EMERGENT BEHAVIOR BASED IMPLEMENTS FOR DISTRIBUTED NETWORK MANAGEMENT

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOKTOR INGENIØR

OTTO WITTNER

Department of Telematics
Norwegian University of Science and Technology
2003

Abstract

Network and system management has always been of concern for telecommunication and computer system operators. The need for standardization was recognised already 20 years ago, hence several standards for network management exist today. However, the ever-increasing number of units connected to networks and the ever-increasing number of services being provided results in significant increased complexity of average network environments. This challenges current management systems. In addition to the general increase in complexity the trend among network owners and operators of merging several single service networks into larger, heterogenous and complex full service networks challenges current management systems even further. The full service networks will require management systems more powerful than what is possible to realize basing systems purely on todays management standards. This thesis presents a distributed stochastic optimization algorithm which enables implementations of highly robust and efficient management tools. These tools may be integrated into management systems and potentially make the systems more powerful and better prepared for management of full service networks.

Emergent behavior is common in nature and easily observable in colonies of social insects and animals. Even an old oak tree can be viewed as an emergent system with its collection of interacting cells. Characteristic for any emergent system is how the overall behavior of the system emerge from many relatively simple, restricted behaviors interacting, e.g. a thousand ants building a trail, a flock of birds flying south or millions of cells making a tree grow. No centralized control exist, i.e. no single unit is in charge making global decisions. Despite distributed control, high work redundancy and stochastic behavior components, emergent systems tend to be very efficient problem solvers. In fact emergent systems tend to be both efficient, adaptive and robust which are three properties indeed desirable for a network management system. The algorithm presented in this thesis relates to a class of emergent behavior based systems known as *swarm intelligence* systems, i.e. the algorithm is potentially efficient, adaptive and robust.

On the contrary to other related swarm intelligence algorithms, the algorithm presented has a thorough formal foundation. This enables a better understanding of the algorithm's potentials and limitations, and hence enables better adaptation of the algorithm to new problem areas without loss of efficiency, adaptability or robustness. The formal foundations are based on work by Reuven Rubinstein on cross entropy driven optimization. The transition from Ruinstein's centralized and synchronous algorithm to a distributed and asynchronous algorithm is described, and the distributed algorithm's ability to solve complex problems (NP-complete) efficiently is demonstrated.

Four examples of how the distributed algorithm may be applied in a network management context are presented. A system for finding near optimal patterns of primary/backup paths together with a system for finding cyclic protection paths in mesh networks demonstrate the algorithm's ability to act as a tool helping management system to ensure quality of service. The algorithm's potential as a management policy implementation mechanism is also demonstrated. The algorithm's adaptability is shown to enable resolution of policy conflicts in a soft manner causing as little loss as possible. Finally, the algorithm's ability to find near optimal paths (i.e. sequences) of resources in networks of large scale is demonstrated.

Keywords Network management, emergent behavior, emergent systems, swarm intelligence, mobile agents, multi-criteria optimization, cross entropy, rare events, routing, protection switching, policy management, policy conflicts, network resource location, large scale swarm intelligence.

Preface

During the last decade users of electronic equipment has learned to appreciated the advantages gained by interconnecting electronic equipment and enabling transport of digitized information between them. The introduction of hyper-link documents with universal resource links (HTML) enabled the emergence of the World-Wide-Web and made computer-to-computer communication interesting for the average computer user, which again triggered a demand for connecting personal computers to the internet. Agreements on second generation mobile phone standards like GSM enabled equipment manufactures to produce cheap, powerful and user friendly mobile handsets which again has lead to mobile communication being an everyday activity for many people.

As more and more valuable information (as well as huge amounts of less valuable information) is published using the new electronic channels, the desire for users to always be "online" is increasing steadily. Even for the more conservative people, who wish to hold on to the traditional information channels, the overall acceptance of email and SMS messaging by friends, relatives and colleagues force them to obtain an email alias and buy a mobile phone, or else vital information exchanges may not take place.

De-regulation in the telecommunication market place in many countries has increased the total number of telecommunication service providers and network operators. Similar services are provided by many provides, but service limitations may exist depending on the networks the communicating parties access and hence which operators are involved.

So far users themselves have been forced to keep track of similarities and differences in functionality, price and quality of the services offered in the "jungle of telecommunications". But times are changing. Many operators have realised that restructuring, integration and optimization of their networks could enable them to present a far more user friendly and attractive overall service offer. In addition higher overall utilization of network resources could be achieved and hence a better revenue experienced.

Automated network management has always been of interest to operators, first of all to enable cost efficient monitoring and execution of basic management operations, e.g. adding new subscribers, adding new links, upgrading switch software etc. Restructuring, integration and optimization of todays networks will increase network complexity significantly, thus automation will become essential for operators to keep minimum control over their networks. Traditional network management technologies and standards developed during the last decade can still handle minimum management of most installations but will soon have to be supplemented (or even replace) by more flexible and powerful management technologies.

The Network Management Project

Late professor Tore Riksaasen at the Department of Telematics of the Norwegian University of Science and Technology was well aware of the trends described above already 10 years ago. He manage after several years of interaction with industry partners in Norway and the Norwegian Research Council to initiate a research project known as the “Network Management Project”. Very unfortunately Tore Riksaasen was hospitalized due to a lethal illness just after his project started. As one out of three PhD student given grants from the Network Management project and hence given the opportunely to learn the art of research, I already here given my sincere thanks to Tore Riksaasen for his project initiative and for the help and clues he managed to give me during my first years as a PhD student despite his weak physical condition.

Mobile Agents

Among several other topics, Tore Riksaasen listed *agent technology* as one potentially important concept for future network management. The term *agent* has many meanings and interpretations, but the instant mood triggered in me and all other fans of Ian Fleming and his stories, has an ooze of novelty, stealth, justice and action. And we all known that if double-O-seven him self would have been given the challenges of future network management as his next assignment, there is hardly any doubt that we would all soon be communicating in ways we never even dared imagine could be possible.

However since 007 still after 40 years is very busy ensuring justice and entertaining those beautiful women, synthetic agents seem to be the only option. But what kind of synthetic agents? I soon discovered that synthetic agents could be anything from physical robots to simple information objects in a software system. Probably inspired by agent 007’s impressive athletic abilities I ended up looking at something in between, the concept known as *mobile software agents*.

So, what is really a mobile software agent? Unfortunately no formal definition exist, but a few capabilities are essential

Autonomity, i.e. a mobile software agent is an encapsulated piece of software containing sufficient code describing its behavior and sufficient data-structures describing its state to be (relatively) independent of other software components.

Mobility, i.e. an mobile software agent is able to somehow wrap it self up, move it self to a new location, unwrap and continue execution.

Communication, i.e. a mobile software agent is able to communicate with other software agents and/or other units in the environment.

For anyone who have experienced the inconvenience of having your personal computer infected by a computer virus, the above list might sound familiar. A computer virus is a good example of a mobile software agent, typical with a malicious behavior. A well designed virus operates in a distributed fashion by moving copies of itself to as many hosts as possible. It adapts to its host environment and is difficult to get rid of, properties to be recognised from agent 007's evil minded opponents and indeed from 007 him self. So, why not adopt the strategies of Mr Bond, and approach the problematic areas of future network management with a potentially distributed, adaptive and robust technology?

Emergent Behavior

Researchers working with mobile agent technology can be divided in two main groups; *architecture* and *application* developers. During the last decade a lot of effort has been put into developing system architectures, platforms, APIs (application programming interfaces), development languages and other tools required and/or believed convenient when developing applications based on mobile software agents, i.e. the architecture developers have been busy. Unfortunately the other group, the application developers, seem to struggle. So far only a limited number of mobile agent based applications have been developed, and only a few of those again can be said to depend upon the mobile software agent concept. That is, so far most mobile agent based applications can be implemented with success using other more traditional development concepts. However there seem to be one area of computer science which may benefit from the use of mobile agents: System development based on *emergent behavior*.

When the resulting behavior from a composition of several basic behaviors is more powerful than the sum of the basic behaviors, the resulting behavior is called *emergent*. For instance, a single ant would not be able to build a proper nest. One ant can find appropriate material for nest building but will not be able to collect and organize sufficient material to build anything resembling an ant hill. However, a thousand ants can indeed build and operate a nest, even when environmental conditions are unpredictable, i.e. a predator suddenly eats 100 ants, weather conditions destroy parts of the nest structures etc. The basic behavior of an ant is the same in both cases. Hence the behavior required to manage proper nets building *emerge* from many simple nest-building-incapable behaviors.

As in ant colonies, emergent behavior is observable in colonies of termites, bees and many other social insects and animals. In other words, emergent behavior is a powerful concept implemented frequently in natural systems. And it is agreeable that social insects and animals indeed have similar capabilities to mobile software agents (and of course many other capabilities). Thus application development based on emergent behavior using mobile software agents seems to have at least some potential.

The awareness among researchers of the potential power of applying emergent behavior in computer and telecommunication systems has increased significantly over the last few years, and today research groups and projects (e.g. IRIDIA in Brussels, BISON in Bologna) strive to understand what building blocks (basic agent behaviors and agent interaction patterns) are required to realize specific emergent behaviors. This thesis presents results from such emergent behavior research. Part of the work considers development of formal foundations, but a strong application focus is always kept with network management applications in mind. In more informal terms, this thesis contributes to the art of developing squads of simple synthetic software agents in the hope that they to some degree, assisted by emergent behavior, can act as a substitute for highly intelligent agent 007, or Bond ... James Bond.

Acknowledgments

Many people have contributed directly or indirectly to the work presented in this thesis. I hereby give my sincere thanks to all of you. However without the help and contributions from my supervisor and coauthor professor Bjarne E. Helvik, this thesis would never have been written. Through all the years I have been pursuing my doctoral degree he as alway been there giving me support, encouragement and inspiration, and many times by just a simple sentence made my thoughts clear and whatever solution I was looking for seem very obvious. I do indeed suspect that Bjarne has magical powers. I also give my special thanks to late professor Tore Riksaasen. Without his initiative and ideas the *Network Management Project* with the scholarships it offered would not have be realized, and I would not have had the opportunity to pursue my doctoral degree. The rest of my colleagues from the department of telematics have also given numerous fruitful contributions to my work through discussions and informal chats over all those cups of coffee. Sincere thanks to all of them, and especially Randi Flønes for her patience and tidiness in administrative matters, Poul for his important contributions as a coauthor, Asbjørn, Pål and Jarle for their technical support, Per-Oddvar, Hein and Fritjof for their patience as my office-mates, Jacqueline for her enthusiasm and happy laughters, and Frank Li for all his help and hints contributing to the format of this thesis. Finally, I thank my family for their encouragement and patience, and last but not least my girl friend Kristin for all the support she has given me.

This thesis has been written in LyX under Linux using a customized version of the LaTeX document class *kapproc.cls* provided by Kluwer Academic Publishers. *kapproc.cls* is originally intended for generating conference proceedings.

Contents

LIST OF PAPERS

The following papers constitute part II of this thesis.

In addition to the papers in part II, the following extended abstract is included as appendix "Paper F"

[1]A slightly extended version of this paper will be published in the *Annals of Telecommunications Journal, Special Issue on "Policy-based Control"* in January/February 2004.

I

INTRODUCTION

Introduction

The main part, Part II, of the thesis is a collection of papers published at different scientific conferences and workshops. The first part, this introduction, is intended to give the reader an overview of the material presented in the papers in Part II.

