply, we let $F(i, a, t)$ represent this.

Our goal is to get X as close to F as possible. If F is suddenly changed, we also want to be able to quickly redistribute the resources so that a new optimum is found. We describe the flow of resource a from site i to j at time t using $T(i, j, a, t)$. This is the variable that we optimize over. Furthermore, let $C(a, i, j)$ be the cost to move resource a from i to j. We then want to minimize the total cost, given by

$$\sum_{i,j,a} C(a, i, j) T(i, j, a, t), \tag{1.4}$$

while simultaneously keeping

$$\| X - F \| \tag{1.5}$$

as small as possible. Here $\| \cdot \|$ could be any metric; it could for instance be used to represent the fact that some sites or resources might be more important than others. A further complication could be introduced by allowing C to vary in time. For brief introductions to traditional ways of formulating and solving logistics related optimization problems, see (Gustafsson, 1994; Brenner, 1986)

1.4 Introduction to swarming

History teaches us that the behavior of large masses of people can sometimes be very stupid. But crowds can also display surprisingly smart behavior. Recently, attempts have been made to mimic this *collective intelligence* of crowds in computers. For a general introduction, see (Wolpert and Lawson, 2002)

Amazon.com uses a seemingly simple form of collective intelligence. When displaying a book, the site also shows books that people who bought the first book have bought. Clicking on one of these books leads to a page where yet more books that have been bought by persons buying the second book are shown. In this way, it is possible for a browser to utilize the collective intelligence of all other book-buyers and find new, interesting books.

The search site Google can be seen as another form of collective intelligence. Here, the page ranking system increases a page's rating if many people link or surf to it. Another example is Alexa, which was a system for finding related sites. (A swarming approach to a similar but more restricted problem is reported in (Semet et al., 2003).)

Many artificial intelligence approaches and optimization methods rely on centralized and hierarchically organized systems. These are most often built from a top-down perspective. Lately, an alternative approach has begun to emerge, inspired in part by swarming and the behavior of social insects. In contrast to naive belief, insects like

ants or bees are not centrally controlled hierarchically by a queen ant or bee. Instead, these insects collaborate to solve complicated optimization problems implicitly (Gordon, 2002; Sumpter and Beekman, 2003; Sahin and Franks, 2002; Buhl et al., 2002). This collaboration is controlled by communication, sometimes directly (*e.g.*, bees dancing to show directions to food) or indirectly, by individuals changing their environment and thus providing guidelines for colleagues (this is called *stigmergy*).

Swarm intelligence is a relatively new methodology that takes its inspiration from the behavior of such social insects and flocking animals. Its uses include crowd modelling for movies (Koeppel, 2004), optimization (Bonabeau et al., 2000), military history (Edwards, 2000). Variants of it have also been used in social sciences (*e.g.*, *sugarscape* (Epstein and Axtell, 1996)). For introductions to swarming and the optimization methods described in this report, see, *e.g.*, (Bonabeau and Meyer, 2001; Bonabeau and Théraulaz, 2000; Bonabeau et al., 1999; Bonabeau, 2002b; Bonabeau et al., 2000; Bonabeau, 2002a; Tarasewich and McMullen, 2002)

Originally, swarm intelligence was used in order to explain emergent biological phenomena suck as the flocking behavior of birds and fish or nest-building of termites. It has also been used for modelling robot behavior (Arkin, 1998). Swarming is a simple case of so-called *agent-based modelling*.

The basic principle of swarming is very simple: by having a relatively large number of agents following very simple rules, complicated group-behaviors emerge. One of the key features of swarming is that it may not be possible to understand the emerging global behavior by analyzing these simple rules. The only way to predict the behavior of a complicated agent system may be to simulate it. After analyzing many such simulation, it might be possible to extract "rules" governing the aggregate behavior of the system. In some cases, it might even be possible to relate these to the microscopic rules followed by the agents. In other cases, however, the microscopic rules may be completely counter-intuitive and seem to destroy rather than create the desired global behavior: "doing wrong locally might be right globally".

Such systems have a number of desirable features. They are robust, flexible and self-organizing. Robust since not all individual ants need to solve the problem. Flexible, since they can adapt in real-time to changing conditions. The self-organizing properties of swarming are important since they mean that there is no central command and control post that decides what the agents should do. This reduces the vulnerability of the system. Swarming can be seen as one kind of *self-organizing system* (Johnson, 2001).

To give a taste of how swarming works, consider the following game[2]. Take a large number of people and randomly assign a protector and an attacker to each person. Tell them that they must move around so that their protector is between them and their attacker. How will the crowd move? Since one person's attacker might be another one's protector, the motion of the crowd will be random. Consider the difficulty faced by an observer who enters the room while the crowd moves around and tries to discern the rules governing it. Contrast this to the situation that arises when the rule is changed in a very simple way: each person tries to move so that they are in between their attacker and protector. Now, everybody will try to move to the center of the crowd.

Similar sets of rules can be used to explain how flocks of birds and schools of fish form[3]. This was the first and arguably still most successful application of swarming. Computer simulations show that such flocks and schools can be formed by having a large number of simple agents following very simple rules (*e.g.*, "don't collide", "try to move in the same directions as your neighbors", and "try to reach the center of your neighbors"). This technique has been used in many movies (one recent example is Return of the King (Koeppel, 2004)). Here swarming is used to create realistic-looking scenes with hundreds of computer generated actors.

The resulting aggregations are an emergent phenomenon, occurring when a large number of agents interact. Similar rules have been found to explain how ants and other social insects find food, how termites build nests, and many other phenomena. The method used by ants to find food has inspired an optimization method (ant colony optimization) that has been successfully used to approximately solve a wide variety of hard optimization problems. The method uses a large number of simple agents that communicate by depositing pheromone in the search landscape of the optimization problem.

The simplest method for solving optimization problems using swarming is based on how some species of ants find food. When an ant leaves its ant-hill, it leaves pheromone in its trail. It uses these to be able to find its way back to the nest. When it finds a food-source, it backtracks in its own steps and enhances the pheromone. This will attract other ants, who will find the food and in turn enhance the smell further. This very strong feedback thus leads to a rapid increase of pheromones on the path that leads to the food. If the food-source is removed, the smell will cease to be enhanced and will soon evaporate. Then when another ant finds a new food-source, a new trail will appear. The ants are thus able to quickly adapt to changes.

In order to solve optimization problems, virtual ants are set loose in a fitness-landscape. When they find a

[2] See http://www.icosystems.com/game.htm , where a Java applet showing the game can be found.

[3] See http://www.red3d.com/cwr/boids/ for a more detailed explanation of this. This site also contains programs and a large list of references.

good solution to the problem, they leave virtual smell that attracts other virtual ants. This leads to a naturally robust method of solving optimization problems.

Each swarming agent reacts according to simple rules. One example of such rules are: "if there are few objects around me, pick one of them up" and "if there are many objects around me, drop the objects that I am carrying". Such rules lead to clustering, and variants of it can be used to explain how termites build their nests (Bonabeau et al., 1999) The same principle can also be used for sorting objects. Feedback is a very important principle in all swarming. Feedback can both reinforce and suppress behavior, depending on its desirability. When ants try to find food, the pheromone-path to a good food-site will be reinforced as long as it remains good. A small initial change in smell will thus be amplified and lead to large smell-differences in the terrain.

In addition to its use in the movie industry, swarming has been utilized for experimental archaeology. Agent-based simulations performed at the Santa Fe Institute have managed to accurately reproduce population and migration patterns of the Anasazi people of the American Southwest[4]. Methodology for including spatial information from Geographical Information Systems in agent-based modelling are discussed in (Gimblett, 2001). Similar methods have also been applied for studying crowd formation in carnivals and how crowds move in panic(Batty et al., 2003b; Batty et al., 2003a; Helbing et al., 1997)

Another application of swarming and agent-based simulation to logistics is implemented as a rule-based flow-handling system. Such a system has been implemented by Southwest Airlines several years ago. The airline reportedly saves $2 million per years on this. The BIOS consulting group (later acquired by NuTechSolutions[5]), used agent-based modelling to simulate the behaviors of local freight-managers in the Southwest flight-network. The problem they were trying to solve was to minimize delays and maximize throughput of parcels. When an airport receives a lot of parcels, it may not have enough outgoing flights going to the correct destinations; this introduces unwanted delays in shipping. By experimenting, the found that introducing counter-intuitive rules for how local managers should allocate parcels to flights led to a very significant decrease in delays. Unfortunately, the exact rules discovered have not been published. One example of such a rule could be "with probability 0.07, send the parcel on the first outgoing flight, even if it's in the wrong direction". Such rules seem to lead to undesired local behavior, but the aggregate behavior leads to a globally better optimum than trying to do best locally.

(Johnson, 2001) gives a popular-level description of how swarming could be used to improve networks such as the World Wide Web. The Internet is the most successful example of a thoroughly decentralized structure that works. There are many ideas for how swarming could be used to improve routing in Internet, although the financial investments in legacy systems have so far been too large for any large scale experiments to be done[6].

1.5 Possible military applications of swarming

Traditionally, military logistics and military command and control in general have been implemented in a top-down, centralized manner. Hence, the use of swarm intelligence for military purposes has so far been rather limited. Recent applications of it to sensor control and adaptation are described in (Gaudiano et al., 2003; Schrage and Gonsalves, 2003). In (Nygard et al., 2001), the authors use the collective intelligence of a swarm of UAVs to find and attack enemies. Possible future uses of UAVs are described in (James, 2000; Lua et al., 2003). Similar approaches to target acquisition by a network of sensors are taken in (Bowyer and Bogner, 2001; Brown et al., 2002).

Flocking has been used to model the behavior of military forces performing various kinds of missions (Carling et al., 2003), but the main focus of that paper was the analysis of the resulting communication network. More advanced models for simulating aircraft using swarming are (Trahan et al., 1998; Agassounon, 2003)

Ideas for high-level uses of swarming are considerably more limited. In a recent special section, Aviation Week and Space Technology (Hughes, 2003), have explored the possibilities of using swarming for network-centric warfare. Of particular interest here is the short article by Arquila and Ronfeldt.

They have also described the possible future uses of swarming on the battlefield in a longer book (Arquilla and Ronfeldt, 2000). An historical essay describing several cases where swarming tactics have been successfully used in military history is (Edwards, 2000). RAND also argues[7] for the need for future military organizations to be decentralized and use results from complex systems science.

In (Palmer et al., 2003), humans are used as a test-bed for swarming algorithms. An approach such as this enables researchers to study the behavior of agents that may not always follow the rules given to them, and also

[4] see, *e.g.*, http://www.santafe.edu/sfi/publications/wpabstract/200212067 and http://www.santafe.edu/sfi/publications/Bulletins/bulletin-winter98/swarm.html

[5] http://www.nutechsolutions.com

[6] http://people.cs.uchicago.edu/∼matei/P2P/BetterNetworks.html

[7] http://www.rand.org/publications/IP/IP193/

makes it possible to try to develop methods for reverse-engineering rules from observed behavior — this is a potentially very important application for Military Operations Other Than War.

Recently, a conference focusing on military application was sponsored by the US DoD's Command and Control Research Program (DoD, 2003). Unfortunately, most of the work described there is both very preliminary and low-level. An algorithm inspired by ant behavior was recently used for a subproblem of threat assessment (Svenson and Sidenbladh, 2003).

Air traffic control is an important problem for both civilian and military purposes. A collective intelligence approach to this is described in (Burdun and Parfentyev, 1999)

Mine detection is a very important problem. An approach to it that uses a large number of "ants" that collectively find and defuse mines is described in (Ferat and Kumar, 2002). Here, the authors assume that defusing a mine requires the cooperation of a certain number of "ants". A related work is (Yuigying et al., 2003), which uses ant algorithms to guide robots; here the number of robots that is needed to accomplish a certain task is determined using pheromone.

Chapter 2

Ant Colony Optimization

Certain types of ants communicate findings of food sources to others in their colony by deploying a pheromone (smell) trail in their own footpaths from the found food source to the hill (Bonabeau et al., 1999). The deployed pheromone evaporates over time, which means that short trails, on average, have stronger smell than long ones.

When other ants come upon the trail, they follow it with a higher certainty the stronger the smell. Since all ants deploy pheromones, all trails together form a complex energy landscape that changes over time. Paths of strong smell will form along the shortest trails from the hill to the major food sources. When a food source disappears, the smell path to this place evaporates (Bonabeau et al., 1999).

This efficient approximative method of finding the shortest path between interesting places in a time-varying environment has inspired a group of optimization algorithms jointly called Ant Colony Optimization (ACO). We will first describe the theory and fundamental ideas behind ACO. Then, applications of this technique to a number of types of optimization problems in the real world are discussed in sections 2.9.1 to 2.9.4.