Section 1 of the introduction presents background information giving the context for topics presented in the papers. Three main areas are discussed; the growth in telecommunication network complexity and hence growth in network management complexity, emergent systems and their properties, and how emergent systems may contribute to network management of complex networks. Related work to the research presented in the papers is also introduced.

Section 2 presents the overall research focus of the thesis. Essential differences between work in the papers and other related work in the area of emergent systems are explained.

Section 4 outlines how the papers relates to each other. A map indicating overlapping sections in the papers is presented. The map is intended as an aid to minimize redundant reading.

Section 5 presents an abstract of each paper pointing out the main contributions. Section 6 discusses implementation issues. Section 7 summarizes and concludes, and finally Section 8 presents future work.

1. Background

Network and system management has always been of concern for telecommunication and computer system operators. The need for standardization was recognised already 20 years ago, and the foundations for todays two leading network management standards, the *Simple Network Management Protocol* (SNMP) [CFSD90] and *Telecommunication Management Network* (TMN) [ITU00] were developed in the early 90s. SNMP was intended to be simple and easy to apply which has been the reason for its success and wide acceptance, but also limits its use. SNMP is most commonly applied as a tool for monitoring. TMN was designed to manage complex telecommunication networks, potentially all types of future networks. Even though TMN's expressive power is strong due to its OSI Management foundation, installation and operation is not at all simple. TMN relies on a complex software stack, and may require significant processing resources.

Both SNMP and TMN are emerging standards. SNMP version 3 [CMPS90] is now available and provides solutions to several of the weaknesses of earlier versions. General increase in processing power and storage capabilities in new network equipment has made the process of TMN-enabling equipment less complicated. New alternative management standards are also appearing. Two of the most promising ones are the *Java Management Extension* (JMX) [McM00] and *Web-Based Enterprise Management* (WBEM) [DMTF03] (see [HAN99][MFZ00] for surveys and appendix D for details on distributed properties).

Several management protocols accompany SNMP, TMN, JMX and WBEM. For system management in general, policy driven management [Slo94b] is an option. Several policy related protocols and languages exist (e.g. COPS, Ponder [DBC$^+$00, DDLS01]). For routing and resource management in IP networks a range of standardized protocols exist, e.g. RIP, OSPF, BGP, Diffserv, IntServ, RSVP, ICMP [PD00]. For telecommunication networks, routing and resource management are typically handled by SS7 [ITU93]. Note that routing management and to some degree resource management nowadays often are considered to be parts of a network's control layer, i.e. separate from the networks management system. In this thesis however, separating control and management is regarded not to improve readability (rather lessen it), and hence both control and management functionality are viewed as parts of a network's overall management system.

Even though the management standards with their accompanied protocols together can provide powerful management systems, the general growth of computer system and telecommunication network complexity as well as the trends among operators of merging single service networks into full service networks present severe challenges. The next subsection takes a closer look at this complexity growth. Further enhancement of the management standards and their accompanied protocols will be necessary for them to efficiently manage the upcoming complex full service networks. The two last subsections present *emergent behavior* as a potential mechanism in this context.

1.1 Network Management and Network Complexity

Since Moore's law for the increase in computing power [Moo65] is still valid and can be expected to be valid for several years to come, computer controlled units in general will soon possess enough computing resources to provide everything from multimedia services to core router functionality. Most of these units will be connected to a network. Assuming that the majority of these units will provide several services as well as access several services provided by other similar units, the overall complexity of a future computer and telecommunication network will become even more devastating than networks existent today. Awareness of this fact has inspired computer and telecommunication companies and research institutes to look for new network and system management approaches able to handle very high system complexity. IBM's *Autonomic Computing* initiative [KZS$^+$01, Res03] and the *BISON* project [BCG$^+$03] are examples.

Adding to the above described trend of complexity growth and general need for new management approaches is another trend: *Mergence of networks*. Network oper-

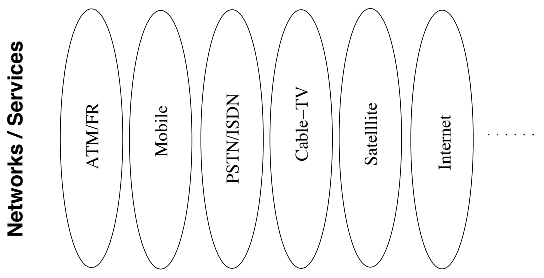


Figure 1. Enterprise viewpoint of a set of *single service networks*.

ators are restructuring and merging their computer and telecommunication networks into large integrated systems, and by this expecting to increase utilization of resources and improve their ability to adapt their networks to upcoming customer needs. Such network mergences are likely to result in an overall leap up in system complexity, and potentially a sudden need for new management solutions. The rest of this section looks into this specific trend in more detail.

Nowadays most telecommunication network operators own, build and managed what is know as *single service networks*. Figure 1 depicts an enterprise viewpoint of a set of single service networks.

An access network technology together with an appropriate transport network is managed and operated as an independent network. Only a single basic services is offered by the network. Variations of the service may be provided, however the basic underlying service is the same, e.g. PSTN voice communication service with added faximile functionality, or mobile voice service with an enhanced signalling channel able to transfere short text messages. If a user requires a wider range of services than a single service network can provide, the user must register independent subscriptions with each relevant network operator.

Each single service network has its own network management system. Even though a very limited number of services are provided, the number of subscribers may be large and the overall structure of the network may be complex. Hence, the management system must cope with many complex problems within fault, configuration, accounting, performance and security management, i.e. the FCAPS functional areas defined in TMN. When the scale of a network increases the overall management complexifies significantly.

However, modelling the network and predicting overall user behavior with reasonable accuracy is possible since only one basic service is provided and new service variations are rarely introduced. A relatively static environment and accurate behavior models open for efficient collection of statistics, off-line system analysis and the use of powerful centralized optimization tools to assist operators in making network management decisions. Customer management is also relatively simple, again, due to the limited number of services provided.

A major disadvantage of single service networks is their lack of ability to support new innovative services. Introducing a new service very different from the existing

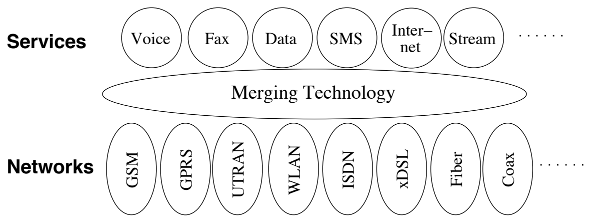


Figure 2. Enterprise viewpoint of a *full service network*.

basic service without degrading overall QoS, typically requires establishing a completely new single service network including a user access technology and transport network resources (i.e. a new eliptic section in Figure 1). Hence new services can be expensive and time consuming to realize.

To simply avoid introducing new innovative services is not a good option for an operator. If a service is desired by a group of user, the users tend to find a way to realize the service by combining and/or exploiting existing services, (e.g. realize internet access by connecting a modem to a fixed cost PSTN voice service, dial up an ISP and keep the connection up permanently). Such exploitation often results in new usage patterns of network resources which, again, may invalidate user behavior models founding the management strategies applied by the operator. Network management may become difficult, inefficient and expensive.

To enable fast and less expensive introduction of new services, many operators have initiated the process of merging their single service networks into what is known as a *full service network*. Figure 2 presents an enterprise viewpoint of a full service network where services no longer depend on one specific network technology. A horizontal merging layer is introduced between the network technologies and the service logic. The layer presents a generic communication interface to all services, thus enables fast deployment and removal of services. Introducing new network technologies below the layer should also be simpler and potentially more cost efficient since a large number of services may start utilizing the technology immediately after it becomes operative. Finally, costumer management can be improved. A user should no longer need to register a subscription for each new service accessed. A single subscription can enable access to a range of services.

However, merging many independent single service networks into one full service network is not riskless. Choosing a suitable merging technology is only the first challenge. Today there is a common belief among operators that IP (i.e. the internet protocol and its related protocols) is the best choice. Figure 3 shows the classical hourglass model of the internet protocol stack and how it may act as a merging technology.

The second challenge is network management in general. Even if all network elements in the networks to be merged happen to provide the same management interfaces, e.g. SNMP or TMN, the overall complexity of the new merged full service network is not to be underestimated. Some management routines may be rationalized due to similar needs from similar network technologies, but optimizing the

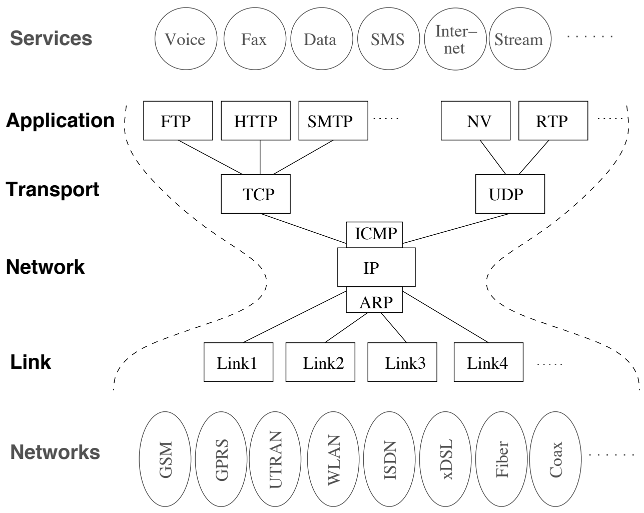


Figure 3. The internet protocol stack as merging technology.

overall network configuration is likely to become an even greater challenge than optimizing configurations of todays single service networks. Frequent reconfiguration of services (deployment and removal) and frequent changes in subscription profiles and hence changes in user behavior, make overall system modelling and analysis extremely difficult if not impossible. Old models and statistical material may no longer be of much value. High traffic rates and complex mixes of traffic classes and flow patterns will make accurate monitoring difficult.

Improvements and extension of todays network management standards and systems are essential for operators both to manage the general growth of system complexity and to gain necessary control over their future full service networks. One such potential extension is *emergent behavior* based management implements.

This thesis contributes to the design process of emergent behavior base management implements, or more specifically *swarm intelligence* based management implements suitable for fault management, configuration management and to some degree traffic management. *Swarm intelligence* systems are typical emergent behavior based systems (see Section 1.2).

1.2 Emergent Behavior for Complex Problems

There is no precise definition of *emergent behavior*, but it is generally understood to be the resulting overall behavior generated by many simple behaviors interacting in some way. *Simple behavior* should be interpreted as a behavior with no true awareness of the overall emergent behavior it is part of. Hence emergent behavior is not easily deductable from descriptions of the simple behaviors generating it.

Taken from [Weg98] a more formal description of emergent behavior is

$$behavior(O1 \mid O2) = behavior(O1) + behavior(O2) + interaction(O1, O2) \quad (1)$$

where $O1$ and $O2$ are simple behaviors relative to the emergent behavior $behavior(O1 \mid O2)$. Hence $behavior(O1 \mid O2)$ is *greater* than just the sum of its sub-behaviors. "New" behavior emerges from interactions between components of simple behaviors.

Emergent behavior may be viewed as a side effect generated by the sum of simple behaviors. An example is a card game. Each player in a card game implements a behavior governed by the rules of the specific game. These behaviors and interactions between the behaviors (players exchanging cards) generate an emergent behavior which results in a side effect. The cards are sorted in some specific order. This side effect is usually undesirable, thus the cards are shuffled between every game played.

Many systems observable in nature may be viewed and described as games, e.g. "game of life" [Con70]. However, the number of participants in such natural games is typically several orders of magnitude larger than those in games invented and played by humans. In biological systems every cell in an organism may be viewed as a participant. Billions of cells play a game of survival which have the side effect of producing a plant, an animal or a human being, i.e. advanced living organisms in nature are emergent behaviors resulting from simple behaviors of a range of simple organisms interacting with each other.

As (1) indicates, emergent behavior is constructed by "gluing" simpler behaviors together with interactions as the gluing mechanism. Hence emergence can also be viewed as a method of construction. In engineering however, emergence has so far to a very limited extent been applied for construction. Hierarchical procedural composition has for centuries been the first choice when constructing, and hence is today widely used and well understood. Hierarchical procedural composition of complex systems commonly involve a top-down designed process using a hierarchy of abstraction levels. Real time software engineering using SDL is an example [ByH93, ITU02]. The top or upper abstraction level describes the whole system by a set of components from the second-upper abstraction level. Each component of the second-upper layer is again described by components from the third-upper layer and so on.

Emergence has no top-down design process, but rather a bottom-up way of constructing what is required to manage complexity. In [Weg98] construction based on emergence is shown to exceed what is possible to achieve using traditional hierarchical procedural composition. Researches from many communities (chemistry, physics, biology) has been aware of the construction power emergence exhibits, and especially biologists since emergence is indeed the most commonly applied construction method in biology. In [Ste90] hierarchical and emergent construction principles are compared, and early theoretical work on emergent functionality is presented. Further theoretical work on emergence can be found in [Cru94b, Cru94a]. An overall historical outline of research on emergence is presented in [Dam00] including references to more theoretical work. For a less theoretical but still general introduction to emergent systems, [Joh01] is recommended.

As mentioned in the preface of this thesis, social animals living in swarms are typical natural systems where emergent behavior is vital and ensures survival. In [PD87] Pasteels and Deneubourg present several examples of activities observed in ant colonies where emergent behavior generates solutions to problems (e.g. finding food and transporting food to the nest) in a stable and efficient manner. Ant colonies are good examples of systems in nature which have been in existence for millions of years much due to the properties most systems of emergent behavior possess. Three dominating properties are:

Adaptiveness The simple behaviors, from which emergent behavior emerges, all implement mechanisms to handle unexpected responses from interactions with the environment. Such mechanisms often include a stochastic component, i.e. by a random choice a response is handle in one out of a set of possible ways. An example is when a ant reaches the end of an ant trail without finding anything of interest. Instead of halting completely or just turning back along the trail, the ant chooses a random direction away from the trail in hope of finding something of interest.

Robustness Three properties of emergent systems ensure great system robustness. Firstly, as already explained in the previous pin, *adaptiveness* avoids system breakdown when the environment changes. Secondly, *weak inter-component dependencies* reduces the probability of system breakdown due to individual component failures. The lack of synchronized hierarchical control implies no single central control component, thus distributed control weakens dependencies between system components. Use of asynchronous indirect communication (by changing the environment) is common in emergent systems. This type of interaction adds to the reduction of inter-component dependencies. And finally, *redundancy* reduces the probability of system breakdown due to individual component failures even further. Many system components have similar/overlapping behaviors. Communication by changing the environment generates distributed and redundant system memory.
Again an ant colony is a good example. No single ant controls the overall behavior of the colony (e.g the queen controls only the production of new ants, [Joh01]), many ants perform similar operations and can replace each other (even the queen is replaceable), and pheromones (chemical substances) are placed along ant trails to exchange information between ants passing along the trail.

Efficiency When encountered with complex problems emergent systems tend to find near optimal solutions with great efficiency. Complex problems should be interpreted as problems having a large solution space with no obvious structure that may be exploited to find good solutions, e.g. problems of the classes NP hard and NP complete. Reasons for this efficiency can be traced to the interplay of positive and negative feedback mechanisms and the stochastic (adaptive) mechanisms already mentioned. For instance, should the random walking ant from above happen to discover a fresh banana, it will encourage other ants to search towards the banana by placement of specific pheromones (positive feedback), and over time an ant trail will develop. The trail will eventually follow a near optimal route from the nest to the banana. When the banana looses its freshness the specific pheromone will no longer be placed (negative feedback) and the ant trail will vanish or be rerouted to a better food source. Overall, the emergent foraging behavior manages to maintain a near optimal network topology of trails leading to the best food sources in the surrounding environment.

Adaptiveness, robustness and efficiency are indeed properties any engineer would like a well designed system to possess. Hence, especially during the last decade, there have been many attempts to engineer systems based on emergent behavior. A community of researchers have been designing systems solving specific complex problems by making them mimic the behavior of social animals. Such systems are denoted *swarm intelligence* systems by Bonabeau et al in [BDT99]. A typical swarm intelligence system consists of a high number of autonomous agents. Note that an autonomous agent in a swarm intelligence context have little resemblence to a traditional agent in a network management system. Autonomous agents are selfcontained software and/or hardware objects which depend only to a limited extent on other objects in their environment. Autonomous agents in a swarm intelligence context do not represent a specific system component (or a set of components) as is typical for a traditional agent in a network management system.

Autonomous swarm agents have simple behaviors strongly influenced by randomness. They search individually for solutions (or subsolutions) to a problem. When a solution is found by an agent, the agent communicates information about the solution (location, quality etc.) to the other agents using indirect communication by leaving messages in the environment. This behavior is iterated, i.e. an agent restarts searching after a solution is found. During search, agents pick up messages left by other agents. A message picked up and read by an agent, will influence how the agent continues its search. No or little information found in messages implies randomness in the search behavior (exploration). Messages giving clear indications of quality solutions within an area of the search space implies search focus in that area (exploitation). Hence randomness, postive feedback and negative feedback applied in iterations guide the overall search process in a swarm intelligence system towards high quality solutions.

Returning to the card game example from above, which illustrates how emergent behavior appears as a side effect of interactions between simpler restricted behaviors, a question arises: *Can true emergent behavior be designed*? A side effect is generally understood to be something unpredicted. Designing, hence predicting, side effects would be a contradiction in terms. However, further philosophical elaboration around this question is left to the reader. Systems designed with emergence in mind are in this thesis considered to be emergent systems as long as they posses the above mentioned properties, i.e. the possession of the properties are considered sufficient gain in overall behavior to argue for the existence of (designed) emergent behavior.

1.3 Swarm Intelligence in Network Management

As already mentioned, systems based of emergent behavior are capable of finding near optimal solutions to complex problems. Many of these complex problems may be formulated as optimization problems, e.g. maintaining an optimal trail topology around an ant hill. A range of these problems relates strongly to classical combinatorial optimization problems, e.g. minimum spanning tree, travelling salesman, sorting and graph partitioning (see e.g. [GJ79] for a list of classical NP-complete problems). In [BDT99] a series of biological systems are studied, modelled and eventually used to develop *swarm intelligence* algorithms and multi agent systems for solving several classical optimization problems.

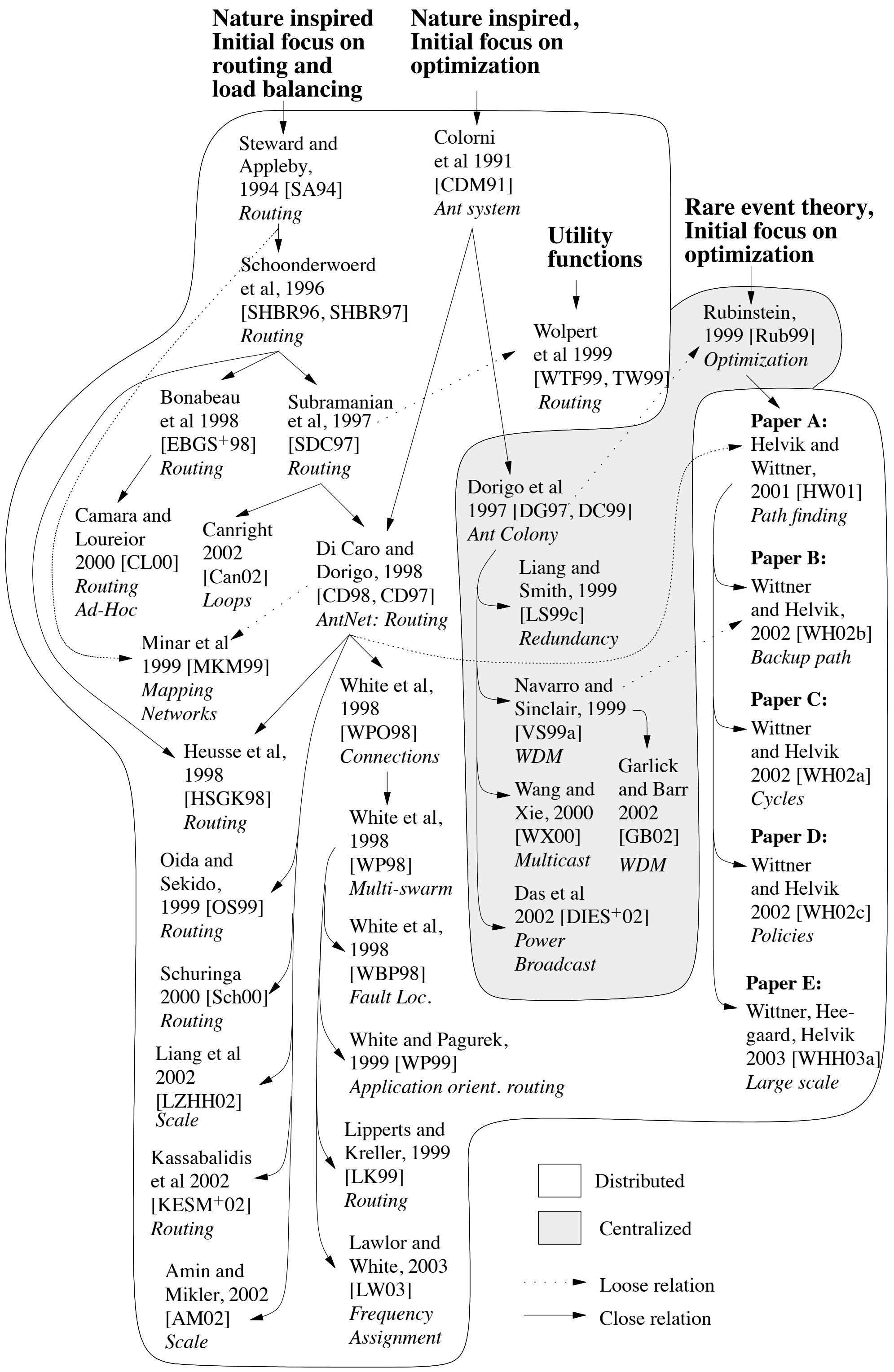
Many challenges within management of computer and telecommunication networks can be formulated as optimization problems. Optimizing the utilization of network resources is one area which has kept researchers busy for decades. Another area is optimal configuration and use of management functions in network management systems. Presenting a comprehensive list of all optimization problems under investigation within the area of computer and telecommunication networks is difficult and out of scope for this thesis, however readers may refer to for instance [Sin99, ALB02] and their citations to get an impression of the variety of such problems.

Most methods developed for solving optimization problems within telecommunications are based on traditional linear, non-linear and integer programming techniques [Wil93]. However, during the last decade the interest for nature inspired optimization methods has increased significantly. One such class of methods is known as evolutionary programs [Mic96] and is founded in Darwin's theories of evolution [Dar95]. A summary of work on evolutionary programs for optimization in telecommunications is presented in [Sin99].