2.1 Introduction

ACO is an approximative optimization algorithm, which means that it, in general (see Section 2.8), does not give a provably optimal solution to the optimization problem. Thus, it is not interesting for problems which can be solved exactly in an efficient manner. Instead it is useful on problems which are impossible or very time consuming to solve exactly (*i.e.*, NP-complete problems, see Appendix A). Such problems include difficult logistics problems, routing in telecommunication networks and scheduling of school classes. This chapter is a brief introduction to ACO. For a more extensive description, see (Bonabeau et al., 1999; Dorigo and Stützle, to appear; Maniezzo and Carbonaro, 1999).

The first ACO method, Ant System (AS) (Dorigo et al., 1996) was described for the famous NP-complete Traveling Salesman Problem (TSP). We will also use TSP to describe AS and ACO for three reasons:

1. It is a logistics problem, and this report is targeted towards logistic applications.

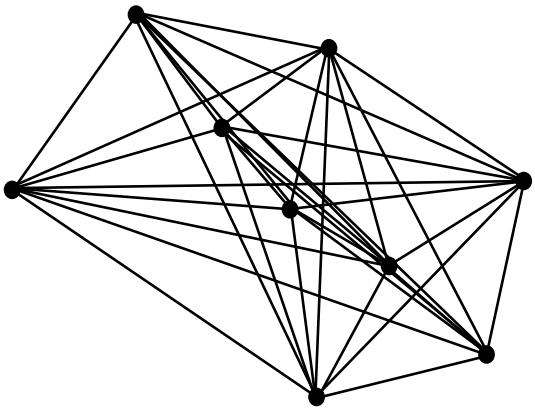
2. It is very easy to visualize.

3. It is often used as a benchmark problem to compare different algorithms.

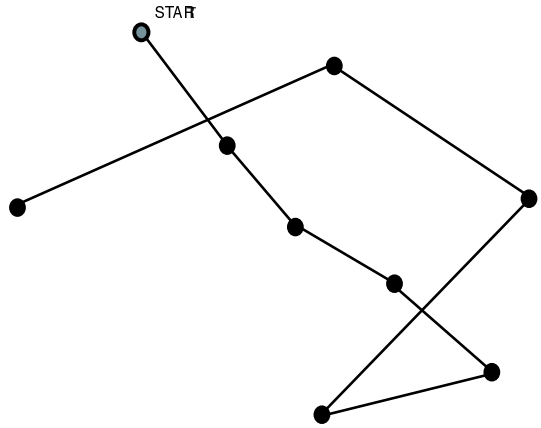
2.2 The Traveling Salesman Problem

A traveling salesman wants to visit N cities. The cities are connected by E roads, where the road between city i and j is of length d_{ij} (see Figure 2.1(a)). The salesman can only visit each city once, and must start start end end in the same city. The optimization problem is here to find the path, *i.e.* the sequence of cities, $[i_1, i_2, \ldots, i_N]$ which minimizes the total traveling distance $d = \sum_{k=2}^{N} d_{i_{k-1} i_k}$.

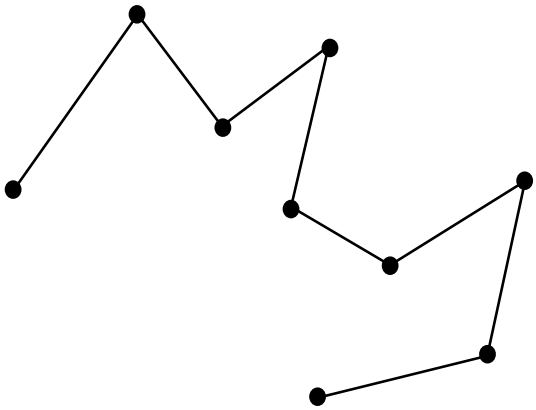
A first attempt at solving this problem approximately could be to use a greedy algorithm, see, *e.g.*, (Bonabeau et al., 1999). Start at city i_1. Select the city i_2 as the one closest to i_1 (with a minimal $d_{i_1 i_2}$). Repeat this until all cities are covered, avoiding cities already visited. This method will not always find the optimal path (see Figure 2.1(b,c)).



(a) A graph. Each node corresponds to a city.



(b) The shortest path as found by the greedy algorithm.



(c) The true shortest path through all cities.

Figure 2.1: The Traveling Salesman Problem. Each node in the graph corresponds to a city. What is the shortest path on which the salesman visits all cities exactly once? A greedy algorithm will not necessarily find the shortest path. Instead, it will find a path which initially has short inter-city distances.

2.3 Ant System

Inspired by the use of pheromone in certain species of ants, Dorigo et al. (1996) developed Ant System (AS). The idea is to let a number of simulated agents, or ants, build different solutions to the TSP by moving in a probabilistic manner between nodes (cities) in the graph. When a tour is completed, the ant deposits "pheromone", implemented as a weight, on the tour. Shorter tours receive more pheromone than longer tours.

When in city i, an ant decides which city j to move to depending on (Bonabeau et al., 1999):

1. Whether or not the city j has already been visited in the ant's tour. Remember that each city should be visited only once. Each ant k maintains a "tabu list" over cities that it has visited. From this list and from the graph, it can produce a list J_i^k over the cities that it is allowed to move to from i.

2. The inverse $\nu_{ij} = 1/d_{ij}$ of the distance between cities i and j. This is the *heuristic desirability* of choosing city i when in j, and corresponds to the local rule used in the greedy algorithm described above.

3. The amount of pheromone $\tau_{ij}(t)$ previously deposited between cities i and j. This is a measure of how successful this selection of route has been for other ants, and is updated over time t (where time can be interpreted as generation of ants).

Expressed in mathematical terms, the probability that ant k moves to city j when in i is (Bonabeau et al., 1999):

$$p_{ij}^k(t) = \frac{(\tau_{ij}(t))^\alpha (\nu_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha (\nu_{il})^\beta} \tag{2.1}$$

where α and β are parameters that control the influence of the trail length and the pheromone. If $\alpha = 0$ and $\beta = 1$, we get a stochastic greedy algorithm.

The amount of pheromone deposited on edge (i, j) by ant k after the completion of the tour is (Bonabeau et al., 1999):

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q/L^k(t) & \text{if } (i,j) \in T^k(t) \\ 0 & \text{if } (i,j) \notin T^k(t) \end{cases} \tag{2.2}$$

where $T^k(t)$ is the tour done by ant k, $L^k(t)$ is the length of this route, and Q is a parameter. Without pheromone evaporation, the ants will quickly get stuck in local minima (Bonabeau et al., 1999). Therefore, the pheromone map is updated as:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2.3}$$

where ρ is a parameter that controls the speed of pheromone evaporation and $\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t)$ is the sum of pheromone trails for all m ants.

While the first generation of ants will select both long and short routes through the graph, the following generations will converge to selecting the shortest routes. The question is whether it finds the optimal route, and how fast it converges. AS was tested against other approximative algorithms dedicated to solving TSP. The results (Dorigo et al., 1996) showed that for small problems (less than 30 cities) AS performed equally or better than other algorithms. However, for large problems, AS found good solutions quickly, but did not find the optimal solutions within a (large) bounded number of iterations t.

In the next sections, we will briefly review a number of suggested improvements to AS. This group of improved algorithms have been named Ant Colony Optimization (ACO) algorithms (Bonabeau et al., 1999).

2.4 Ant Colony System

Due to the failure of AS of finding the optimal solution to TSP on large graphs, Dorigo and Gambardella (1997a; 1997b) suggest an improved algorithm, called Ant Colony System (ACS).

In general, the changes in ACS are intended to make the ants' search more efficient to enable a fast solution to TSP on large graphs. In benchmark tests ACS performed well in comparison to other state of the art algorithms (Bonabeau et al., 1999; Dorigo and Gambardella, 1997a; Dorigo and Gambardella, 1997b).

The major differences (Bonabeau et al., 1999) between AS and ACS are outlined in the paragraphs below.

Transition rule. In ACS, ant k chooses city j to travel to according to the rule:

$$j = \begin{cases} \arg\max_{l \in J_i^k} \left(\tau_{ij}(t) (\nu_{ij})^\beta\right) & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \tag{2.4}$$

where q is a random variable uniformly distributed over $[0, 1]$, $q_0 \in [0, 1]$ is a parameter and $J \in J_i^k$ is a city chosen according to the probability distribution:

$$p_{iJ}^k = \frac{\tau_{iJ}(t) (\nu_{iJ})^\beta}{\sum_{l \in J_i^k} \tau_{il}(t) (\nu_{il})^\beta} \tag{2.5}$$

which is similar to the AS transition probability in Equation 2.1. A high q_0 means that the ants do not explore to the same extent as in AS - instead, they tend to exploit the previously deployed pheromone trails. When q_0 is low, the ants explore new tracks with a certain probability. The parameter q_0 can be tuned during algorithm execution, and fills the same purpose as the temperature parameter in simulated annealing.

Pheromone trail update rule. In AS, all ants update the pheromone map. To further direct the exploration of the trails in ACS, only the ant that found the shortest tour T^+ is allowed to deposit pheromone in each tour. Due to this, ACS is sometimes denoted an "elitist approach". Furthermore, no evaporation of pheromone occurs on edges

not belonging to T^+. This means that the pheromone map is intact save for the shortest tour which is updated in the same manner as were all tours in AS. The update rule is:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{2.6}$$

where (i,j) are edges belonging to T^+, ρ is a parameter that define rate of pheromone evaporation and $\Delta\tau_{ij}(t) = 1/L^+$, L^+ being the length of T^+.

Local updates of pheromone trail. The pheromone on the edges is also updated locally: Every time an ant travels on an edge (i,j), the pheromone concentration on that edge is updated as

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \rho\tau_0 \tag{2.7}$$

where τ_0 is the initial pheromone value. This will have the effect that ants are discouraged to go on edges frequently visited by earlier ants, making less visited edges more attractive. Thus, the ants tend to spread out, lowering the probability that the best route is left unexplored.

Use of candidate list. A candidate list is often used in algorithms designed to solve the TSP, and has been included into ACO (Dorigo and Gambardella, 1997a; Dorigo and Gambardella, 1997b). Each city i in the graph has a candidate list, which is a list of preferred cities to travel to from i. An ant first selects a city from the candidate list, only if there are no possible options left (all cities in the list have already been visited in the tour) will the ant select another city.

Local search. To further enhance the performance of ACS, a local search procedure (Dorigo and Gambardella, 1997a) is employed. The purpose of local search is in the case of ACS to bring each ant's tour to its local optimum. Dorigo and Gambardella (1997a) use 3-opt, an iterative method in which three edges at a time are switched until a local optimum is found.

2.5 $\mathcal{MAX} - \mathcal{MIN}$ Ant System

Independently from ACS, Stützle and Hoos (2000) present another algorithm derived from AS, the $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS). The primary goal with $\mathcal{MM}$AS (as with ACS) is to prevent stagnation, *i.e.*, prevent the ants to get stuck in a suboptimal local minimum. $\mathcal{MM}$AS performs comparably with ACS on large TSP problems. The main differences between AS and $\mathcal{MM}$AS are described in the following paragraphs.

Pheromone trail update rule. As in ACS, only the ant that found the shortest route T^+ is allowed to update the pheromone map (which also makes $\mathcal{MM}$AS an elitist approach). The update rule, similarly to ACS, is:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2.8}$$

where (i,j) are edges belonging to T^+, ρ is a pheromone evaporation parameter and $\Delta\tau_{ij}(t) = 1/L^+$, where L^+ is the length of T^+.

Pheromone trail limits. To prevent one trail from being overly enforced early on in the search, the pheromone values on all edges (i,j) is limited to the interval $[\tau_{\min}, \tau_{\max}]$.

Pheromone trail initialization. While in AS, the trails are initialized with a low start pheromone value τ_0, all pheromone trails are here initialized to a maximal start value, $\tau_{\max}$. This encourages early exploration of the whole graph, so that the global optimum is reached faster.

Pheromone trail smoothing. Stützle and Hoos also suggest smoothing of pheromone trails, which would be useful to any elitist approach. Mathematically, the smoothing can be expressed as:

$$\tau^*_{ij}(t) = \tau_{\max} - \delta(\tau_{\max} - \tau_{ij}(t)) \tag{2.9}$$

where δ is a parameter $\in (0,1]$ where 1 corresponds to no smoothing. The benefit of smoothing in this way is that exploration is encouraged since paths with low pheromone value become more probable choices.

2.6 Other improvements

Bullnheimer et al. (1997a) introduce another elitist strategy which they call AS_{rank}. The ants are ranked according to solution quality (trail shortness in the case of TSP). Instead of using only one elitist ant, the σ best ants are here allowed to deposit pheromone. The pheromone trails are scaled so that no ant can deploy a higher amount of pheromone than an ant higher in the ranking. The results of AS_{rank} were significantly better than with AS, but no comparisons to ACS or $\mathcal{MM}$AS were made.

Other improvements generally build on ACS. Montgomery and Randall (2002) builds on ACS by adding the concept of anti-pheromone, thus discouraging ants from following the bad trails along with the encouragement of ants to follow the good trails. Roux et al. (1999) augment ACS by using a Tabu search instead of the 3-opt search suggested by Dorigo and Gambardella (1997a). To prevent stagnation early on in the search, Fidanova (2002) suggests additional reinforcement of pheromone to guide the search to unexplored areas. The performance is comparable to $\mathcal{MM}$AS. Merkle and Middendorf (2002a) introduce a normalization procedure for the pheromone.