Another class of nature inspired optimization methods is *swarm intelligence* based systems. The interest for, and hence the number of publications of, swarm intelligence based optimization methods solving problems within telecommunications are growing. Figure 4 shows a graph of relevant publications and how they relate to each other. Note that Figure 4 does not show all publications describing swarm intelligence and emergent behavior based systems, but only pioneering publications and key publications with direct focus on problem solving within telecommunications and network management. As indicated in Figure 4, four key pioneering publications have been the source of inspiration for many authors.

Steward and Appleby's work from 1994 [SA94] proposed the idea of creating management systems for routing and load balancing in telecommunications network by mimicking the behavior of insects. Colorni et al [CDM91] already in 1991 presented a combinatorial optimization algorithm inspired by ants and their foraging behavior. Steward and Appleby and Colorni et al's publications have inspired further research on both distributed and centralized systems for network management all based on swarm intelligence.

Wolpert et al [WTF99] presented in 1999 theoretical work and simulation results for systems denoted COllective INtelligences (COINs). Global near optimal behavior emerges in a COIN when individual agents (denoted neurons) learn their behavior by



Nature inspired Initial focus on routing and load balancing
Nature inspired, Initial focus on optimization
Utility functions
Rare event theory, Initial focus on optimization
Steward and Appleby, 1994 [SA94] Routing
Colorni et al 1991 [CDM91] Ant system
Schoonderwoerd et al, 1996 [SHBR96, SHBR97] Routing
Wolpert et al 1999 [WTF99, TW99] Routing
Rubinstein, 1999 [Rub99] Optimization
Bonabeau et al 1998 [EBGS+98] Routing
Subramanian et al, 1997 [SDC97] Routing
Dorigo et al 1997 [DG97, DC99] Ant Colony
Paper A: Helvik and Wittner, 2001 [HW01] Path finding
Camara and Loureior 2000 [CL00] Routing Ad-Hoc
Canright 2002 [Can02] Loops
Di Caro and Dorigo, 1998 [CD98, CD97] AntNet: Routing
Liang and Smith, 1999 [LS99c] Redundancy
Paper B: Wittner and Helvik, 2002 [WH02b] Backup path
Minar et al 1999 [MKM99] Mapping Networks
White et al, 1998 [WPO98] Connections
Navarro and Sinclair, 1999 [VS99a] WDM
Garlick and Barr 2002 [GB02] WDM
Paper C: Wittner and Helvik 2002 [WH02a] Cycles
Heusse et al, 1998 [HSGK98] Routing
White et al, 1998 [WP98] Multi-swarm
Wang and Xie, 2000 [WX00] Multicast
Paper D: Wittner and Helvik 2002 [WH02c] Policies
Oida and Sekido, 1999 [OS99] Routing
White et al, 1998 [WBP98] Fault Loc.
Das et al 2002 [DIES+02] Power Broadcast
Schuringa 2000 [Sch00] Routing
White and Pagurek, 1999 [WP99] Application orient. routing
Paper E: Wittner, Hee-gaard, Helvik 2003 [WHH03a] Large scale
Liang et al 2002 [LZHH02] Scale
Lipperts and Kreller, 1999 [LK99] Routing
Kassabalidis et al 2002 [KESM+02] Routing
Lawlor and White, 2003 [LW03] Frequency Assignment
Amin and Mikler, 2002 [AM02] Scale
Distributed
Centralized
Loose relation
Close relation

Figure 4. Publications on swarm intelligence systems for telecommunications

individual reinforcement learning algorithms based on given utility functions, i.e. no centralized control is involved.

Finally, Rubinstein's work on cross-entropy driven combinatorial optimization [Rub99] presented in 1999 represents the foundation for the research presented in this thesis. Rubinstein's work, and hence research in this thesis, is based on rare event theory and importance sampling.

On the contrary to Steward and Appleby and Colorni et al who based their work on models derived empirically from natural systems, Wolpert et al and Rubinstein founds their work in mathematical theories. Similarities and differences between these two categories of approaches are further discusses in the next section.

2. Thesis Research Focus

The overall research focus of this thesis is, as already indicated, to contribute to the development of methods and algorithms which may enhance todays network management systems and enable them to manage complex networks of the future. Among many potential problem solving techniques, emergent behavior based problem solving has been chosen much due to the valuable properties of emergent systems described in Section 1.2.

The recent increase of interest in the problem solving power of emergent systems attests the significance of the focus chosen in this thesis. Recall the two examples mentioned in Section 1.1, i.e. the *Autonomic Computing* initiative [KZS+01, Res03] and the *BISON* project [BCG+03]. IBM announced a major research initiative denoted *Autonomic Computing* in March 2001 which aims at providing more autonomous computer components which again can enable robust and self-managing systems, i.e. systems where self-management emerge from the interactions between autonomous components. In January 2003 a project titled *Biology-Inspired techniques for Self-Organization in dynamic Networks* (BISON) kicked off. BISON is funded by the Future & Emerging Technologies initiative of the Information Society Technologies Programme of the European Commission, and strives to increase general understanding of emergent systems as well as how specific emergent systems in the computer and telecommunication domain may be constructed.

As already indicated in the previous section, approaches for designing emergent behavior based management systems may be divided into two categories

- System design based on models empirically derived from natural systems
- System design based on models with well understood mathematical foundations

This thesis focuses on an approach from the second category. The next sections describe the two categories in more detail.

2.1 Empirical Construction of Behavior Models

One way of constructing the desired simple behaviors of an emergent system is to mimic existing behaviors found in nature by observation, trail and error. This process may be described in four steps:

1 Find a similar problem in both environments (network management and nature), e.g. problems of routing or sorting.

2 Observe and accept that the natural systems problem solving abilities are stable and efficient, i.e. verify that the three valuable emergent behavior properties from Section 1.2 exist.

3 Create a model, by trail and error, mimicking (as closely as possible) the natural system, and then formalize the model.

4 Tailor the formal model by empirical means (e.g. introduce heuristics) to handle the exact management problem in question.

Work inheriting ideas from Steward and Appleby's pioneer publication [SA94] (the left most branch in Figure 4) and ideas from Colorni et al's publication [CDM91] (second left most branch) is in general based on empirical construction of emergent behavior.

The construction process has one significant advantages. Ethologists have developed and formalized models for many natural behaviors, i.e. a behavior library exists which simplifies work in step 1-3 above. However, the heuristics involved in the approach and the lack of formal foundations may limit the understanding of the core mechanisms in the model, which again can make further development of the model difficult. Lack of formal foundations may also exclude some application areas, since formal proof of operation is difficult. The heuristics in the approach often lead to not well understood parameters requiring configuration, and a trail and error process is usually the only option for deciding reasonable parameter settings.

Several researchers work on developing a better understood formal foundation for emergent behavior systems realized by the empirical process described above. See for instance [ZBMD00, BCD02].

2.2 Formal Founding of Behavior Models

An alternative approach to empirical construction of behavior models, is to ensure a formal foundation from the very start of the construction process:

1 Find a similar problem in both environments (network management and nature), e.g. problems of routing or sorting.

2 Find a formally founded method for solving the problem with proofs of operation (convergence, efficiency and stability).

3 By re-formulations produce a model resembling the natural behavior.

The same advantage as for empirical construction applies. Models developed by ethologists may be used as starting points. However, several new challenges in the construction process are introduced. Finding an appropriate formally founded method may be difficult. If found, reformulating the method may still be difficult, and may alter properties of the original method (e.g. reduce efficiency, stability). Even if difficult, these challenges are manageable. Work presented in this thesis illustrates this.

Among the immediate advantages of ensuring a formal foundation is first of all a better awareness of core mechanisms and model limitations. Better awareness can again make loss of efficiency avoidable when tailoring the model to solve specific problems. Fewer system parameters and a better understanding of the parameters is also to be expected, and hence better control of configuration.

Work inheriting ideas from Rubinstein's publication [Rub99] shown in Figure 4 can be claimed to follow the above described process of construction by initial formal founding. Work in this thesis inherits ideas from Rubinstein, and it follows the above described process however not in a strict manner. Some approximations are made and some parameters are set by heuristic means.

Wolpert et al's work [WTF99, TW99] also presents thorough formal foundations, but can not be claimed to follow the above process. Wolpert et al choose a more general approach with utility functions in focus and do not explicitly aim for mimicking behaviors of natural systems.

3. Thesis Research Methology

As mentioned in the previous section the overall design approach for the algorithm described in this thesis focuses on ensuring a formal foundation and keeping algorithm extensions within limits implied by the foundations. To make sure the approach is followed and sound research results are generated, work presented in each of the included papers follow a traditional research methology.

Work hypotheses Each paper presents a unique research contribution. The contribution has its source in an initial idea, i.e. a hypotheses. The introduction and first sections of each included paper presents such ideas and relevant background information.

Hypotheses testing Relevant test cases are constructed to investigate the value of a research idea. *Monte Carlo simulations* have be chosen as the method of investigation in all papers. There is a combined reason why simulations have been chosen and why analysis only have been applied to a limited extent. Firstly, since the algorithm is constructed for solving complex problems, complex test scenarios must be constructed to test performance properly, i.e. a typical scenario involves NP-hard complexity. NP-hard complexity in general makes analysis difficult. Secondly, the stochastic behavior components in the algorithm together with its distributed construction, disqualify all (to the authors knowledge) relevant modelling techniques for stochastic systems. Hence simulation is the only relevant option for testing the algorithm and its extensions.

Paper A and a short paper included in appendix C describe the simulation tool package "Network Simulator 2" (NS2) used during testing. NS2 is a discrete event simulator providing functionality for realistic simulations of IP network.

Result validation Results from tests of the algorithm are in all papers generated from sets of between 10 and 20 repeated simulations. In cases where results from related research are available and relevant, comparisons are made to validate performance (paper A). Analytic optimal solutions are used for validation when available (paper B and D) as well as validation by comparing and ranking several related algorithm variants and/or test scenarios (paper C and E).

Simulations have been the first choice for testing the algorithm of this thesis, but moving to a real world implementation for further testing does not necessary require great amounts of effort. Section 6 looks into this topic.

4. Guidelines for Reading

All five papers included in part II of this thesis presents closely related work. As already indicated in Figure 4, work presented in [HW01] and reproduced as paper A, describes the foundations for the other four papers, paper B, C, D and E. To make paper B, C, D and E self contained, creating overlap between them and paper A has been necessary. Figure 5 presents a map indicating overlap between different sections of the papers.

Since all papers are self contained they may be read in any sequence. However, reading them in chronological order relative to publication date is a good option, i.e. the sequence "A-B-C-D-E". In any case reading part I first, i.e. this introduction, is recommended.

Note that the appendix in paper C titled "Additional Results" was not a part of the original version of the paper (and is left out from Figure 5). Since the results presented in this appendix have significant value the appendix is considered relevant to include.

Part III of the thesis, appendices, presents unpublished work related to the papers in part II. In appendix A "Autoregression Details" the autoregressive expressions introduced by paper A is re-derived in a less compressed manner to ease readability. Some adjustments to the expressions not mentioned in any of the papers are also presented. Appendix B "Update of Global Values" presents results from intermediate work not included in the papers. Active explicit global state propagation is found not to be significantly more efficient than passive indirect propagation. Appendix C "Paper F" reproduces a short paper describing the simulator tool package used to generate simulation results in paper A-E. Finally, appendix D "Implementing Technologies" presents a short survey on potential technologies for realizing emergent behavior based network management implements.