ACO algorithms in general rely on a number of parameters such as the number of ants, the initial pheromone value and the speed of pheromone evaporation. By optimizing these parameters, the performance of the algorithm can improve significantly (Botee and Bonabeu, 1998). Botee and Bonabeu (1998) and, independently, Pilat and White (2002) use genetic algorithms to optimize the parameters.

Merkle and Middendorf (2002d) introduce a method for handling permutation problems using ant colony optimization. They find that dividing the problems into subproblems and possibly also freezing the pheromone gives better results for their problem.

The process of laying pheromone that real ants do is called *stigmergy*. Another possible way for social insects to communicate is by *recruitment*, *i.e.*, asking a neighbor to do the same thing you do. A method based on this has been introduced (Dro and Siarry, 2002). The difference is that in one extreme case of the method they present, ants communicate one-to-one instead of one-to-many as in the normal algorithm.

The Best-Worst Ant Colony System (Cordón et al., 2002) is a combination of ant colony optimization and evolutionary computing. It introduces changes to the pheromone updating and restarts all ants when the difference between the best and worst solutions found is small. In a similar vein, (Acan, 2002) describes a combination of genetic algorithms and ant colony optimization.

A problem with any heuristical search method is that there are always many possible moves to consider in the search-space. In an attempt to overcome this difficulty, Randall and Montgomery (2002) investigate the benefits of restricting the candidate set of moves. They introduce several different methods of restriction, and find that different methods work best for different problems. This is a preliminary study; future work both on finding a general restriction that work for many problems and on finding restrictions for specific problems is needed.

2.7 Parallel implementations

The inherent structure of the ACO algorithms, with a large number of agents working relatively autonomously, makes them suited for parallel implementations (Piriyakumar and Levi, 2002; Randall and Lewis, 2002). Randall and Lewis (2002) suggest several different parallelization strategies depending on the problem at hand (*e.g.*, how much communication between ants is necessary).

Another parallel implementation of ant colony optimization is described in (Merkle and Middendorf, 2002b). The authors change the behavior of the algorithm slightly and manage to find an algorithm whose running time is linear in the input size.

2.8 Will the algorithms always find the global optimum?

ACO methods are heuristic in the sense that they do not originate from mathematical deduction, but from studies of the behavior of real ants. However, there are empirical indications (Bonabeau et al., 1999; Dorigo and Gambardella, 1997a; Dorigo and Gambardella, 1997b; Stützle and Hoos, 2000) that certain ACO algorithms always find the global optimum even in very large problems. The question is what mathematical properties give this stable performance. Can it be mathematically proven that an ACO algorithm is bound to find the optimal solution?

One way to achieve this is to map an ACO algorithm to a well known algorithm. Birattari et al. (2002) develop the Ant Programming framework which encompasses ACO. This wider framework makes it possible to compare ACO directly to reinforcement learning algorithms, which are well studied. Meuleau and Dorigo (2002) study the relationship between ACO and stochastic gradient descent, which also is extensively studied. They show that

some ACO algorithms can be directly reformulated as stochastic gradient descent algorithms in the pheromone trail space.

Gutjahr (2000) develop a Graph-Based AS, which can be seen as a restricted version of ACS. The restrictions are intended to make sure that the algorithm is convergent. However, the empirical performance of this algorithm is not tested (Stützle and Dorigo, 2002). To this end, a convergence proof was developed by Stützle and Dorigo (2002) for the two well-known ACO variants $\mathcal{MM}$AS and ACS. The proof shows that optimal solution will be found with a probability $1 - \epsilon, \epsilon \rightarrow 0$ using any of these algorithms. Gutjahr (2002) has after this published a proof that an extended version of Graph-Based AS will converge with probability 1, a stronger claim.

Merkle and Middendorf (2002c) take another direction. While others (Gutjahr, 2000; Gutjahr, 2002; Stützle and Dorigo, 2002) have proven convergence for elitist approaches (where only one of a few ants may deposit pheromone), Merkle and Middendorf suggest a deterministic model of the ACO dynamics, which makes it possible to study analytically a more general ACO algorithm. They show that their deterministic model corresponds closely to an ACO algorithm in simulations.

2.9 Applications

The ant colony optimization algorithm has been successfully applied to a number of problems. In this section we briefly describe some of them, with an emphasis TSP and routing, since these problems appear naturally in logistics.

Ant colony optimization has a number of very attractive features that makes it a natural candidate to apply to any new optimization problem. Among these features are

- It is easy to implement.
- It is a generic algorithm, that works for any optimization problem that can be formulated in terms of an objective function.
- There are mathematical proofs that it converges on the exact solution for some problems.
- It is faster than other approximative methods for solving optimization problems.

2.9.1 Traveling Salesman Problem

Many difficult logistics problems can be described in the form of a TSP. A number of TSP extensions have been made, including probabilistic TSP and dynamical TSP. These are probably more relevant than TSP to regard if interested in a logistics application.

The probabilistic TSP can be described as a TSP where each node has a cost associated with not being visited. The objective function – the function to minimize to find the optimal solution – takes this cost, as well as the length of travel into regard. In the original TSP, this cost would be infinite, *i.e.*, the solution requires a visit to all nodes. Bianchi et al. (2002a; 2002b) are the first to apply ACO to probabilistic TSP. They compare two approaches, ACS and pACS. The latter approach updates the pheromone slightly differently than ACS, using a statistical measure of tour length instead of the actually observed length. When the probabilistic TSP is more "TSP-like" with high costs, ACS perform better, otherwise pACS is more suited to solve the problem. Branke and Guntsch (2003) proposes two ways of improving the ACO performance on probabilistic TSP. Firstly, they suggest making the computation of the tour length more approximate to speed up computations. Secondly, they suggest using problem-specific heuristics to guide the optimization.

Up to now, all work on ACO described in this chapter has concerned static TSP formulations. However, as discussed in chapter 1 there is great potential in using swarm methods to solve dynamical problems.

Many time-varying logistics problems can be described as a dynamical TSP, *i.e.*, a TSP in which the objective function (the distance between nodes) changes over time. An application of this is traffic jams (Eyckelhof and Snoek, 2002), in which the travel times between cities change over time. Angus and Hendtlass (2002) applies ACS to a dynamic TSP, and compares the computational cost of letting the pheromone map update continuously over time against starting optimization from scratch multiple times. They find that the pheromone map is able to adapt continuously to the changing underlying conditions, and that the computational cost is lower than that of performing the whole search over and over. Eyckelhof and Snoek (Eyckelhof and Snoek, 2002) instead adapts AS to the dynamical problem, and introduce a number of improvements such as "shaking", a type of smoothing of the pheromone map. Guntsch and Middendorf (2002) suggests transferring a set of solutions over time, instead of the whole pheromone map. This could speed up computations.

Other applications of ACO to logistics problems include a continuous problem of layout optimization (Sun and Teng, 2002) which can be formulated as a form of TSP, a generalized minimum spanning tree problem (Shuy et al., 2003) which is an NP-complete problem just as TSP, and design of a water distribution system (Maier et al., 2003). A combination of swarm optimization and genetic algorithms has been used to optimize the layout of highway networks (Jha, 2002). A related application is (Hoar et al., 2002), where swarm simulation is used to minimize traffic congestion in cities.

An application that seems very similar to spare parts optimization is described in (Silva, Runkler and Sousa, 2002). Here the task is to optimize the logistics processes at Fujitsu-Siemens Computers. According to the paper, the results were successful: a new, improved scheduling process was found. The problem they solve can be described as minimizing stored stock while simultaneously delivering goods to consumers at the correct time. As in the spare parts optimization problem (Alfredsson, 1997), a Poisson process is used for determining demand. Since there is not an exact one-to-one correspondence between the two problems, the approach taken here must be modified somewhat before tackling the spare parts optimization problem. The approach taken in (Silva, Runkler and Sousa, 2002) nevertheless seems very promising also for our problem.

An interesting application of scheduling to determine positions using a minimum number of GPS-receivers is described in (Saleh, 2002)

2.9.2 Routing

Routing denotes the problem of directing information flow in a network so that it reaches its recipients as fast as possible. The advent of the Internet has made it a very important problem. The problem would be trivial if we could connect every site to every other, but this is not possible except for very small network sizes. Instead, messages containing information must be transmitted via other sites. Since most sites are connected to several others, they must chose which one(s) to transmit a packet of information through. The most straight-forward approach would be to transmit it to every neighbor, but this would lead to serious bandwidth problems. Instead, an algorithm for choosing which neighbor to send it to must be used.

Similar problems are faced by distribution companies that must send its buses or trucks so that all customers are serviced without too large delays. The difference between this problem and the Traveling Salesman Problem is that a milk-delivery company, for instance, in general has several delivery trucks and does not always need to enforce the constraints of visiting each customer at most once.

Ant algorithms for routing in computer and communication networks have been studied by (Kassabalidis et al., 2001; Randall and Tonkes, 2001; Okino, 2002; Garlick and Barr, 2002; Sim and Sun, 2002; Suiliong et al., 2002; Denby and Hegarat-Mascle, 2003), while the problem of loops in routing is described in (Canright, 2002). This paper argues that ant-based routing algorithms will not form stable loops. A paper by Wittner and Helvik (2002) studies the opposite problem where one wants to find two paths from a source node to the destination node. In some networks, this is necessary from a security point of view, when it is crucial that there exists a backup path for information. The algorithm they use deviate substantially from “traditional” ant methods, and needs to be further refined before it can be applied to real networks.

(Bullnheimer et al., 1997b) solve the routing problem where there is one central depot and a number of customers that must be visited by exactly one vehicle. Each customer is associated with a service time (keeping the vehicle busy for that time) as well as a demand giving the amount of materiel it needs. This problem is hence very similar to the spare optimization problem described in section 1.3. One wants to plan movements of a number of vehicles so that the total length travelled is minimized, while meeting the constraints of satisfying each customer. For the example problems solved in (Bullnheimer et al., 1997b), their ant algorithm managed to find the previously known best solutions but did not improve on them. They suggest several options for improving their algorithm, *e.g.*, by combining it with a good local search method in a similar way as is often done when solving TSP using ants.

Introducing time constraints on when particular customers should be serviced leads to further complications. This problem has been tackled by (Ellabib et al., 2002). This problem combines the difficulties of both routing and scheduling, and the solution proposed by Ellabib et al should be augmented with a local search procedure to further improve their results.

An interesting combination of ant colony optimization and particle swarm optimization is presented as future work in (Reimann et al., 2002). This paper also solves a routing problem where there are time constraints. Reimann et al combine their method with local search, and find better results than Ellabib et al. The most interesting aspect of this paper, however, is their suggestion to use a particle swarm-like method of adjusting each ants individual parameters during the optimization run.

2.9.3 Scheduling

The archetypical scheduling problem is how to schedule classes in a school so that no student is required to be at two different places at the same time. In the literature, the problem is sometimes mapped to a graph coloring problem (Costa and Hertz, 1997). Here, the task is to color all vertices in a graph so that no edge joins two vertices of the same color. Allowing such edges but associating them with a cost leads to the set partitioning problem discussed above.

The $\mathcal{MAX} - \mathcal{MIN}$ Ant System discussed in section 2.5 is applied to such a problem in (Socha et al., 2002). The method compares favorably to other local search methods, and shows that the ant system can handle problems with a large number of constraints. Blum (Blum, 2002) shows that several improvements of the $\mathcal{MAX} - \mathcal{MIN}$ algorithm can be made. By initializing the pheromone randomly at each restart of the algorithm, the probability of reaching the same solution several times is reduced. Hence, this increases the explorative capabilities of ants. Another improvement comes from retaining a set of elite solutions, convergence is then tested against this set instead of just against the best found solution so far.

For manufacturing industries, it is important to schedule production in the most efficient manner. Often the same machines are used for producing different types of goods, but since there is often a setup time before a machine can be switched to another mode, it is important to optimize the allocation of machines. There have been several attempts to solve this using ants-inspired methods (Bauer et al., 1999; Merkle and Middendorf, 2001; T'kindt et al., 2002; Wang and Wu, 2002; Gravel et al., 2002; Gagné et al., 2002; Vogel et al., 2002; Blum and Sampels, 2002; Nouyan, 2002; Bautista and Pereira, 2002; Merkle and Middendorf, 2003).

A comparison between ant colony optimization and local search for this type of problems can be found in (Gottlieb et al., 2003).

Other problems related to scheduling are studied in (Cincotti et al., 2003; Mohanty et al., 2003).

Load balancing is a problem that is related both to routing and scheduling. Here, the objective is to maintain functionality in a network when the amount of traffic in it reaches so high levels that optimal paths cannot be used for all transmissions. In (Schoonderwoerd et al., 1996), a comparison between an ant-based method and some traditional algorithm is made. The authors find that their method leads to fewer lost transmission than the others.