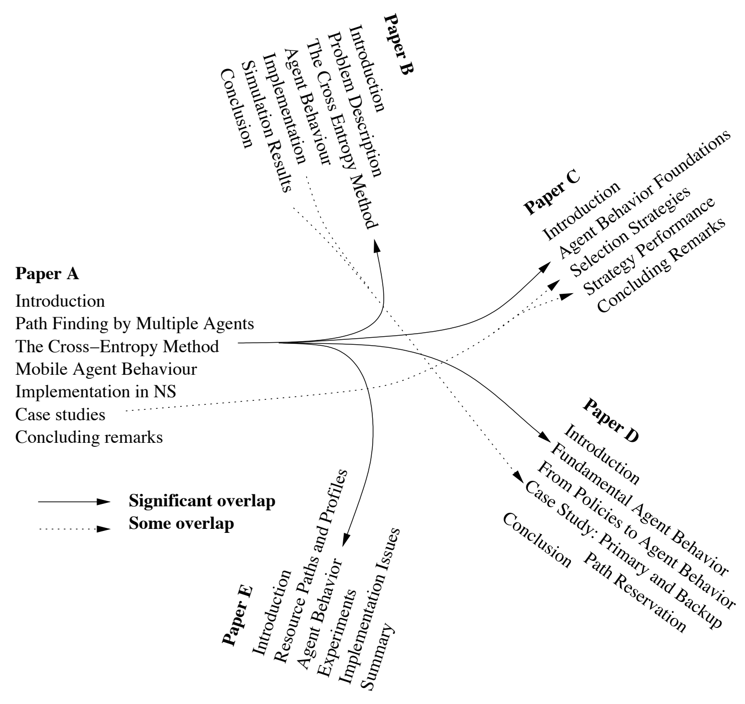


Figure 5. Overlap between section in papers included in part II.

5. Contributions

The following paragraphs present the main contributions of each of the five papers included in part II of this thesis. Figure 6 illustrates schematically how the major contributions in the papers relates to the overall algorithm.

Paper A

Using the Cross-Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks

The problem of finding paths in networks is general and many faceted with a wide range of engineering applications in communication networks. Finding the optimal path or combination of paths usually leads to NP-hard combinatorial optimization problems.

The cross-entropy method proposed by Rubinstein, manages to produce optimal solutions to such problems in polynomial time. However this algorithm is centralized and batch oriented. In this paper we show how the cross-entropy method can be reformulated to govern the behaviour of multiple mobile agents which act independently and asynchronously of each other. The algorithm represents the first relation between cross entropy driven optimization and distribute swarm intelligence systems,

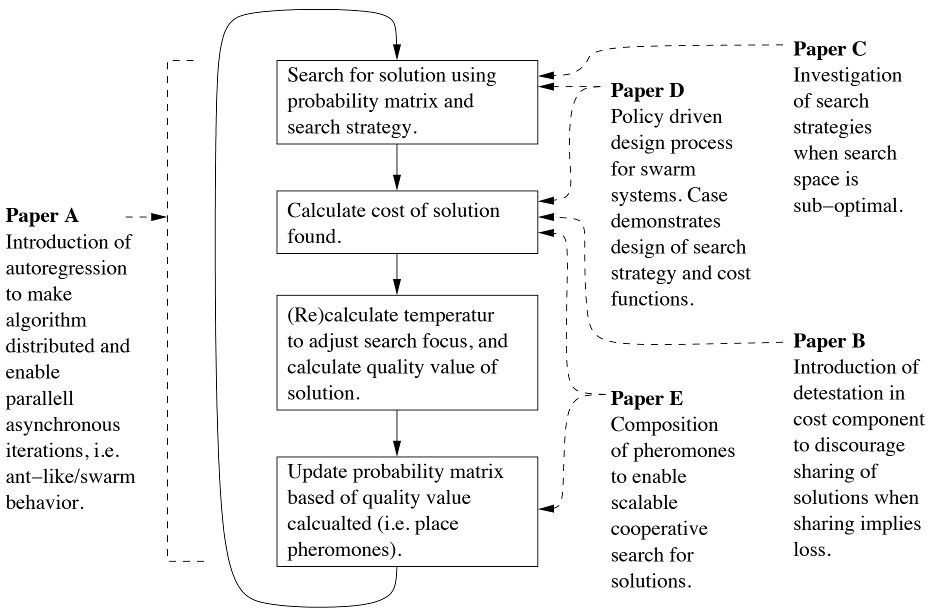

Search for solution using probability matrix and search strategy.
Calculate cost of solution found.
(Re)calculate temperatur to adjust search focus, and calculate quality value of solution.
Update probability matrix based of quality value calcualted (i.e. place pheromones).
Paper A Introduction of autoregression to make algorithm distributed and enable parallell asynchronous iterations, i.e. ant–like/swarm behavior.
Paper C Investigation of search strategies when search space is sub–optimal.
Paper D Policy driven design process for swarm systems. Case demonstrates design of search strategy and cost functions.
Paper B Introduction of detestation in cost component to discourage sharing of solutions when sharing implies loss.
Paper E Composition of pheromones to enable scalable cooperative search for solutions.

Figure 6. Schematic overview relating contributions from the included papers to the overall algorithm.

and may be regarded as a proof of concept. Compared to other similar swarm intelligence systems the algorithm is unique in two ways: It has a formal foundation, and it has a search process of two stages where the second stage adjusts a temperature parameter (comparable to a “simulated annealing” temperature) which regulates the search focus of the algorithm in a highly efficient manner.

The algorithm is evaluated on a set of well known Travelling Salesman Problems. A simulator, based on the Network Simulator package, has been implemented which provide realistic simulation environments. Results show good performance and stable convergence towards near optimal solutions for the problems tested.

Paper B

Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks

Telecommunication network owners and operators have for half a century been well aware of the potential loss of revenue if a major trunk is damaged, thus dependability at high cost has been implemented. A simple, effective and common dependability scheme is 1:1 protection with 100% capacity redundancy in the network.

A growing number of applications in need of dependable connections with specific requirements to bandwidth and delay have started using the internet (which only provides best effort transport) as their base communication service. In this paper we adopt the 1:1 protection scheme and incorporate it as part of a routing system applicable for internet infrastructures. By load sharing 100% capacity redundancy is no longer required.

A distributed stochastic path finding (routing) algorithm based on swarms of cross entropy guided ant-like agents is presented in the paper. By making species of agents sensible to each others path resource requirements (e.g. bandwidth), the algorithm is the first to enable a fully distributed swarm intelligence based search system where species of agents cooperately, by detesting each other, find valuable combinations of paths. The algorithm enables realization of optimal load sharing as well as differentiation between classes of traffic without need of centralized management.

In this paper the algorithm is applied in a routing system, which finds near optimal patterns of primary and backup paths for a given set of connections, such that single link failure protection is realized in a mesh network. Results from Monte Carlo simulations of a scenario, where far less than 100% bandwidth capacity redundancy exist, indicate that the algorithm indeed is capable of finding pairs of independent primary and backup paths satisfying specific bandwidth constraints.

Paper C

Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks

In this paper an extended version of the algorithm in paper A capable of finding cyclic paths in scarcely meshed networks is described. Cyclic paths are especially interesting in the context of protection switching (Grover et al' *P-cycles* [GS98]), and scarce meshing is typical in real world telecommunication networks. Two new next-node-selection strategies for the ant-like agents, believed to better handle low degrees of meshing, are introduced and compared to the original strategy applied in paper A. The original strategy terminates agents when they reach a dead end during forward search. The first new strategy applies a "backtrack-and-retry" technique whenever an agent reaches a dead end. The second new strategy allows an agent to revisit nodes to escape dead ends, i.e. construction of infeasible solutions are allowed.

Performance results from Monte Carlo Simulations of systems implementing the strategies are presented. Results show that the second new strategy outperforms both the original and the first new strategy. Hence the paper clearly demonstrates that allowing infeasible intermediate solutions (in this case cyclic paths with loops) may enable efficient constructions of near optimal feasible solutions.

Paper D

Robust Implementation of Policies using Ant-like Agents

Policy based management is a powerful means for dealing with complex heterogeneous systems. However, the policies are commonly strictly interpreted, and it is tempting to resort to centralized decisions to resolve conflicts. At the same time, swarm intelligence based on “ant like” mobile agents has been shown to be able to deal with challenging optimization and trade-off problems.

This paper is the first to discuss and demonstrate how policies may be used to govern the behavior of mobile agents in a swarm intelligence based system, such that near optimal solutions for the implementation of the (potentially conflicting) policies are found in a truly distributed manner. By this a more dependable and robust system is obtained. The enforcement of the policies is soft in the sense that it is probabilistic and yields a kind of “best effort” implementation. To the authors knowledge probabilistic “best effort” policy conflict resolution is conceptually new and first introduced by this paper.

A case study illustrating how ant like mobile agents may implement load distribution and conflict free back-up policies, is presented.

Paper E

Scalable Distributed Discovery of Resource Paths in Telecommunication Networks using Cooperative Ant-like Agents

Future user controlled development of telecommunication services combined with powerful terminal equipment result in many heterogenous services running in a peer-to-peer execution environment. Locating a desired service in such an environment is challenging. In this paper a cross entropy driven swarm based optimization algorithm is presented capable of finding paths of resources in a complex network environment. On the contrary to existing localization mechanisms for peer-to-peer systems the algorithm considers all accessed resources between (and including) the client side and server side when a resource path is evaluated.

Scalability is achieved by enabling agents to cooperate during search when they have overlapping search profiles. On the contrary to cooperation by detestation presented in paper B, this paper presents an algorithm where agents share and cooperatively construct a pheromone “road map” by which relevant near optimal paths of resources may be found. A solution (a path) is identified by a combination of pheromones each relating to a QoS parameter in the relevant search profile. By this in total several orders of magnitude fewer unique pheromone types are required, and hence the number of state values to be managed by network nodes can be significantly reduced. Even so the number of unique search profiles are virtually unlimited. The new cooperative behavior is realized without invalidating the formal foundations of the algorithm.

Early results from simulations are very promising. The expected cooperative behavior is shown to be present, i.e. a set of near optimal resource paths conforming to a set of different but overlapping search profiles are found efficiently. Comparisons of scenarios with none-cooperative and cooperative agents give clear indications that cooperation lead to improved performance.

Table 1. Number of operations performed and state variables updated in one iteration of the algorithm (i.e. one search and backtracking sequence performed by one agent). N is the number of nodes involved in a solution. d is the connection degree of the network. S is the number of solution types.

	Operations		
	Exp.	Multiplications	Additions
Forward search	1	$2 + 4Nd$	$N(4d - 1)$
Backtracking	3	$11 + N(6d + 4)$	$9 + 2N$
Total of unique ($N = 1,\ d = 1$)	4	27	14

	State i node	State in agent	
	Floats	Floats	Integers
Forward search	0	3	N (tabulist)
Backtracking	$S(3 + 3d)$	4	N (tabulist)

6. Implementation Issues

Paper A together with appendix A “Autoregression Details” present the formal foundations for the algorithm in this thesis. When studying these parts a reader may get an initial impression that the overall algorithm is difficult to implement. This is not at all the case. Even though the process of deriving the autoregressive expressions has many steps, the actual autoregressive schemas output are uncomplicated. Table 1 indicates the number and type of mathematical operations and state values required by the algorithm.