2.9.4 Other applications

The satisfiability problem is one of the corner-stones of computational complexity theory (see Appendix A). An ant colony optimization method for solving it is described in (Schoofs and Naudts, 2002).

Another important NP-complete problem that is related to scheduling problems it set covering. Here the task is to partition a set into disjoint subsets so that the sum of the costs in each subset is as small as possible. An ant-based approach to it is described in (Maniezzo and Milandri, 2002). (Fenet and Solnon, 2003) apply ant optimization to the max clique problem.

A combination of the ant system and local search is used for the set covering problem in (Rahoual et al., 2002). The authors also describe a parallel implementation of it. The set covering problem consists of finding the minimum number of vertices in a graph that need to be populated in order to watch all nodes, given that an agent at node i can watch all neighboring nodes. A variant of it is the Museum Guard problem, where the task is to assign guards to rooms in a museum. The paper finds that the combination with a local search method is crucial for getting good results. The results here could be used for determining optimal placements of supply sites or watchposts in military operations other than war, for instance.

Clustering is an important problem in many applications that need to group similar objects. Military applications include force aggregation, where one wants to find, *e.g.*, platoons of vehicles. It can also be used to find Web pages that are similar, or to extract information from text. Papers that solve this problem using swarm optimization include (Hoe et al., 2002; Bin et al., 2002; Folino et al., 2002; Yang and Kamel, 2003)

Game theory has application far beyond the often very silly games often used to describe it. In (Branke et al., 2002), the results of using ant colony optimization to develop strategies for Nim and Tic-Tac-Toe are presented. Among other things, game theory has uses in sensor adaptation and allocation for information fusion (Johansson et al., 2003).

Other, rather esoteric, applications of ant colony optimization include image processing (Vallone, 2002; Ouadfel et al., 2002), area covering by real robots (Svennebring and Koenig, 2002), and protein folding (Shmygelska et al., 2002). It has also been used for time-series prediction of pollution (Lu et al., 2003). Many learning problems can be solved using some sort of Bayesian network (Pearl, 1988). Ant colony optimization has been used to optimize the process of designing and using these (Gámez and Puerta, 2002; de Campos et al., 2002), and also for data mining (Parpinelli et al., 2002) and task coordination (Bianchi and Costa, 2002).

Chapter 3

Particle Swarm Optimization

In 1995 Kennedy and Eberhart introduced the Particle Swarm to simulate social interactions. The purpose was to demonstrate the idea of individuals gaining evolutionary advantage by sharing information. Instead of just blindly running through life reflecting only over your own personal experiences, taking account of the behaviors and success rates of surrounding people could be useful. By imitating the most successful of your fellow companions it is plausible that you could reach a more advantageous position. This way of sharing experiences by imitation is commonly believed to be one of the key ingredients of intelligent life. Considering the human superior ability to imitate, this could give us reason for the sudden success of our own species (Kennedy et al., 2001).

The Particle Swarm model is not only promising in the field of social simulations. A stripped-down version of the model was presented as a novel and generic optimization technique, the Particle Swarm Optimizer (PSO). The model is as simple as it is powerful. Consider a set of individuals, a *population*, moving around in a search space. Every point in this space is associated with a certain score, corresponding to the evolutionary concept of fitness (or utility). In order to explore the space efficiently and maximize their own score, each individual is equipped with two special features. First, it has a memory of the best point it has visited so far, *i.e.*, its personal experience. Secondly, it can look at neighboring individuals and compare their positions and scores with its own. In order to find new positions these features are combined to yield an acceleration towards both the individual's own previous best and the best of its neighbors.

As an optimization technique the PSO is known to be both simple to implement and robust. It does not need a differentiable objective function and can easily be adapted to new types of problems. The search mechanism is completely heuristic, which according to the inventors should be seen as an additional feature: "We don't agree that human thinking is faulty; we suggest on the contrary that formal logic is insufficient to solve the kinds of problems that humans typically deal with." (Kennedy et al., 2001)

3.1 The basic PSO algorithm

The PSO comes in many different variants. There are, however, seldom any major conceptual differences. We will present the most common one, used for optimizing a real-valued objective function, $f(\vec{x})$, defined on a real-valued D-dimensional search space, i.e. $\vec{x} \in \Re^D$. The algorithm is initialized with a swarm of n particles randomly distributed over the search area. The swarm is then set loose in the sense that for a number of iterations each particle moves and updates its velocity in an autonomous manner striving for the optimal position. The updating rules for the ith particle are described as follows:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{3.1}$$

$$\vec{v}_i(t) = w\vec{v}_i(t-1) + c_1\varphi_1\big(\vec{p}_i - \vec{x}_i(t-1)\big) + c_2\varphi_2\big(\vec{p}_g - \vec{x}_i(t-1)\big) \tag{3.2}$$

The new position of the particle is given by its old position plus its current velocity. The velocity in turn, is determined by a linear combination of three terms. The first is simply the speed from the previous time step, weighted with an *inertia weight*, w. The second term represents the particle's desire to return to its previous best position, which is denoted $\vec{p}_i$. Finally, the third term gives the attraction of seeking the best position among the particle's topological neighbors. We denote this shared high-score $\vec{p}_g$, where index g stands for global. The impact of the last two terms are controlled by two constants c_1 and c_2 together with the random numbers φ_1 and φ_2, taking values (uniformly) between 0 and 1.

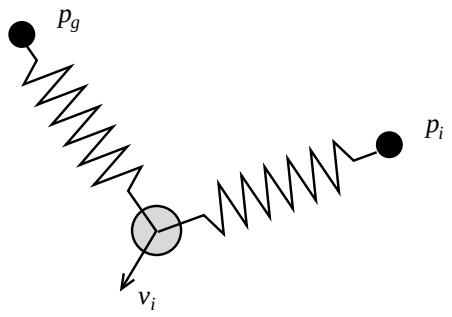


Figure 3.1: The forces acting on particle i are proportional to the distance to the particle's previous best, $\vec{p}_i$, and to the best of its neighbors, $\vec{p}_g$, multiplied with random numbers. The effect is the same as if it was attached to two springs with random varying spring constants.

Note that equation (3.2) is simply Newton's second law,

$$m\ddot{\vec{x}}_i(t) = \vec{F}, \tag{3.3}$$

and the terms on the right hand side can be seen as the effective force acting on particle i (Figure 3.1). Note that the inertia w is thus a kind of inverse mass.

Depending on the specific problem and parameter choices it is sometimes necessary to add a constraint on the size of the velocity. If not, the particles may gain too much momentum and spread out to infinity. A V_{max} parameter is therefore introduced that cuts the velocities according to:

$$v_{ij}(t) = V_{max}, \qquad \text{if} \quad v_{ij}(t) > V_{max} \tag{3.4}$$

$$v_{ij}(t) = -V_{max}, \qquad \text{if} \quad v_{ij}(t) < -V_{max} \tag{3.5}$$

The stopping criterion for the PSO algorithm is user defined. It commonly consists of a maximum number of allowed iterations complemented with some sort of convergence criterion. This can be, *e.g.*, when no improvement in best position has been found in a certain number of time steps, either for the whole swarm or for each individual separately.

The neighborhood connected to and influencing a single particle can take on many different shapes. Conceptually one distinguishes between the *gbest*- (global) and the *lbest*-neighborhoods (local). In the gbest all particles are connected to one another and consequently share the same p_g, the current best position in the whole swarm. In the lbest the neighborhood is a subset of the swarm. The choice of subset varies, but the most basic version is to choose the k-nearest neighbors defined on a chain-lattice topology (the 2-nearest neighbors of the ith particle are the ones indexed $i-1$ and $i+1$, modulus n. These are usually not the closest particles in a geometric sense).

3.1.1 Parameters

Most results for parameter settings in the PSO are based on empirical investigations. In most cases the PSO turn out to be robust. Reasonably good parameter selections do not seem very hard to find.

The ratio of the parameters c_1 and c_2, sometimes referred to as the *trust* parameters, determine to what extent the individuals of the swarm should rely on own experience, called *cognitive learning*, or that of others, *social learning*. Experiments have shown that the trust parameters do not play an essential role for the convergence of the PSO. In some cases though, evidence exist that a slight bias towards cognitive influence can provide better and faster solutions (Parsopoulos and Vrahatis, 2002). The most popular values are $c_1 = c_2 = 2$.

More effort has been put on investigating the influence of the inertia weight, w. It was introduced to alleviate some convergence problems encountered by early versions of the PSO (Shi and Eberhart, 1998a). In some cases the PSO showed great global exploring capabilities, but very poor performance when closing in on an optimum. In other cases it was the opposite and the algorithm converged prematurely to a non-global optimum. The inertia weight presented a solution. Decreasing its value reduces the impact from a particle's previous velocity, making the swarm more sensitive to influences from previously found optima, thereby increasing local exploitation. This ability to tune the balance between local and global search is one of the major advantages of the PSO, and makes it well suited to handle complicated objective functions. A fruitful approach for making the tuning of w problem

independent is to adopt an annealing scheme (Shi and Eberhart, 1998b). Starting with a large w and then gradually lower it will yield an initially extensive search that later turn into a more and more focused local exploitation (compare simulated annealing (Kirkpatrick et al., 1983)).

There is no recipe in the literature for making a proper choice of the V_{max} parameter. Shi and Eberhart (1998b) have empirically determined a relationship between V_{max} and the inertia weight. On one specific function they found that a larger V_{max} worked best with a smaller inertia weight and the other way around. The actual values are problem dependent and usually determined by trial and error.

Recently a more thorough theoretical approach for determining criteria for PSO parameter settings has been undertaken (Clerc and Kennedy, 2002; Trelea, 2003). Rearranging the equations of motion and neglecting the stochastic parts makes it possible to analyze the dynamical system of each particle. The characteristics of the system's eigenvalues, which depend on the choice of parameters, determine if and how the particle will converge. Experiments support that the results also approximately apply to the fully stochastic system. Hence, starting from such calculations the trade-off between global and local exploration can be determined independent of the problem at hand.

Although these results are promising the effect of the social interaction remains to be theoretically examined. The number of particles and the choice of a particle's topological neighborhood play a vital role in the overall convergence properties. So far, these have been chosen more or less *ad hoc* (ranging from at least 10 to 300) and the only experience is that more particles give a more extensive search but is computationally more costly. Kennedy has done a study to clarify the importance of the different neighborhood topologies (Kennedy, 1999). He proves that expanding a local neighborhood with a few randomly chosen connections, *short-cuts* (cf. Small-World[1]), can lead to a faster and better search. The ideal number of short-cuts to add seems to be dependent on the objective function.

3.1.2 Similarities to Evolutionary Computing

Particle Swarm Optimization is sometimes referred to as an evolutionary algorithm. Obviously the social context in which the idea took off deeply resembles the darwinistic concept of "survival of the fittest". Each particle in the swarm strives for an optimal fitness score to attract fellow particles who have been less successful in their search. However, there is no actual selection or reproduction process in the Particle Swarm. In contrast to traditional Evolutionary Computing methods, such as Genetic Algorithms (GA) (Mitchell, 1996) and Evolutionary Programming (Koza, 1992), all individuals of the population survive and continue to take part in the ongoing search. Nevertheless, the sharing of information through social learning does imitate the effects of reproduction. In GA *cross-over* is a common strategy for reproduction. Two individuals mix their states to produce new hybrid offspring. This is very similar to a particle being attracted to its superior neighbor.

A second key concept of evolution is mutation. Mutation allows an individual to move to any nearby location with some probability. Although not explicitly present in the Particle Swarm the stochastic terms in equation (3.2) work in a similar manner. The particle moves around in search space with a certain amount of randomness. However, unlike mutation the update equations of the PSO may sometime put restrictions on the direction of a change. If all terms in (3.2) pull in a similar direction, exploration in the opposite direction is impossible.

Some authors argue that these sometimes halting similarities are not enough to make the PSO a full member of the Evolutionary Computing paradigm (Nunes de Castro, 2002). Kennedy and Eberhart (2001) on the other hand claim that they are enough. They also note that the Particle Swarm possesses other features even more crucial to evolution, such as self-organization and emergent behavior.

3.2 Extensions and performance

The relatively large number of scientific papers dealing with Particle Swarm Optimization reveals the fact that it has left at least its most immediate infancy. Much activity is still focused on quite basic aspects of the method, but the diversity of applications to which it has been applied confirms the wide-spread popularity of the PSO. The performance reported is almost everywhere very pleasing, except for some early papers when the method still suffered of growing pains. Due to the inherent simplicity, the algorithm is also very rewarding to extend and adapt to cope with new problem categories. A binary version was introduced early and a huge number of smaller modifications are suggested throughout the literature (Kennedy et al., 2001).

[1] Small world networks (Watts, 1999) are graphs that have both short diameters and high degrees of clustering. They have been studied extensively the last years as models of various social networks (Newman et al., 2001); together with so called scale free graphs they represent important extensions of classical random graph theory (Bollobás, 1985). Graphs such as these have been studied in the FOI project Metanet (Carling and Carlsen, 2002; Carling et al., 2003).