To understand the content of Table 1 and short introduction to the basic principles of the algorithm is required. The algorithm is realized by having one (or several) ant-like agent search alone (or in parallel) for solutions of a certain type or several types (S represents the number of types). Each agent needs to visit a set of nodes in a graph to compose a solution. N is the number of nodes visited. Each agent builds a solution by completing a forward search phase visiting a set of nodes. A backtracking phase is then entered where the agent revisits every node in the set visited in the forward search phase. During backtracking the type and the quality of the solution found is registered in all nodes visited. Table 1 separates between operations performed and state information updated during the forward search phase and the backtracking phase. To find a near optimal solutions for complex problems the above process is iterated, i.e. the agents search for solutions (and backtrack) again and again until a solution of satisfactory quality is found. See paper A for a comprehensive description of the algorithm.

As Table 1 indicates, the number of operations required are in general very limited. The row titled “Total of unique” presents the total number of unique operations required to be performed per iteration. From this the simplicity of an agent is quite visible. Only 4 exponential operations, 27 multiplications and 14 additions are required to implement an agent, excluding logic required for the agent to navigate between nodes.

State information required in an agents memory is also very limited, only in total 7 floating point values and one integer per node visited. However, the amount of state information required to be store in a node at all times is potentially large and very much influenced by the number of solution types S being searched for in parallel. This is addressed further in a paragraph below.

For problem solving in a network management context nodes and graph will often map one-to-one with network elements and a communication network respectively. Hence many nodes are typically placed at physically different locations. To implement the algorithm in such an environment there are three possible approaches.

- All operations are realized by service functionality in nodes, and agents become just messages containing state information.
- Some operations are implemented in the nodes and some carried by the agent. Hence agents carry operations and state while they move around. Such agents are known as *mobile agents* [PK98].
- All operations are carried by agents, and hence only state information is stored in nodes.

The first approach is likely to be most efficient and most secure. Operations can be tailored/compiled to run efficiently at each specific node, and well known security mechanisms for messaging can be applied. Technically however, every node must be extended/upgraded with new functionality (operations) whenever a new variant of the algorithm is to be made operative.

The last approach resembles how many emergent systems in nature are implemented. An ant carries all the operations required in its brain and body, and it makes changes to the environment when it moves around. This approach enables fast activations of new algorithm variants by just releasing a new type of agents into the network. No upgrade of node functionality should be required. However, the flexibility gained results in a need for untraditional security mechanisms protecting nodes from malicious agents and opposite.

The second approach is a combination of the first and the last approach. Hence enabling relevant nodes in a network to handle implementations by the second approach also enables the other approaches. Appendix D "Implementing Technologies" presents a short survey on potential technologies which may realize the second approach (and hence the other approaches too).

Different solution types being searched for in parallel can in a network management context for instance be connections and/or services. Hence the number of solution types S may potentially be very large, e.g. in the order of hundred million for a nation wide network in a relative small country like Norway. Paper E looks into and suggests solutions to this scalability problem.

7. Summary and Conclusion

The continuous increase of overall complexity in computer and telecommunication systems is becoming a severe challenge for todays management systems. Operators and researcher agree that to coupe with this increase, management systems of the future are required more than ever to be *adaptive*, *robust* and highly *efficient*. These properties match properties found in systems base on *emergent behavior*. Emergent behavior systems are common in nature and consists of many simple (relative to the overall system behavior) autonomous components interacting with each other. Component distribution and redundancy result in robustness, moments of stochastic behavior boost the systems adaptability, and clever interaction schemes based on positive and negative feedback ensures high efficiency.

Enhancing future management systems with emergent behavior based implements has great potential. Results from research in this area are promising. However, designing emergent behavior based implements with specific behaviors is not at all trivial. In this introduction two design approaches for emergent behavior based systems have been discussed. The first approach which by empirical means adopts and tailors emergent behaviors found in natural systems, has so far been applied to construct implements for several management problems, *routing* being the most treated problem area. However due to the empirical means applied, limited understanding exist of the core mechanisms driving the systems designed. In the second approach discussed empirical means are replaced by formal founding and formally sound tailoring of the foundations. Hence the second approach enables a designer to better understand core mechanisms and avoid miss-configuration of the designed system.

Work presented in the five included papers of this thesis strive to follow the second approach. A formal foundation in rare event theory has been chosen, and a fully distributed multi-criteria optimization algorithm has been developed. In addition to having the properties mentioned above, the algorithm is simple to implement. The behaviors of the ant-like agents in the algorithm are simple, and only a limited amount of new server functionality must be installed in the network components of the system to be optimized.

Performance results for the algorithm are very promising. Near optimal solutions to NP-hard multi-criteria optimization problems are found efficiently. However there is still work to be done. The the next section presents future research tasks.

8. Future Work

The development of the algorithm presented in this thesis is far from complete. Many challenges are still ahead. The following list of future work indicate a set of areas where more research is required. The first seven relates to core development of the algorithm while the latter three suggests/relates to potential application areas.

Parameter tuning A small set of parameters are required to be configured for the algorithm to run efficiently. Looking further into how (near) optimal values for theses parameters can be ensured is important, and can potentially increase the performance of the algorithm.

Including heuristics Adopting empirical techniques to a greater extent may be of interest and could boost performance of the algorithm to some degree. However, as discussed in Section 2, if care is not taken this may lead to reduced understanding of the algorithm's core behavior.

Dynamic networks So far during simulations there has been only a limited level of dynamics in the network environments. Test scenarios with different levels of dynamics should be designed. One such class of scenarios include traffic sources configured to inject a certain amount of traffic in certain patterns into a network topology.

Large scale environments Even if large scale challenges have been address in paper E, no thorough large scale test has been performed so far. A set of representative network environments of large scale should be chosen and a set of relevant simulation scenarios designed.

Cost efficiency The algorithm is show in paper B and D to be suitable as an implement in a system for resource management in a distributed network environment. Cost efficiency is an important aspect of any resource management system. To what degree does the system ensure utilization of resources compared to the amount of resources consumed by the system itself? So far the cost efficiency of the algorithm has not been address. Relevant scenarios should be designed where the amount of network resources consumed by the ant-like agents is measured and compared to overall resource management efficiency.

Real world prototype To verify that the algorithm is as simple to implement as expected a real world prototype should be developed. By cooperating with other research centres even a (to some degree) large scale real world experiment may be realizable.

Relation to COINs Better understanding of the relationships between Wolpert et al's COllective INtelligences and the foundation of the algorithm in this thesis is desirable. Comparing and examining the detestation based cost functions from paper B and typical utility functions of COINs systems may give valuable insight in the general relationship between COINs and swarm intelligence algorithms.

P-cycle design Grover and Stamatelakis introduce in [GS98] the idea of establishing protection cycles in meshed networks to realize protection against single link failures. So far to the authors knowledge no distributed algorithm exists that find optimal P-cycle designs, i.e. the set of cycles which minimize redundancy and loss on failures. The algorithm presented in this theses may potentially be tailored to find such sets of cycles.

Routing in Ad-hoc networks Considering the results in paper B, the algorithm may be used as a foundations for an ad-hoc routing scheme. If a relevant set of backup paths can be found fast enough, keeping a connection up while roaming in a highly dynamic network environment may be possible.

Formalizing policy-to-ant design process Since policy management are claimed by many to be the management approach an operator truly needs, putting more effort into establishing an efficient design process for ant-like agent implementations of policies may be valuable, i.e. proceed with the work presented in paper D. Indeed if a clear and unambiguous process description can be established, it should likely give strong indications to how specific emergent systems in general may be designed.

II

INCLUDED PAPERS

PAPER A

Using the Cross-Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks

Bjarne E. Helvik and Otto Wittner

Third International Workshop on Mobile Agents for Telecommunication Applications, MATA 2001

August 14-16th 2001, Montreal, Canada.

USING THE CROSS-ENTROPY METHOD TO GUIDE/GOVERN MOBILE AGENT'S PATH FINDING IN NETWORKS

Bjarne E. Helvik

Otto Wittner

Abstract The problem of finding paths in networks is general and many faceted with a wide range of engineering applications in communication networks. Finding the optimal path or combination of paths usually leads to NP-hard combinatorial optimization problems. A recent and promising method, the cross-entropy method proposed by Rubinstein, manages to produce optimal solutions to such problems in polynomial time. However this algorithm is centralized and batch oriented. In this paper we show how the cross-entropy method can be reformulated to govern the behaviour of multiple mobile agents which act independently and asynchronously of each other. The new algorithm is evaluate on a set of well known Travelling Salesman Problems. A simulator, based on the Network Simulator package, has been implemented which provide realistic simulation environments. Results show good performance and stable convergence towards near optimal solution of the problems tested.

1. Introduction

The problem of finding paths in networks is general and many faceted with a wide range of engineering applications in communication networks. Examples: end to end paths in (virtual) circuit switched networks both for primary paths and backup path in SDH, ATM and MPLS, routes in connectionless networks, shortest (or longest) tours visiting all nodes (STST). Path is used as a collective term encompassing a number of the more specific technical terms path, route, circuit, tour and trajectory.

Finding the optimal path or combination of paths usually leads to NP-hard combinatorial optimization problems, see for instance [Bal95a, Bal95b]. A number of well known methods exist for solving these problems, e.g. simulated annealing, [KGV83], tabu search [Glo96] genetic algorithms [Gol98] and the Ant Colony System [DG97]. A recent and promising method, the cross-entropy method, is proposed by Rubinstein which finds a near optimal solution in polynomial time (O(3)) [Rub99]. However, when we implement path finding as a management functionality of a network, we have another additional requirement, which is not easily met by the above algorithms:

- The algorithm should be distributed, i.e. the path should be decided by a cooperative task among the network elements. This increases the dependability of the network by avoiding the single point of failure of a centralized network management system, and by avoiding the management system to rely on the network being managed.

Multiple mobile agents, exhibiting an insect like swarm intelligence, has been proposes as a means to path finding in communication networks in a distributed and adaptive manner [SHBR97][CD98][Sch00][WPO98]. Hereto, these mobile agent systems have concentrated on solving the shortest path routing problem. A more general approach is desirable to enable implementation of a wider range of management applications. Constructing systems capable of finding good solutions to the travelling salesman problem (TSP) may fulfil this generality since TSPs are among the hardest routing problems (NP complete).

In this paper we will show how the cross-entropy method of [Rub99], which has been evaluated successfully on TSPs, can be reformulated to govern the behaviour of multiple mobile agents towards finding optimal paths in networks. This reformulation is presented in section 4. How this behaviour is implemented in the Network Simulator [WH00] is presented in section 5. The ability to find (near) optimal paths are demonstrated through some case studies in section 6, before we conclude. First, however, an introduction to path finding by multiple agents is given in section 2 and a brief review to the cross-entropy method in section 3.

2. Path Finding by Multiple Agents

Schoonderwoerd & al.'s paper [SHBR97] introduces the concept of multiple mobile agents cooperatively solving routing problems in telecommunication networks. A number of simple agents move themselves from node to node in a network searching for paths between a given pair of source and destination nodes. A probability matrix, represented as probability vectors in each node, controls the navigational behaviour of the agents. When a path is found the probability matrix is adjusted according to the quality of the path such that a better path will generally have a higher probability of being reused. By iterating this search process high quality paths emerge as high probabilities in the matrix.