3.2.1 Benchmarking

Since it is probabilistic and heuristic, the PSO has many advantages compared to traditional deterministic optimization techniques. It is in no need for differentiable (or even continuous) objective functions and it possesses great capabilities for escaping local minima, features shared with many other heuristics. Benchmarking of the PSO is based on comparing the PSO with these heuristics, mainly Evolutionary Algorithms, but in some cases also with deterministic annealing (DA) and simulated annealing (SA) [2].

In an influential paper, Angeline (1998a) compared the performance of the early PSO with a Genetic Algorithm (GA) on four non-linear functions. The PSO was the fastest to find a near optimum solution, but in the long run it did not find as accurate solutions as the GA (this was before the introduction of the inertia weight). The result inspired a lot of work on improving the convergence, some of which are briefly mentioned in section 3.2.2. Similar convergence behavior has been detected on other testbeds (Boeringer and Werner, 2003).

The binary PSO has also been compared with a set of different GA. Using a multimodal problem generator the PSO was shown to outperform its competitors on almost all problems. It was also least sensitive to increasing problem dimensionality and modality (number of local optima) (Kennedy and Spears, 1998).

In (Tillett et al., 2003) the PSO and simulated annealing are compared on a clustering problem. The PSO is reported to be both faster and more accurate. The same relationship is claimed when comparing the PSO to GA on a number of Task Assignment problems (Salman et al., 2001; Salman et al., 2002). On the same testbed DA and SA were the fastest but didn't find at all as good solutions as the PSO and GA.

In conclusion the overall performance of the PSO seems very satisfactory. Although more comparative studies are needed, the algorithm so far proves to be very competitive. For many problem instances it outperforms other more well-known heuristics.

3.2.2 Improving convergence

Numerous attempts to improve the overall convergence properties of the PSO can be found in the literature. Many of them are influenced by the field of evolutionary computing. Angeline (1998b) introduces selection as a new feature in a hybrid PSO variant. The resulting algorithm shows improved performance on some functions but not on all. In a paper by Xie et al. (2002b) mass extinction of particles in certain time intervals is used to prevent premature convergence. This is also accomplished on three benchmark functions, but at the expense of introducing a new parameter. A third algorithm inspired by evolution is a hybrid of PSO and Predator Prey Optimization (Silva, Neves and Costa, 2002). An additional population, the predators, is added to the original swarm, the prey. The predator particles are attracted by the best particle in the prey swarm. The prey particles on the other hand are repelled by the presence of predators, giving rise to interesting dynamics. By controlling the strength of the attraction/repulsion the swarm can be kept from converging too soon and hopefully avoiding sub-optimal local minima.

When a particle comes close to the currently best global position there is a risk that it stagnates and stops contributing to the search. A method to avoid this is proposed in (Xie et al., 2002a). By identifying the inactive particles and re-initializing them the social diversity of the swarm is maintained, leading to improved average performance. Another risk with stagnation is that the local exploration around the global best is slowed down. This can be solved by restarting the best particle at the global best in each iteration and give it a small kick in a random direction (van den Bergh and Engelbrecht, 2002). The method is guaranteed to always find at least a local optimum.

An approach similar to that of a gradually decreasing inertia weight is proposed in (Fan, 2002). An adaptive scaling factor is introduced that restricts the maximum velocity v_{max} more and more in order to gradually move from global to local search. The algorithm is reported to statistically outperform the original one.

Inspired by the social-psychological concept of *stereotyping* Kennedy looks into the effect of clustering in the particle swarm (2000). The swarm seems to improve performance when the individual best, p_i, is replaced by a clustered group's central position, the stereotype of the group.

A very elaborate hybrid algorithm is presented by Krink and Løvberg (2002). Depending on where in the search space a particle is they let it change identity between a PSO particle, a GA individual and a simple hillclimber. The method performs very well on common benchmark functions and the authors point out many directions for further research on this type of hybrids.

[2] For descriptions of DA and SA, see (Peterson and Söderberg, 1989) and (Kirkpatrick et al., 1983) respectively

3.2.3 Discrete and constrained optimization

To find an algorithm for a specific category of problem one wants to solve, some adjustments and customizations usually have to be made. In real-world applications, *e.g.*, the search space of feasible solutions is often limited by some (m) constraints

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, ..., m \tag{3.6}$$

For the PSO, different techniques to ensure convergence in a valid area have been proposed. In (Venter and Sobieszczanski-Sobieski, 2003) a Lagrangian penalty term is suggested,

$$\tilde{f}(\vec{x}) = f(\vec{x}) + \alpha + \beta \sum_{i=1}^{m} max[0, g_i(\vec{x})] \tag{3.7}$$

where α and β are the positive Lagrangian multipliers [3]. In addition, for particles lost in an infeasible region, a slight modification of equation (3.2) is used. The first term is omitted so that the particle only will be influenced by previous optima, all of which are feasible points.

When the division of the problem space into feasible regions is more complicated, the penalty approach is not always automatically applicable. A more brute method for keeping restive particles inside legal zones is to simply move them back to either the border of the feasibility zone (Abido, 2002) or their previous best positions (El-Gallad et al., 2002). Another approach to this is presented in (Hu et al., 2003a).

Another modification attempt is to use the PSO for discrete optimization. Inherently the PSO is a method for solving continuous problems, but in (Venter and Sobieszczanski-Sobieski, 2003) a relaxation approach for adaptation to a discrete problem is suggested. By simply rounding off variable values in a continuous space to the closest in the discrete counterpart, the algorithm seems to work well. For a discussion on relaxation techniques, see (Gustafsson, 1994).

3.2.4 Multi-objective optimization

An interesting type of problem with many important applications, is multi-objective optimization. This is a particularly difficult problem because competing objective functions have to be optimized simultaneously. The optimal solution is generally not clearly defined and often there might be a whole set of interesting solutions. Three papers suggest different modifications of the PSO for using it as a multi-objective optimizer. In (Hu and Eberhart, 2002b) and (Parsopoulos and Vrahatis, 2002) a single optimum is sought, while in (Brits et al., 2002) the swarm is configured to converge to a many-point solution. (Mostaghim and Teich, 2003) look at the problem of determining which particle position to use as the global best for multi-objective optimization. (Hu et al., 2003b) introduce an algorithm that stores several good positions for the objective-functions.

3.2.5 Adapting to changing environments

All variants of the PSO discussed so far have been for static objective functions. What if we not only would update the position and speed of the swarm but simultaneously also would alter the environment? Carlisle and Dozier reported (2000) that the original swarm formulation failed on solving problems with continuously changing objective functions. As the optimum moved away the swarm tended to get stuck around their individual and global memories, $\vec{p}_i$ and $\vec{p}_g$. To cure this Carlisle and Dozier suggested a new algorithm, called Adaptive PSO (APSO). By clearing the memory of the swarm, either at constant time intervals or when the environment has changed more than some allowed amount, the swarm can more easily adapt to a new optimum. A different approach was used in (Hu and Eberhart, 2002a). Instead of clearing the memory of the whole swarm a portion of the population was re-randomized. Best results were obtained with a medium portion of the particles being re-randomized, but no comparison between the suggested methods was done.

Methods like these must be used for a dynamically changing optimization problem.

3.3 Applications

In the last couple of years the Particle Swarm Optimizer has reached the level of maturity necessary to be interesting from an engineering point of view. It is a potent alternative optimizer for complex problems and possesses many attractive features such as:

[3] For an introduction to Lagrangian relaxation, see (Gustafsson, 1994)

- Ease of implementation. The PSO is implemented with just a few lines of code, using only basic mathematical operations.
- Flexibility. Often no major adjustments have to be made when adapting the PSO to a new problem.
- Robustness. The solutions of the PSO are almost independent of the initialization of the swarm. Additionally, very few parameters have to be tuned to obtain quality solutions.
- Possibility to combine discrete and continuous variables. Although some authors present this as a special feature of the PSO (Sensarma et al., 2002), others point out that there are potential dangers associated with the relaxation process necessary for handling the discrete variables (Abido, 2002). Simple round-off calculations may lead to significant errors.
- Possibility to easily tune the balance between local and global exploration.
- Parallelism. The PSO is inherently well suited for parallel computing. The swarm population can be divided between many processors to reduce computation time.

Below we take a look at some of the applications where the PSO has been used. The first section is devoted to different types of optimization in power systems. These problems are actually similar to those of logistics. After that we briefly overview some other applications found in the literature.

3.3.1 Power system control

An interesting application for which PSO has been successfully implemented is the problem of Optimal Power Flow (OPF) (Abido, 2002). The solution of the OPF aims to minimize some objective function in a power system, *e.g.*, the total energy cost, by adjusting the system control variables. Simultaneously the different power sources and the power grid have to satisfy various equality and inequality constraints. This could for instance be load flow equations (equality constraints) and upper and lower limits on generator voltages (inequality constraints). In his paper, Abido examines the performance of the PSO on a standard OPF test system (the IEEE 30-bus system) and compares his results with previous results obtained with evolutionary programming. The PSO was found to deliver the lowest costs. Other works treating similar problems are (El-Gallad et al., 2001; El-Gallad et al., 2002; Tsukada et al., 2003; Koay and Srinivisan, 2003).

A related problem but with a wider scope is that of Optimal Expansion Planning. Here the optimization is not limited to a power network with a fixed topological structure. Instead, all aspects of constructing a new power network are evaluated to give a complete optimization of the economic consequences. In (Sensarma et al., 2002) a formulation of the Optimal Expansion Planning problem is approached using the PSO. The objective function used is a weighted sum of subobjectives, originating from the large number of economic aspects considered. Although being a very complex problem the PSO is reported to perform well. Given different initial conditions the results seem to be consistent.

3.3.2 Other applications

Neural network methods are popular for solving different kinds of pattern recognition problems, such as classification and machine learning (for an introduction, see (Bishop, 1995)). Lately, the PSO has been used to replace or complement other traditional neural network training algorithms, such as the back-propagation and radial basis function (Kennedy et al., 2001; Salerno, 1997; Al-Kazemi and Mohan, 2002). In (Zhenya et al., 1998) the PSO is used for tuning the parameters of a fuzzy classification network and (Braendler and Hendtlass, 2002) discusses the advantages of using it for hardware neural networks. Peng et al. (1999) successfully train a network with PSO for the use as a battery pack state of charge estimator.

Automatically discovering traffic accidents and other incidents on highways using a combination of PSO and neural networks is studied in (Srinivasan et al., 2003).

Another neural network application which relates to power systems is described in (Kassabalidis et al., 2002). In order to operate close to the system's security border one needs to identify its position as closely as possible. A neural network is trained to answer how close to the security border a certain point is. The PSO is then used in combination with the network to find where the actual border points are situated.

Salman et al. (2001; 2002) suggest the use of PSO for the Task Assignment Problem. The approach taken is straight-forward and good results are reported.

The PSO has also been used to cluster wireless sensors in a sensor network (Tillett et al., 2003). The purpose is to maximize the energy efficiency by minimizing long-range communication. This can be accomplished by grouping sensors in communication clusters and only let one central node communicate with the other clusters.

Finding the period of a function is tackled using particle swarm optimization in (Parsopoulos and Vrahatis, 2003). The authors apply their method to the standard map as well as the Hénon map, and point out that their method works for non-differentiable mappings, in contrast to standard methods.

(Hu et al., 2003c) is one of the few papers on PSO that attempts to solve a permutation problem.

Chapter 4

Conclusions and recommendations

Ants and other social insects solve very complicated logistical problems when they forage for food or build nests. The insects must respond and adapt quickly to changes in the environment in order to survive. In a similar way, the future network based defence must be able to quickly adapt to new conditions. The environment for network-based defense consists of the requirements for resources at different sites. In order to solve the missions facing the armed forces in the future, we must have an agile logistics component. Adapting a swarming-mind-set could play an important part in acquiring this. The main motivation for using swarming based methods for spare parts optimization and supply-chain management is their robustness, flexibility, and success for a wide variety of hard optimization problems.

We see three major uses of swarm-inspired methods for logistics problem. The first two of these solve optimization problem using either ant colony optimization or particle swarm optimization based methods. The third uses swarming as a command and control concept, to optimize the handling of the supply chain.

The natural focus for a follow-up study is the first two uses of swarm-inspired methods, but we believe that using swarming and simple rules for the individual entities will be essential for logistics and also general command and control in a future network based defense.

As seen in section 2.9.1, ant colony optimization has been successfully applied to a number of problems related to logistics. We find it especially interesting to note that variants of the method have been proven to converge to the best solution in a number of cases, see section 2.8. The method has been applied to a number of dynamical optimization problems (Guntsch and Middendorf, 2002; Eyckelhof and Snoek, 2002; Angus and Hendtlass, 2002). The problem solved by (Bullnheimer et al., 1997b) and the paper by Silva (Silva, Runkler and Sousa, 2002) present applications that are very similar to supply chain management in network-based defense. This makes us confident that we can apply swarm intelligence-based methods also for dynamically changing supply chain management.

Also particle swarm optimization has been applied successfully to problems related to military logistics, *e.g.*, (Abido, 2002).