We regard a network with n nodes with an arbitrary topology, where the only requirement is the it is feasible to establish the required path. A link connecting two adjacent nodes k, l has a link cost L_{kl}. The link cost may be in terms of incurred delay by using the path, "fee" paid the operator of the link, a penalty for using a scare resource like free capacity, etc. or a combination of such measures.

Path i through the network is represented by $\pi_i = \{r_1, r_2 \ldots, r_{n_i}\}$ where n_i is the number of nodes traversed. For a TSP tour $n_i = n + 1$, $\forall i$ and $r_{1_i} = r_{n_i}$.

The cost function, L, of a path is additive,

$$L(\pi_i) = \sum_{j=1}^{n_i - 1} L_{r_j r_{j+1}} \tag{1}$$

The foraging behaviour of ants has so far been the major inspiration for all research on multi mobile agent systems for routing. When an ant has found a food source it marks the route between its ant hill and a food sources with a pheromone trail. Other ants searching for food will with a higher probability follow such a trail than move about randomly. On their way home from the food source they will reinforce the pheromone trail and increase the probability of new ants following the trail.

Viewing mobile agents as artificial ants and network nodes as the environment we can interpret pheromone trails as routing probabilities. We have an unconditioned probability $p_{t,rs}$ of an agent choosing to go to node s when it is in node r at time t. The actual choice of next node may be conditioned on the agents past history according to the selection strategies of the agents (section 4.2). We denote the set of unconditional routing probabilities as $\mathbf{p}_t = \{p_{t,rs}\}_{\forall rs}$. The probability of choosing a specific path, π, under the current selection strategies is $p_t(\pi)$ which is uniquely determined by $\mathbf{p}_t$.

3. The Cross-Entropy Method

A new and fast method, called the cross-entropy method, for finding the optimal solution of combinatorial and continuous nonconvex optimization problems with convex bounded domains is introduced by Rubinstein [Rub99]. To find the optimal solution, a sequence of simple auxiliary smooth optimization problems are solved based on Kullback-Leibler cross- entropy, importance sampling, Markov chains and the Boltzmann distribution. In the rest of this section we review the method and state some results in the context of the problem at hand. For details it is referred to [Rub99].

The basic notions of the method is that in a random search for an optimal path, the probability of observing it is a rare event. For instance, finding the shortest travelling salesman tour in a fully meshed network with 25 nodes and an uniformly distributed routing probability from one node to the next is $1/25! \approx 10^{-25}$. Hence, the probability of observing the optimal path is increased by applying importance sampling techniques [Hei95]. However, doing this in a single step is not feasible. A performance function of the current routing probabilities, $h(\mathbf{p}, \gamma)$, is introduced:

$$h(\mathbf{p}, \gamma) = E_p(H(\gamma, \pi)) \tag{2}$$

which is based on the Boltzmann function:

$$H(\gamma, \pi) = \exp(-\frac{L(\pi)}{\gamma}) \tag{3}$$

In (3) $L(\pi)$ is denoted the potential function and γ the control parameter or temperature. It is seen that as the temperature decreases an increasing weight is put on the smaller path costs, see Fig. 1.

A temperature is determined which puts a certain emphasis on the shorter routes, i.e. the minimum temperature, γ_t, which yields a sufficiently low performance function

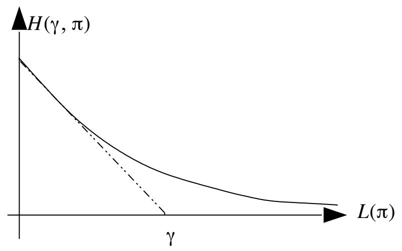


Figure 1. Illustration of the Boltzmann function

$$\min\ \gamma_t\ s.t.\ h(\mathbf{p}^*_{t-1}, \gamma_t) \geq \rho \tag{4}$$

where $10^{-6} \leq \rho \leq 10^{-2}$ and $\mathbf{p}^*_{t-1}$ is the current routing probabilities. The index t indicates the step in the iteration procedure and the initial routing probabilities $\mathbf{p}^*_0$ is chosen to be uniformly distributed.

It is shown, [Rub99], that the set of routing probabilities $\mathbf{p}^*_t$ which is the solution to

$$\max_{\mathbf{p}_t}\ E_{\mathbf{p}_{t-1}} \left(\prod_{r_i \in \pi} H_{r_j r_{j+1}}(\gamma) \sum_{r_i \in \pi} \ln p_{t, r_j r_{j+1}} \right) \tag{5}$$

will minimize the cross entropy between the previous routing probabilities, $\mathbf{p}^*_{t-1}$ weighted with the performance function and $\mathbf{p}^*_t$, and represents an optimal shift in the routing probabilities towards estimating the performance function with temperature γ. In the above it is used that the path cost is an additive function which enables the Boltzmann function to be rewritten as

$$H(\gamma, \pi) = \exp(-\frac{L(\pi)}{\gamma}) = \prod_{r_i \in \pi} \exp(-\frac{L_{r_j r_{j+1}}}{\gamma}) = \prod_{r_i \in \pi} H_{r_j r_{j+1}}(\gamma) \tag{6}$$

It is shown that the solution to (5) is

$$p^*_{t,rs} = \frac{\sum_{\forall j:(\{r,s\} \in \pi_j)} \prod_{i=1}^{n_j-1} H_{r_i r_{i+1}}(\gamma) p^*_{t-1, r_i r_{i+1}}}{\sum_{\forall j:(\{r\} \in \pi_j)} \prod_{i=1}^{n_j-1} H_{r_i r_{i+1}}(\gamma) p^*_{t-1, r_i r_{i+1}}} \tag{7}$$

An optimal shift of routing probabilities, $\mathbf{p}^*_t$, toward the lower cost paths is obtained. We may now increment the iterator, $t \leftarrow t+1$, lower the temperature by employing (4) to shift the emphasis further toward the smaller costs and find an improved set of routing probabilities. Hence, an iterative procedure is obtained which yields a sequence of strictly decreasing temperatures, $\gamma_1 > \gamma_2 > \ldots > \gamma_t > \ldots$ and a series of routing probabilities $\mathbf{p}^*_0, \ldots, \mathbf{p}^*_t, \ldots$ which almost surely convergence to the optimal solution [Rub99], where

$$p^*_{t \to \infty, rs} = \begin{cases} 1, & \{rs\} \in \pi^*, L(\pi^*) = \min_{\forall \pi} L(\pi) \\ 0, & \text{otherwise.} \end{cases}$$

Note that the above outlined method employs a global random search procedure, which is different from the local search heuristics of other well known random search algorithms for global optimization like simulated annealing, tabu search and genetic algorithms.

The procedure outlined is by Rubinstein applied in a batch oriented manner, i.e. a sample of N paths[1], is drawn from $\widehat{\mathbf{p}}_t^*$. On this basis the temperature is determined by the stochastic counterpart of (4), i.e.

$$\min \gamma_t \; s.t. \; N^{-1} \left(\sum_{j=1}^{N} \prod_{r_i \in \pi} H_{r_j r_{j+1}}(\gamma) \right) \geq \rho$$

and routing probabilities by the stochastic counterpart of (7), i.e.

$$\widehat{p}_{t,rs}^* = \frac{\sum_{j=1}^{N} I(\{r,s\} \in \pi_j) \prod_{i=1}^{n_j - 1} H_{r_i r_{i+1}}(\gamma_t)}{\sum_{j=1}^{N} I(\{r\} \in \pi_j) \prod_{i=1}^{n_j - 1} H_{r_i r_{i+1}}(\gamma_t)}$$

where $I(\cdots)$ is the indicator function. Rubinstein reports that empirical studies suggest the cross entropy method to have polynomial, in the size of the problem running time, complexity, e.g. O(3).

The above result is valid both for deterministic link costs and for stochastic link costs [Rub01]. Hence, the cross-entropy method may be used to find optimal paths in networks were the link costs are random variables like queuing delays and unused capacity. The application to such networks (obviously) is at the cost of larger sample sizes and/or more iterations.

4. Mobile Agent Behaviour

Studying the cross entropy method, it is seen that it forms the basis for a distributed implementation in a network using multiple simple mobile agents. The destination node of the agents keep track of the temperature. The agents move through the network according to the routing probabilities and the path selection requirements/constraints. For each path followed the cost is accumulated, cf. (1), which reflects the quality of the path. When a certain number of such paths have been found the temperature and the routing probabilities are updated.

However, a batch oriented decision of new temperature and new routing probabilities based on the information collected by a large number of agents, e.g. several thousands, is contrary to the basic ideas of swarm intelligence and is unsuited since it delays the use of the collected information, incurs storing of a large number of agents midway in their life cycle and a load peak when a probability update takes place. It also hampers the cooperation between families of agents. An incremental update of temperature and path probabilities is required.

[1]N is typically chosen in the order of $10 \cdot n \cdot m$ to $n \cdot m$, where n is the number of nodes in the network and m is the average number of outgoing links per node.

4.1 Autoregressive Distributed Computations

To meet the requirement of an incremental update of temperature and routing probabilities, we have introduced autoregressive stochastic counterparts of (4) and (7).

When an agent reaches its destination node the autoregressive performance function, h_i is updated as

$$h_0 = \beta \cdot h_{-1} + (1-\beta) \cdot H(\gamma_0, \pi_0) \tag{8}$$

where, for the sake of notational simplicity the last arriving agent is indexed 0, second last -1, etc., and $\beta \in [0,1]$ is the autoregressive memory factor, typically close to one.

In (8) the temperature after the agent has arrived is used immediately. If we had M previously arriving agents, replace $h(\mathbf{p}^*_{t-1})$ by h_0 in (4) and use (3), (4) may be rewritten as

$$\min \gamma_t \; s.t. \; \frac{1-\beta}{1-\beta^{M+1}} \sum_{i=-M}^{0} \beta^{-i} \exp(-\frac{L(\pi_i)}{\gamma_0}) \geq \rho \tag{9}$$

It is seen that the minimum is at equality. The equation is unsuited for solving in a network node since it is transcendental and the storage of a potentially infinite number of path costs $L(\pi_i)$ is required. Assuming that the inverse of the temperature does not change radically, a first order Taylor expansion of each term in (9) around the inverse of the temperature which were current when the corresponding agent arrived, is carried out, i.e.

$$\begin{array}{rcl} \rho\frac{1-\beta^{M+1}}{1-\beta} & \approx & \sum_{i=-M}^{0} \beta^{-i} \exp(-\frac{L(\pi_i)}{\gamma_i}) \left(1 - L(\pi_i)\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_i}\right)\right) \\ & = & A - \frac{1}{\gamma_0} B + \exp(-\frac{L(\pi_0)}{\gamma_0}) \\ & \approx & A - \frac{1}{\gamma_0} B + \exp(-\frac{L(\pi_0)}{\gamma_{-1}}) \left(1 - L(\pi_0)\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_{-1}}\right)\right) \end{array} \tag{10}$$

It is seen that the implicitly defined constants in (10) maintain the history of Boltzmann function values and temperatures. An approximation of γ_0 is obtained from (10) and we arrive at the following scheme to compute the current temperature for each arriving agent:

$$\begin{array}{rcl} \gamma_0 & = & \frac{B \cdot \exp(\frac{L(\pi_0)}{\gamma_{-1}}) + L(\pi_0)}{1 + \frac{L(\pi_0)}{\gamma_{-1}} + \exp(\frac{L(\pi_0)}{\gamma_{-1}})\left(A - \rho\frac{1-\beta^{M+1}}{1-\beta}\right)} \\ A & \leftarrow & \beta A + \left(1 + \frac{L(\pi_0)}{\gamma_0}\right) \exp(-\frac{L(\pi_0)}{\gamma_0}) \\ B & \leftarrow & \beta B + L(\pi_0) \exp(-\frac{L(\pi_0)}{\gamma_0}) \\ \gamma_{-1} & \leftarrow & \gamma_0 \\ M & \leftarrow & M + 1 \end{array} \tag{11}$$

where the initial values are $A = B = M = 0$ and $\gamma_{-1} = -L(\pi_0)/\ln(\rho)$.