We believe that a follow-up study should explore ant colony optimization for a specific logistical problem as a first priority. Particle swarm optimization should, however, not be completely overlooked. Since there is publically available code (Birge, 2003) for it, it should prove comparatively easy to test it for the same optimization problem. Special care should be taken when deciding what kind of ant colony optimization method to use — as we have seen in chapter 2, some combination of methods must often be used.

The development of new techniques and methodologies inspired by swarming and biology in general proceeds very rapidly. For instance, during the writing of this report, the conference Bio-ADIT 2004[1] was arranged. Browsing the program reveals several papers that might be of peripheral interest for logistics, and at least two that seem to be very interesting: “An Ant Inspired Technique for Storage Area Network Design” and “Media Streaming on P2P Networks with Bio-inspired Cache Replacement Algorithm”, which both solve problems that are relevant for supply chain management in a network-based defense. Several of the other papers at the conference present material that might be relevant for other parts of the future network-based defense. This presents strong reasons for why FOI and FMV should continue to watch this area closely.

[1] http://lslwww.epfl.ch/bio-adit2004/

Bibliography

Abido, M. A. (2002). Optimal power flow using particle swarm optimization, *Electrical power and Energy Systems* **24**: 563–571.

Acan, A. (2002). Gaaco: A ga + aco hybrid for faster and better search capability, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 300.

Agassounon, W. (2003). Distributed information retrieval and dissemination in swarm-based networks of mobile, autonomous agents, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 152.

Al-Kazemi, B. and Mohan, C. K. (2002). Training feedforward neural networks using multi-phase particle swarm optimization, *International Conference on Neural Information Processing*, Vol. 5, pp. 2615–2619.

Alfredsson, P. (1997). *On the Optimization of Support Systems*, PhD thesis, KTH.

Angeline, P. J. (1998a). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, *Evolutionary Programming VII, 7th International Conference, EP98*, pp. 601–610.

Angeline, P. J. (1998b). Using selection to improve particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, pp. 84–89.

Angus, D. and Hendtlass, T. (2002). Ant colony optimisation applied to a dynamically changing problem, *IEA/AIE Conference*, pp. 618–627.

Arkin, R. C. (1998). *Behavior-based Robotics*, MIT Press.

Arquilla and Ronfeldt (2000). Swarming and the future of conflict, *Technical report*, RAND. http://www.rand.org/publications/DB/DB311/DB311.pdf.

Batty, M., Desyllas, J. and Duxbury, E. (2003a). The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades, *Int J Geographical Information Science* **17**(7): 673.

Batty, M., Desyllas, J. and Duxbury, E. (2003b). Safety in numbers? Modelling crowds and designing control for the Notting Hill carnival, *Urban Studies* **40**(8): 1573.

Bauer, A., Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999). An ant colony optimization approach for the single machine total tardiness problem, *Proceedings 1999 congress on evolutionary computation*, p. 1445.

Bautista, J. and Pereira, J. (2002). Ant algorithms for assembly line balancing, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 65.

Bianchi, L., Gambardella, L. M. and Dorigo, M. (2002a). An ant colony optimization approach to the probabilistic traveling salesman problem, *International Conference on Parallel Problem Solving from Nature*.

Bianchi, L., Gambardella, L. M. and Dorigo, M. (2002b). Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic, *Ant Algorithms*, pp. 176–187.

Bianchi, R. A. C. and Costa, A. H. R. (2002). Ant-vibra: A swarm intelligence approach to learn task coordination, *16th Brazilian symposium on artificial intelligence*, Vol. 2507 of *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 195–204.

Bin, W. Y., Shaohui, Z. and Zhongshi, S. (2002). CSIM: A document clustering algorithm based on swarm intelligence, *Congress on evolutionary computation*, pp. 477–482.

Birattari, M., Di Caro, G. and Dorigo, M. (2002). Toward the formal foundation of Ant programming, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 188.

Birge, B. (2003). Psot - a particle swarm optimization toolbox for use with matlab, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 182.

Bishop, C. M. (1995). *Neural Networks*, Oxford University Press.

Blum, C. (2002). Aco applied to group shop scheduling: A case study on intensification and diversification, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 14.

Blum, C. and Sampels, M. (2002). Ant colony optimization for fop shop scheduling: A case study on different pheromone representations, *Proceedings of the congress on evolutionary computation*, p. 1558.

Boeringer, D. W. and Werner, D. H. (2003). A comparison of particle swarm optimization and genetic algorithms fo a phased array synthesis problem, *IEEE Antennas and Propagation Society International Symposium*, pp. 181–184.

Bollobás, B. (1985). *Random Graphs*, Academic Press, New York.

Bonabeau, E. (2002a). Agent-based modeling: Methods and techniques for simulating human systems, *Proc Nat Acad Sci* **99**: 7280.

Bonabeau, E. (2002b). Predicting the unpredictable, *Harvard Business Review* p. 109.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford, UK.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour, *Nature* **406**: 39.

Bonabeau, E. and Théraulaz, G. (2000). Swarm smarts, *Scientific American* pp. 72–79.

Bonabeau and Meyer (2001). A whole new way to think about business, *Harward Business Review* .

Botee, H. M. and Bonabeu, E. (1998). Evolving ant colony optimization, *Advances in Complex Systems* **1**(2): 149–159.

Bowyer, R. S. and Bogner, R. E. (2001). Self-organizing team formation for target observation, *Signal processing, sensor fusion, and target recognition X*, Vol. 4380, SPIE, pp. 339–350.

Braendler, D. and Hendtlass, T. (2002). The suitability of particle swarm optimisation for training neural hardware, *15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2002*, pp. 190–199.

Branke, J., Decker, M. and Merkle, D. (2002). Coevolutionary ant algorithms playing games, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 298.

Branke, J. and Guntsch, M. (2003). New ideas for applying ant colony optimization to the probabilistic TSP, *EvoWorkshops*, pp. 165–175.

Brenner, P. (1986). Tillämpad optimeringslära: Nätverksanalys, heltalsprogrammering, Chalmers tekniska högskola.

Brits, R., Engelbrecht, A. P. E. and van den Bergh, F. (2002). Solving systems of unconstrained equations using particle swarm optimization, *IEEE International Conference on Systems Man and Cybernetics*, Vol. 3.

Brown, K., Bowyer, R. and Koks, D. (2002). Target location by self-organizing autonomous air vehicles, *Battlespace Digitization and Network-Centric Warfare II*, pp. 156–168.

Buhl, J., Deneubourg, J.-L. and Theraulaz, G. (2002). Self-organized networks of galleries in the ant messor sancta, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 163.

Bullnheimer, B., Hartl, R. F. and Strauß, C. (1997a). *A New Rank Based Version of the Ant System*, Working Paper No. 1, SBF, Adaptive Information Systems and Modelling in Economics abd Management Science, Vienna, Austria.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1997b). An improved ant system algorithm for the vehicle routing system, Preprint.

Burdun, I. Y. and Parfentyev, O. M. (1999). AI knowledge model for self-organizing conflict prevention /resolution in close free-flight air space, *IEEE Aerospace Conference*, pp. 409–428.

Canright, G. (2002). Ants and loops, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 235.

Carling, C. and Carlsen, H. (2002). Project metanet: Methods for analysis of complex networks, *7th International Command and Control Research and Technology Symposium*.

Carling, C., Svenson, P., Mårtenson, C. and Carlsen, H. (2003). A flock-based model for ad hoc communication networks, *International Command and Control Research and Technology Symposium*, Washington, DC.

Carlisle, A. and Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments, *International Conference on Artificial Intelligence*, pp. 429–433.

Cincotti, A., Cutello, V. and Pappalardo, F. (2003). An ant-algorithm for the weighted minimum hitting set problem, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 1.

Clerc, M. and Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation* **6**(1): 58–73.

Cordón, O., Fernández de Viana, I. and Herrera, F. (2002). Analysis of the best-worst ant system and its variants on the QAP, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 228.

Costa, D. and Hertz, A. (1997). Ants Can Colour Graphs, *J. Oper. Res. Soc.* **48**: 295–305.

de Campos, L. M., Fernández-Luna, J. M., Gámez, J. and Puerta, J. (2002). Ant colony optimization for learning Bayesian networks, *Int J Approximate Reasoning* **31**: 291.

Denby, B. and Hegarat-Mascle, S. L. (2003). Swarm intelligence in optimisation problems, *NUCLEAR INSTRUMENTS AND METHODS IN PHYSICS RESEARCH SECTION A* **502**: 364.

DoD (2003). *Swarming: Network Enabled C4ISR*, McLean, VA.

Dorigo, M. and Gambardella, L. M. (1997a). Ant colonies for the traveling salesman problem, *BioSystems* **43**: 73–81.

Dorigo, M. and Gambardella, L. M. (1997b). Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computing* **1**(1): 53–66.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B* **26**(1): 29–41.

Dorigo, M. and Stützle, T. (to appear). The ant colony optimization metaheuristic: Algorithms, applications and approaches, *in* F. Glover and G. Kochenberger (eds), *Handbook of Metaheuristics*.

Dro, J. and Siarry, P. (2002). A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 216.

Edwards (2000). Swarming on the battlefield: past, present, and future, *Technical report*, RAND.

El-Gallad, A., El-Hawary, M., Sallam, A. and Kalas, A. (2001). Swarm intelligence for hybrid cost dispatch problem, *Canadian Conference on Electrical and Computer Engineering*, Vol. 2, pp. 753–757.

El-Gallad, A., El-Hawary, M., Sallam, A. and Kalas, A. (2002). Particle swarm optimizer for constrained economic dispatch with prohibited operating zones, *Canadian Conference on Electrical and Computer Engineering*, Vol. 1, pp. 78–81.

Ellabib, I., Basir, O. A. and Calamai, P. (2002). An experimental study of a simple ant colony system for the vehicle routing problem with time windows, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 3.

Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*, MIT Press.

Eyckelhof, C. J. and Snoek, M. (2002). Ant systems for a dynamic TSP: Ants caught in a traffic jam, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 88.

Fan, H. (2002). A modification to particle swarm optimization algorithm, *Engineering Computations* **19**(8): 970–989.

Fenet, S. and Solnon, C. (2003). Searching for maximum cliques with ant colony optimization, *Lecture Notes in Computer Science*, p. 236.

Ferat, S. and Kumar, V. (2002). A swarm intelligence based approach to the mine detection problem, *International conference Systems, man and cybernetics*, p. MP2P3.

Fidanova, S. (2002). ACO algorithm with additional reinforcement, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 292.

Folino, G., Forestiero, A. and Spezzano, G. (2002). Discovering clusters in spatial data using swarm intelligence, *European conference on artificial life*, number 2801 in *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 598–605.

Gagné, C., Price, W. L. and Gravel, M. (2002). Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times, *Journal of the Operational Research Society* **53**: 895.

Gámez, J. A. and Puerta, J. (2002). Searching for the best elimination sequence in Bayesian networks by using ant colony optimization, *Pattern Recognition Letters* **23**: 261.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability. A guide to the Theory of NP-Completeness*, W H Freeman, New York.

Garlick, R. M. and Barr, R. S. (2002). Dynamic wavelength routing in wdm networks via ant colony optimization, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 250.

Gaudiano, Shargel, Bonabeau and Clough (2003). Swarm intelligence: A new C2 paradigm with an application to control of swarms of UAVs, *International Command and Control Research and Technology Symposium*, Washington, DC.

Gimblett, H. R. (2001). *Integrating Geographic Information Systems and Agent-based Modeling Techniques*, Oxford University Press.

Gordon, D. M. (2002). The organization of work in social insect colonies, *Complexity* **7**(6).

Gottlieb, J., Puchta, M. and Solnon, C. (2003). A study of greedy local seach and ant colony optimization approaches for car sequencing problems, *Lecture Notes in Computer Science*, Vol. 2611, p. 246.

Gravel, M., Price, W. L. and Gagne, C. (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic, *European Journal of Operational Research* **143**: 218–229.

Guntsch, M. and Middendorf, M. (2002). Applying population based ACO to dynamic optimization problems, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 111.

Gustafsson, I. (1994). Tillämpad optimeringslära, Chalmers hekniska högskola.

Gutjahr, W. J. (2000). A graph-based ant system and its convergence, *Future Generation Computer Systems* **26**(8): 873–888.

Gutjahr, W. J. (2002). ACO algorithms with guaranteed convergence to the optimal solution, *Information Processing Letters* **82**: 145–153.

Helbing, D., Schweitzer, F., Keltsch, J. and Molnar, P. (1997). Active walker model for the formation of human and animal trail systems, *Physical Review* **E56**: 2527–2539.

Hoar, R., Penner, J. and Jacob, C. (2002). Evolutionary swarm traffic: if ant roads had traffic lights, *Evolutionary Computation*, pp. 1910–1915.

Hoe, K. M., Lai, W. K. and Tai, T. S. (2002). Homogeneous ants for web document similarity modeling and categorization, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 256.

Hu, X. and Eberhart, R. (2002a). Adaptive particle swarm optmization: Detection and response to dynamic systems, *Congress on Evolutionary Computation*, Vol. 2, pp. 1666–1670.

Hu, X. and Eberhart, R. (2002b). Multiobjective optimization using dynamic neighborhood particle swarm optimization, *Congress on Evolutionary Computation*, Vol. 2, pp. 1677–1681.

Hu, X., Eberhart, R. C. and Shi, Y. (2003a). Engineering optimization with particle swarm, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 53.

Hu, X., Eberhart, R. C. and Shi, Y. (2003b). Particle swarm with extended memory for multiobjective optimization, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 193.

Hu, X., Eberhart, R. C. and Shi, Y. (2003c). Swarm intelligence for permutation optimization: a case study of n-queens problem, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 243.

Hughes, D. (2003). Special section on swarm intelligence, Aviation Week and Space Technology.

James, G. K. (2000). *Unmanned aerial vehicles and special operations: Future directions*, Master's thesis, Naval Postgraduate School.

Jha, M. K. (2002). Optimizing highway networks: a genetic algorithm and swarm intelligence based approach, *Computing in civil engineering 2002*, p. 76.

Johansson, R., Xiong, N. and Christensen, H. I. (2003). A game theoretic model for management of mobile sensors, *Proceedings of the 6th International Conference on Information Fusion*, International Society of Information Fusion, pp. 583–590.

Johnson, S. (2001). *Emergence*, Penguin.

Kassabalidis, I., El-Sharkawi, M. A., Marks II, R. J., Arabshahi, P. and Gray, A. A. (2001). Swarm intelligence for routing in communication networks, *IEEE Globecom*.

Kassabalidis, I. N., El-Sharkawi, M. A., Marks, R. J., Moulin, L. S. and da Silva, A. P. A. (2002). Dynamic security border identification using enhanced particle swarm optimization, *IEEE Transactions on Power Sytems* **17**(3).

Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, *Congress on Evolutionary Computing*, pp. 1931–1938.

Kennedy, J. (2000). Stereotyping: Improving particle swarm performance with cluster analysis, *Congress on Evolutionary Computing*, Vol. 2, pp. 1507–1512.

Kennedy, J., Eberhart, R. C. and Shi, Y. (2001). *Swarm Intelligence*, Morgan Kaufmann Publishers, San Fransisco, CA, USA.

Kennedy, J. and Spears, W. M. (1998). Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator, *IEEE Transactions on Evolutionary Computation*, pp. 78–83.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by Simulated Annealing, *Science* **220**: 671–680.

Knuth, D. E. (1998). *Sorting and Searching*, Vol. 3 of *The Art of Computer Programming*, 2 edn, Addison-Wesley, Reading, Ma.

Koay, C. A. and Srinivisan, D. (2003). Particle swarm optimization-based approach for generator maintenance scheduling, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 167.

Koeppel, D. (2004). Massive attack, *Popular Science* (January).

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press.

Krink, T. and Løvberg, M. (2002). The lifecycle model: Combining particle swarm optimisation, genetic algorithms and hillclimbers, *Conference on Parallel Problem Solving from Nature*, pp. 621–630.

Lu, W. Z., Fanb, H. Y. and Loa, S. M. (2003). Application of evolutionary neural network method in predicting pollutant levels in downtown area of hong kong, *Neurocomputing* **51**: 387.

Lua, C. A., Altenburg, K. and Nygard, K. E. (2003). Synchronized multi-point attack by autonomous reactive vehicles with simple local communication, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 95.

Maier, H. R., Simpson, A. R., Zecchin, A. C., Foong, W. K., Phang, K. Y., Seah, H. Y. and Tan, C. L. (2003). Ant colony optimization for design of water distribution systems, *J Water Resources Planning and Management* **129**(3): 200.

Maniezzo, V. and Carbonaro, A. (1999). Ant colony optimization: An overview, *MIC III, Meta Heuristics International Conference*.

Maniezzo, V. and Milandri, M. (2002). An ant-based framework for very strongly constrained problems, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 222.

Marcin L. Pilat, T. W. (2002). Using genetic algorithms to optimize acs-tsp, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 282.

Merkle, D. and Middendorf, M. (2001). A new approach to solve permutation scheduling problems with ant colony optimization, *Lecture Notes in Computer Science*, Vol. 2037, p. 484.

Merkle, D. and Middendorf, M. (2002a). Ant colony optimization with the relative pheromone evaluation method, *EvoWorkshops*, pp. 325–333.

Merkle, D. and Middendorf, M. (2002b). Colony optimization on runtime reconfigurable processor arrays, *Genetic Programming and Evolvable Machines* **3**(4): 345.

Merkle, D. and Middendorf, M. (2002c). Modeling the dynamics of ant colony optimization, *Evolutionary Computation* **10**(3): 235–262.

Merkle, D. and Middendorf, M. (2002d). Modelling aco: Composed permutation problems, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 149.

Merkle, D. and Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine, *Applied Intelligence* **18**: 105.

Meuleau, N. and Dorigo, M. (2002). Ant colony optimization and stochastic gradient descent, *Artificial Life* **8**: 103–121.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, MIT Press.

Mohanty, I., Kalita, J., Das, S., Pahwa, A. and Buehler, E. (2003). Ant algorithms for the optimal restoration of distribution feeders during cold load pickup, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 132.

Montgomery, J. and Randall, M. (2002). Anti-pheromone as a tool for better exploration of search space, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 100.

Mostaghim, S. and Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization (mopso), *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 26.

Newman, M. E. J., Strogatz, S. H. and Watts, D. J. (2001). Random graphs with arbitrary degree distribution and their applications, *Physical Review E* **64**: 026118.

Nouyan, S. (2002). Agent-based approach to dynamic task allocation p. 28, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 28.

Nunes de Castro, L. (2002). Immune, swarm and evolutionary algorithms, *International Conference on Neural Information Processing*, Vol. 3, pp. 1464–1473.

Nygard, K. E., Chandler, P. R. and Pachter, M. (2001). Dynamic network flow optimization models for air vehicle resource allocation, *American Control Conference*, pp. 1853–1858.

Okino, C. M. (2002). On the performance of swarm routing utilizing light-weight agents, *World conference on computational intelligence; Proceedings of the international joint conference on neural networks*, pp. 361–364.

Ouadfel, S., Batouche, M. and Garbay, C. (2002). Ant colony system for image segmentation using markov random field, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 294.

Palmer, D. W., Kirschenbaum, M., Murton, J. P., Kovacina, M. A., Steinberg, D. H., Calabrese, S. N., Zajac, K. M., Hantak, C. M. and Schatz, J. E. (2003). Using a collection of humans as an execution testbed for swarm algorithms, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 58.

Papadimitriou, C. H. (1994). *Computational Complexity*, Addison-Wesley, Reading, MA.

Parpinelli, R. S., Lopes, H. S. and Freitas, A. A. (2002). Mining comprehensible rules from data with an ant colony algorithm, *Lecture Notes in Artificial Intelligence*, Vol. 2507, p. 259.

Parsopoulos, K. E. and Vrahatis, M. (2002). Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing* **1**: 235–306.

Parsopoulos, K. E. and Vrahatis, M. N. (2003). Computing periodic orbits of nondlifferentiable/discontinuous mappings through particle swarm optimization, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 34.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann.

Peng, J., Chen, Y. and Eberhart, R. C. (1999). Battery pack state of charge estimator design using computational intelligence approaches, *Annual Battery Conference on Applications and Advances*, pp. 173–177.

Peterson, C. and Söderberg, B. (1989). A new method for mapping optimization problems onto neural networks, *International Journal of Neural Systems* **1**: 3–22.

Piriyakumar, D. A. L. and Levi, P. (2002). A new approach to exploiting parallelism in ant colony optimization, *IEEE International Symposium on Micromechatronics and Human Science*, pp. 237–243.

Rahoual, M., Hadji, R. and Bachelet, V. (2002). Parallel ant system for the set covering problem, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 262.

RAND (2003). Rand institute research brief: Strategies for an expeditionary army, http://www.rand.org/publications/RB/RB3042/.

Randall, M. and Lewis, A. (2002). A parallel implementation of ant colony optimization, *Journal of Parallel and Distributed Computing* **62**: 1421–1432.

Randall, M. and Montgomery, J. (2002). Candidate set strategies for ant colony optimisation, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 243.

Randall, M. and Tonkes, E. (2001). Solving network synthesis problems using ant colony optimisation, *LNAI*, p. 1.

Reimann, M., Doerner, K. and Hartl, R. F. (2002). Insertion based ants for vehicle routing problems with backhauls and time windows, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 135.

Roux, O., Fonlupt, C., Talbi, E.-G. and Robilliard, D. (1999). ANTabu, *Technical Report LIL–99–01*, Laboratoire d'Informatique du Littoral, Université du Littoral, Calais, France.

Sahin, E. and Franks, N. R. (2002). Simulation of nest assessment behavior by ant scouts, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 274.

Saleh, H. A. (2002). Gps positioning networks design: An application of the ant colony system, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 302.

Salerno, J. (1997). Using the particle swarm optimization technique to train a recurrent neural model, *IEEE International Conference on Tools with Articial Intelligence*, pp. 45–49.

Salman, A., Ahmad, I. and Al-Madani, S. (2001). Particle swarm optimization for task assignment problem, *International Conference on Artificial intelligence and Applications*, pp. 231–236.

Salman, A., Ahmad, I. and Al-Madani, S. (2002). Particle swarm optimization for task assignment problem, *Microprocessors and Microsystems* pp. 363–371.

Schoofs, L. and Naudts, B. (2002). Swarm intelligence on the binary constraint satisfaction problem, *Proceedings of the congress on evolutionary computation*, p. 1444.

Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L. (1996). Ants for load balancing in telecommunication networks, *Technical Report HPL–96–35*, Hewlett-Packard Laboratories, Bristol, UK.

Schrage and Gonsalves (2003). Sensor scheduling using ant colony optimization, *6th International Conference on Information Fusion*, Cairns, Australia.

Semet, V., Lutton, E. and Collet, P. (2003). Ant colony optimisation for e-learning: observing the emergence of pedagogic suggestions., *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 46.

Sensarma, P. S., Rahmani, M. and Carvalho, A. (2002). A comprehensive method for optimal expansion planning using particle swarm optimization, *2002 IEEE Power Engineering Society Winter Meeting*, Vol. 2, pp. 1317–1322.

Shi, Y. and Eberhart, R. (1998a). A modified particle swarm optimizer, *IEEE International Conference on Evolutionary Computation*, pp. 69–73.

Shi, Y. and Eberhart, R. (1998b). Parameter selection in particle swarm optimization, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 591–601.

Shmygelska, A., Aguirre-Hernndez, R. and Hoos, H. H. (2002). An ant colony optimization algorithm for the 2d hp protein folding problem p. 40, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 40.

Shuy, S. J., Yin, P. Y., Lin, B. M. T. and Haouari, M. (2003). Ant-tree: An ant colony optimization approach to the generalized minimum spanning tree problem, *Journal of Experimental and Theoretical Artificial Intelligence* **15**: 103–112.

Silva, A., Neves, A. and Costa, E. (2002). An empirical comparison of particle swarm an predator prey optimisation, *Irish Conference on Artificial Intelligence and Cognitive Science*, pp. 103–110.

Silva, C. A., Runkler, T. A. and Sousa, J. (2002). Ant colonies as logistic processes optimizers, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 76.

Sim, K. M. and Sun, W. H. (2002). Multiple ant-colony optimization for network routing, *Cyber Worlds, 2002*, pp. 277– 281.

Socha, K., Knowles, J. and Sampels, M. (2002). A $\mathcal{MAX} - \mathcal{MIN}$ ant system for the university course timetabling problem, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 1.

Srinivasan, D., Loo, W. H. and Cheu, R. L. (2003). Traffic incident detection using particle swarm optimization, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 144.

Stützle, T. and Dorigo, M. (2002). A short convergence proof for a class of ant colony optimization algorithhms, *IEEE Transactions on Evolutionary Computation* **6**(4): 358–365.

Stützle, T. and Hoos, H. H. (2000). $\mathcal{MAX} - \mathcal{MIN}$ ant system, *Journal of Future Generation Computer Systems* **16**: 889–914.

Suiliong, L., Zincir-Heywood, A. N. and Heywood, M. I. (2002). The effect of routing under local information using a social insect metaphor, *Evolutionary Computation*, pp. 1438–1443.

Sumpter, D. J. T. and Beekman, M. (2003). From nonlinearity to optimality: pheromone trail foraging by ants, *Animal Behaviour* **65**.

Sun, Z.-G. and Teng, H.-F. (2002). An ant colony optimization based layout optimization algorithm, *IEEE Region 10 Conference on Computers, Communications, Control and Power Engines*, Vol. 1, pp. 675–678.

Svennebring, J. and Koenig, S. (2002). Towards building terrain-covering ant robots, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 202.

Svenson, P. and Sidenbladh, H. (2003). Determining possible avenues of approach using ANTS, *6th International Conference on Information Fusion*, Cairns, Australia.

Tarasewich, P. and McMullen, P. R. (2002). Swarm intelligence: Power in numbers, *C. ACM* **45**(8): 62.

Tillett, J., Rao, R., Sahin, F. and Rao, T. (2003). Particle swarm optimization for the clustering of wireless sensors, *SPIE*, Vol. 5100, pp. 73–83.

T'kindt, V., Monmarche, N., Tercinet, F. and Laugt, D. (2002). An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem, *European Journal of Operational Research* .

Trahan, M. W., Wagner, J. S., Stantz, K. M., Gray, P. C. and Robinett, R. (1998). Swarms of UAVs and fighter aircraft, *International Conference on Nonlinear Problems in Aviation and Aerospace*, pp. 745–752.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters* **85**: 317–325.

Tsukada, T., Tamura, T., Kilagawa, S. and Fukkiyama, Y. (2003). Optimal operational planning for cogeneration system using particle swarm optimization, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 138.

Vallone, U. (2002). Bidimensional shapes polygonalization by aco, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 296.

van den Bergh, F. and Engelbrecht, A. P. E. (2002). A new locally convergent particle swarm optimiser, *IEEE International Conference on Systems Man and Cybernetics*, Vol. 3.

Venter, G. and Sobieszczanski-Sobieski, J. (2003). Particle swarm optimization, *AIAA Journal* **41**(8): 1583–1589.

Vogel, A., Fischer, M. and Jaehn, H. (2002). Real-world shop floor scheduling by ant colony optimization, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 268.

Wang, X. and Wu, T. (2002). Ant colony optimization for intelligent scheduling, *4th World Congress on Intelligent Control and Automation*, p. 66.

Watts, D. J. (1999). *Small Worlds*, Princeton University Press.

Wittner, O. and Helvik, B. E. (2002). Cross-entropy guided ant-like agents finding cyclic paths in scarcely meshed networks, *Ant Algorithms: 3rd International Workshop*, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 123.

Wolpert, D. H. and Lawson, J. W. (2002). Designing agent collectives for systems with markovian dynamics, *Autonomous Agents and Multi Agent Systems*, AAI, p. 1066.

Xie, X. F., Zhang, W. J. and Yang, Z. L. (2002a). Adaptive particle swarm optimization on individual level, *International Conference on Signal Processing*, pp. 1170–1173.

Xie, X. F., Zhang, W. J. and Yang, Z. L. (2002b). Hybrid particle swarm optimizer with mass extinction, *Conference on Communiction, Circuits and Systems (ICCCAS)*, pp. 1170–1173.

Yang, Y. and Kamel, M. (2003). Clustering ensemble using swarm intelligence, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 65.

Yuigying, D., Yan, H. and Jingping, J. (2003). Multi-robot cooperation method based on the ant algorithm, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, IEEE, p. 14.

Zhenya, H., Chengjian, W., Luxi, Y., Susu, Y., Eberhart, R. C. and Shi, Y. (1998). Extracting rules from fuzzy neural networks by particle swarm optimisation, *IEEE International Conference on Evolutionary Computation*, pp. 74–77.

Appendix A

NP-Completeness

Computer[1] scientists classify problems according to the maximal amount of resources needed for their solution. The most important resource is time, but it is also possible to distinguish between problems that require qualitatively different amounts of memory. For example, a list of N elements can always be sorted in time less than $kN \log N$, where k is some constant (Knuth, 1998). The problems whose running time on a *universal Turing machine* is bounded by a polynomial in their size are said to be in the class P. The important class NP (for non-deterministic polynomial) consists of those problems where it can be checked in polynomial time whether a proposed solution actually solves the problem. (A *non-deterministic* Turing machine would be able to solve NP problems in polynomial time.) It is obvious that $P \subseteq NP$, but there is no proof that $P \neq NP$. However, most people believe that there are NP problems whose worst-case instances take exponential time to solve on a universal Turing machine.

The class NP-complete (or NPC) are the most important problems in NPs. A problem of size N is in NPC if all other NP problems can be transformed into it in time at most polynomial in N. A method to solve an NPC problem efficiently can thus be used to solve any NP problem efficiently. It is known that if $P \neq NP$ then there are problems in NP that are in neither P nor NPC. A problem is called NP-hard if it is at least as difficult as the most difficult NP problems; NPC is the intersection of NP and NP-hard. A modern reference on complexity theory and NP problems is (Papadimitriou, 1994), while (Garey and Johnson, 1979) has an extensive list of NPC problems.

Two important problems in NPC are graph coloring and satisfiability testing (**SAT**). Graph coloring is the problem of coloring a graph with N vertices and M edges using K colors so that no two adjacent vertices have the same color.

The most natural application of graph coloring is in scheduling problems. For example, a school where each teacher and student can be involved in several different classes must schedule the classes so that no collisions occur. If there are K different time slots available, this is K-COL

Satisfiability was the first problem shown to be in NPC. It is the problem of finding an assignment of true or false to N variables so that a boolean formula in them is satisfied. In K-SAT, this formula is written in *conjunctive normal form* (CNF), that is, it consists of the logical **AND** of M clauses, each clause being the **OR** of K (possibly negated) variables, where the same clause may appear more than once in a formula. For example, $(x \vee y) \wedge (y \vee \neg z)$ is an instance of **2-SAT** with two clauses and three variables. Applications of K-SAT include theorem proving, VLSI design, and learning.

In K-SAT, each clause forbids one of the 2^K possible assignments for its variables. In the same way, an edge in a graph forbids K of the K^2 different colorings of its vertices. For both problems, there are M *constraints* on the solutions. More general problems can be formulated by having N variables and a set of M constraints that are of different nature than those in K-SAT or K-COL. Note, though, that the NP-completeness of both problems means that a method to solve either one quickly would work on any NP-problem. Both K-SAT and K-COL are in P for $K = 2$ and in NPC for $K \geq 3$ (Garey and Johnson, 1979). The related problem (**MAX-K-SAT**) of trying to minimize the number of unsatisfied clauses in k-SAT is in NPC even for $K = 2$.

[1] This appendix is based on section II of Svenson, Nordahl, "Relaxation in graph coloring and satisfiability problems", *Phys Rev* **E59** 3983.

Appendix B

Web resources

Here we provide a list of web resources that are related to swarming, ant colony optimization and particle swarm optimization.

- Several swarm-related conferences will be arranged in the near future:
  - Bio-ADIT 2004 http://lslwww.epfl.ch/bio-adit2004/
  - GECCO 04 http://gal4.ge.uiuc.edu:8080/GECCO-2004/index.html
  - ANTS 2004 http://iridia.ulb.ac.be/∼ants/ants2004/index.html
  - PPSN VIII http://events.cs.bham.ac.uk/ppsn04/
  - CEC 04 http://www.cs.unr.edu/∼sushil/cec/#SI
  - SIP 04 http://alfa.ist.utl.pt/∼cvrm/staff/vramos/SIP.html
  - ICCRTS 2004 http://www.dodccrp.org/Activities/Symposia/9thICCRTS/9thICCRTSCallforPapers.pdf

  FOI should track the proceedings from all these conferences/special sessions to find relevant material.
- There are a number of companies specializing in applying swarm-based methods to problems:
  - Icosystems http://www.icosystems.com/
  - NuTechSolutions http://www.nutechsolutions.com/
  - AntOptima (http://www.antoptima.com) has a tool for logistical optimization http://www.antoptima.com/antplan.html
- A conference on swarming for military purposes, arranged by the US Department of Defense http://www.dodccrp.org/Publications/pdf/Swarming_Conf_Pro.pdf
- Wired recently ran a short article on swarming http://www.wired.com/wired/archive/12.02/machines.html?pg=6
- A site that collects a lof ot introductory articles on swarming http://dsp.jpl.nasa.gov/members/payman/swarm/
- Central site for PSO http://www.particleswarm.net/
- Website of the inventor of particle swarm optimization http://www.particleswarm.net/JK/
- Web site for the book (Kennedy et al., 2001) http://www.engr.iupui.edu/∼eberhart/web/PSObook.html
- Site with PSO examples http://clerc.maurice.free.fr/pso/index.htm
- Website for ant colony optimization is maintained by Dorigo and contains links to many papers and conferences on ACO: http://iridia.ulb.ac.be/∼mdorigo/ACO/ACO.html
- Article in the Santa Fe Institute Bulletin on swarming and its use in experimental archaeology http://www.santafe.edu/sfi/publications/Bulletins/bulletin-winter98/swarm.html
- Website of a robotics project http://www.swarm-bots.org/
- Swarm is a software package for agent-based simulation http://www.swarm.org/

Swarm Intelligence for logistics: Background

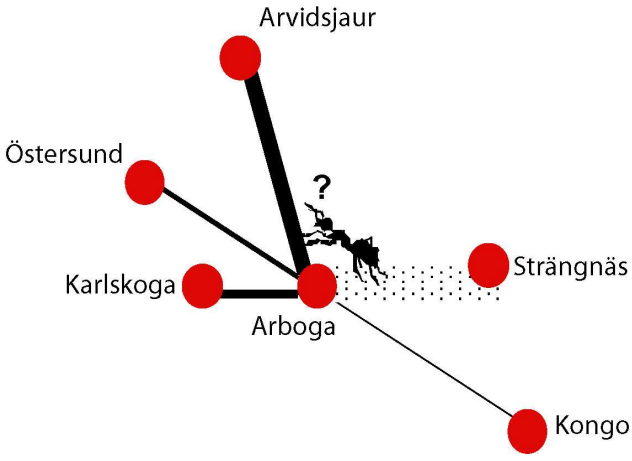
Rapportöversikt FOI-R—1180--SE

Författare:Pontus Svenson, Christian Mårtenson, Hedvig Sidenbladh, Michael Malm
Projekt: Swarm Intelligence för logistik (forskningsområde Spaning och ledning)

Swarm intelligence

Vissa typer av myror kommunicerar information om matställen genom att lägga ut doftspår mellan matfynd och myrstack. Doften avdunstar med tiden vilket gör att korta stigar i medeltal kommer ha starkare doftspår än långa. Andra myror attraheras av doften och ju starkare doft desto större är sannolikheten att ett spår följs. Så småningom kommer myrstigar som utgör kortaste vägar mellan föda och stack att bildas och myrkolonin har löst ett optimeringsproblem utan att vara medvetna om det.

Detta är inte det enda exemplet på hur insekter och även fiskar och fåglar organiserar sig med hjälp av mycket enkla individuella regler. På så sätt bildar fiskar stim, fåglar flockar och termiter och getingar bygger bon med komplexa strukturer. Fenomenet kallas *swarm intelligence* och har gett inspiration till en rad olika datalogiska optimeringsmetoder. Principen är att många enkla agenters enskilda (lokala) ansträngningar i samverkan kan ge en i det närmaste explosiv (global) synergieffekt, så kallat *emergent uppträdande*. Metoderna är väldigt robusta - det gör inget om en eller ett par myror misslyckas - och har dessutom ofta förmågan att anpassa sin lösning om problemvillkoren plötsligt skulle förändras - en matkälla som tar slut eller flyttas leder till att doftspåret dit snart dunstar bort och nya myrstigar bildas.



Dagens datorer möjliggör användandet av swarm intelligence-metoder för att lösa komplexa optimeringsproblem. Ett typiskt sådant är hanteringen av försvarets logistik. Det nya försvaret kräver förmåga att snabbt kunna göra insatser på olika håll, antingen det rör sig om internationella operationer, katastrofhjälp eller traditionellt försvar av Sverige. Det är med andra ord viktigt att snabbt kunna planera logistikflöden som tar hänsyn till behov i ett stort antal insatsområden. Behoven kan dessutom ändras snabbt, vilket kräver omplanering. Swarm intelligence-baserade metoder klarar sådana krav.

Rapportinnehåll

Den här rapporten är resultatet av en litteraturstudie om swarm intelligence. Studien har främst inriktats på logistiknära tillämpningar med användning inom det nätverksbaserade försvaret. Vi beskriver överskådligt bakgrunden till swarming och hur det använts inom biologi och sociologi. Två huvudvarianter av swarm-metoder för optimering beskrivs i detalj. Den första, ant colony optimization, baseras på hur myror letar efter mat, medan den andra, particle swarm optimization, baseras på hur individer i folkmassor låter sitt beteende påverkas av sin omgivning. Vi beskriver originalversionerna samt några intressanta tillämpningar och utvidgningar för respektive metod.

Vi nämner också tidigare framförda idéer om militära tillämpningar av swarming (t ex sensorstyrning och hotanalys), och ger en kort introduktion till vissa logistikproblem som är relevanta för det framtida nätverksbaserade försvaret.

Studiens huvudresultat är bibliografin. Här har vi samlat en stor mängd artiklar som tillämpar swarm intelligence på olika problem, främst logistikrelaterade sådana.

Kontaktperson: Pontus Svenson
pontus.svenson@foi.se
http://www.foi.se/fusion

www.foi.se

Avdelningen för Ledningssystem
Institutionen för Data- och Informationsfusion
172 90 Stockholm
Telefon 08-55 50 30 00