Similarly, after having updated the current temperature of its destination node, the agent backtracks along its path π_0 and updates the probabilities $p_{0,rs}$ according to an autoregressive stochastic counterparts of (7),

$$\widehat{p}^*_{0,rs} = \frac{T_{rs}}{\sum_{\forall s} T_{rs}} \tag{12}$$

$$T_{rs} = \sum_{i=-M}^{0} I(\{r,s\} \in \pi_0)\beta^{-i} \exp(-\frac{L(\pi_i)}{\gamma_0}) \tag{13}$$

Due to the constant temperature regime of (13) we have the same infeasible storage and computing requirements as when solving (9). Thus again we assume γ not to change radically and apply a second order Taylor expansion to each term, i.e. (13) is approximated by

$$\sum_{i=-M}^{0} I(\{r,s\} \in \pi_i)\beta^{-i} \exp\left(-\frac{L(\pi_i)}{\gamma_i}\right) \quad \left(1 - L(\pi_i)\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_i}\right) + \frac{L^2(\pi_i)}{2}\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_i}\right)^2\right)$$

The second order expansion is used to better approximate the hyperexponential numerator and denominator of $\widehat{p}^*_{0,rs}$ and hence avoid non-physical (negative) values of T_{rs} in case of a rapid decay of the temperature. However, this may result in a non-physical increase of the approximation of T_{rs} as $1/\gamma_0$ increases. Hence, when the derivative of the approximation above becomes positive, it is replaced by its minimum which yields:

$$T_{rs} \approx I(\{r,s\} \in \pi_0) \exp\left(-\frac{L(\pi_0)}{\gamma_0}\right) + A_{rs} + \begin{cases} -\frac{B_{rs}}{\gamma_0} + \frac{C_{rs}}{\gamma_0^2}, & \frac{1}{\gamma_0} < \frac{B_{rs}}{2C_{rs}} \\ -\frac{B_{rs}^2}{4C_{rs}}, & \text{otherwise} \end{cases} \tag{14}$$

where, as for the temperature, we have an autoregressive updating scheme for the parameters yielding the second order approximation:

$$\begin{aligned} A_{rs} &\leftarrow \beta A_{rs} + I(\{r,s\} \in \pi_0) \exp\left(-\frac{L(\pi_0)}{\gamma_0}\right) \left(1 + \frac{L(\pi_0)}{\gamma_0}\left(1 + \frac{L(\pi_0)}{2\gamma_0}\right)\right) \\ B_{rs} &\leftarrow \beta B_{rs} + I(\{r,s\} \in \pi_0) \exp\left(-\frac{L(\pi_0)}{\gamma_0}\right) \left(L(\pi_0) + \frac{L^2(\pi_0)}{\gamma_0}\right) \\ C_{rs} &\leftarrow \beta C_{rs} + I(\{r,s\} \in \pi_0) \exp\left(-\frac{L(\pi_0)}{\gamma_0}\right) \frac{L^2(\pi_0)}{2} \end{aligned} \tag{15}$$

The initial values of (15) are $A_{rs} = B_{rs} = C_{rs} = 0$.

The next agent arriving at node r will according to the unconditional probability of (12) depart towards node s, where the "pheromones" T_{rs} are determined according to (14) and updated according to (15) in its return. This is detailed in section 4.3.

4.2 Initialization and Selection Strategies

An initialization phase is needed to establish a rough estimate of the temperature γ under the initial routing probabilities. These probabilities are chosen to be uniformly distributed, $\mathbf{p}_u$, which is similar to [Rub99]. During this phase, the parameters of the autoregressive temperature computations in each node, i.e. (11), are obtained as well as initial values of the pheromone parameters of (15). The number of agents completing a tour during the initialization is $D \sim n \cdot m$ where n is the number of nodes in the network and m is the average number of outgoing links per node. The convergence of the algorithm is robust with respect to the initial routing probabilities and number of agents.

The actual next hop probability, $\mathbf{q}_t$, the agents use must take into account the previously visited nodes. Both during the initialisation phase and the rest of the search process our agents use the following selection strategy of the next node:

$$X_{t,r}(s) = \begin{cases} 1, & \text{if node } s \text{ has not already been visited} \\ 1, & \text{if all nodes have been visited and } s = homenode \\ 0, & \text{otherwise} \end{cases}$$

In networks that are not fully connected, an agent may experience that $\sum_{\forall s} X_{t,r(s)} = 0$, i.e. it is stuck. In this case the agent is terminated.

After the initialization phase there will be a non-zero probability that $X_{t,r(s)}$ may cause the vector $\mathbf{q}_{t,r}$ to be zero, i.e. all feasible routes are found to be inferior. When such a no-next-hop event occurs $\mathbf{p}_t$ is replaced by $\mathbf{p}_u$. By introducing a small noise component ϵ as shown in (16), $\mathbf{p}_u$ is generated both during the γ-initialisation phase and when a no-next-hop events occur.

$$q_{t,rs} = \frac{[I(t > D)p_{t,rs}(1-\epsilon) + \epsilon]\, X_{t,r}(s)}{\sum_{\forall k} [I(t > D)p_{t,rk}(1-\epsilon) + \epsilon]\, X_{t,r}(k)} \quad \text{when} \sum_{\forall s} X_{t,r}(s) > 0 \qquad (16)$$

where $I(\ldots)$ is the indicator function and ϵ is chosen very small, e.g. 10^{-60}.

The parameter ρ in (4), (9), etc. governs the emphasis put on the shorter routes. In [Rub99] it is proposed to introduce an adaptive ρ, i.e. ρ is decreased during the search process, which resulted in a slightly faster convergence. Our experiments show that our mobile agent algorithm converges significantly faster if ρ is decreased by 5% when no improvement in minimum path cost has been observed after D tours. Decreasing ρ by a higher factor did not improve the convergence significantly but a lower factor reduced the speed-up notably. Hence, each agent home node performs the operation

$$\rho \leftarrow 0.95 \cdot \rho,\ l \leftarrow i \quad \text{when } (i - l = \lceil D/k \rceil) \wedge (\min_{j \le l}(L(\pi_j)) = \min_{j \le i}(L(\pi_j))) \qquad (17)$$

where k is the number of agent home nodes in the network.

4.3 Agent Behaviour Algorithm

Fig. 2 shows pseudo-code describing the behaviour of a mobile agent implementing our algorithm. Each node in the network is assumed to store the autoregressive parameters required by (15), its own address (*current_node.address*) and a minimum cost observed (*current_node.L_min*). The address is set when the network topology is created. The minimum cost is updated by agents visiting the node (as described

```
$\rho$ := 0.01;
M = 0;                                              /* No of competed tours */
min_L = $\infty$ ;                                  /* Minimum tour cost known to agent */
no_min_L_change := 0;                               /* No of tours since min_L changed */
home_node := current_node.address;
do /* Main loop */
    visitlist := {};                                /* Initalize list of nodes visited */
    start_time := current_time;
    do /* Forward search */
        push(visitlist, current_node.address);      /* Added current node to visit list */
        min_L := Min(min_L, current_node.min_L);    /* Update agent's min. tour cost */
        current_node.min_L := min_L;                /* Update node's min. tour cost */
        r := current_node.address;
        Q(s): = Accummulate( $q_{t,rs}$ )           /* Create accum. prob.dist. of (16) */
        X := UniformDistribution(0.0, 1.0);         /* Generate uni. dist. random number */
        foreach s in neigbor_nodes                  /* Select next node to visit */
            if ( X < Q(s) )
                move_to(s)
                break;                              /* Exit foreach loop */
            end if
        end foreach
        if (Q(s) = 0 for all s in neigbor_nodes)    /* Terminate if a dead end is reached */
            terminate;
        end if
    until (current_node.address = home_node)
    if Not_changed?(min_L)                          /* Count no of tours without observing */
        no_new_min_L++                              /* change in L_min */
    else
        no_new_min_L := 0
    end if
    if no_new_min_L_counter > D                     /* Decrease $\rho$ if lack of change in */
        $\rho$ := $\rho$ * 0.95;                    /* min_L exceeds limit (17) */
        new_min_L_counter = 0;
    end if
    $L(\pi_0)$ := current_time – start_time;        /* Calculate cost of last tour */
    current_node.Update_Temp( $L(\pi_0)$ , $\rho$ ); /* Equation (11) */
    $\gamma_0$ := current_node. $\gamma_0$ ;        /* Get and carry temp.from home node */
    while (visitlist not empty) /* Backtracking */
        s := current_node;
        move_to( pop(visitlist));                       /* Remove last node in list and move to it */
        r := current_node;
        current_node.Update_Probabilities(r,s, $L(\pi_0)$ , $\gamma_0$ , $\rho$ );/* Equation (14) and (15) */
    done
    M++;                                            /* Increase counter of completed tours */
until (simulation is terminated)
```

Figure 2. Mobile agnet pseudo code.

in Fig. 2) and is later used to trigger adjustments of the search focus parameter ρ according to (17).

Each node acting as a home node must in addition to the parameters required by (15) store autoregressive parameters required by (11).

No synchronisation between agents takes place during a search scenario. Only indirect communication is performed by accessing path quality marks ($\mathbf{p}_t$) and the propagated minimum cost value.

Each agent starts every search for a path from its home node. Agents with the same home node cooperate in adjusting the temperature stored in the node. Thus a range of search scenarios are possible where one extreme is having all agents share the same home node and another extreme is letting each agent have its private home node. In section 6 we examine simulation results from both extremes.

5. Implementation in the Network Simulator

Due to the stochastic nature of our mobile agents it is difficult to predict the exact behaviour demonstrated. The behaviour of a single agent is to some extent trackable but when a number of agents are executing concurrently and asynchronously the overall system behaviour becomes too complex for formal analysis which leaves us with the option of collecting results using Monte Carlo simulations.

Instead of designing and implementing a complete simulator with configurable environmental parameters we chose to enhance an already well tested open source simulator package, the Network Simulator (NS) [DAR]. NS is capable of running realistic simulation scenarios of traffic patterns in IP-based networks. Dynamic topologies both wireline and wireless, miscellaneous protocols and a collections of traffic generators are supported. The package is implemented as a mix of OTcl and C++ classes.

We have made the NS-package capable of handling mobile code simulations by adding functionality for Active Networking (AN) [WH00, Pso99]. The extension is based on work done in the PANAMA project (TASC and the University of Massachusetts).

Fig. 3 illustrates how some of the environmental objects and a mobile agent interact during a simulation. A Tcl simulation control object creates node and mobile agent kernel objects (C++). The new kernel objects are controlled through Tcl-mirror objects but to avoid unnecessary overhead during simulations only infrequent operations (e.g. initialisation) are executed though this interface. The numbered message sequence illustrates how a mobile agent is transferred between two active network enabled nodes. For performance reasons only references (and size info) are passed between the nodes.

6. Case studies

We selected four different topologies from TSPLIB [Rei01] to demonstrated the performance of our algorithm. Three of the topologies where selected specifically

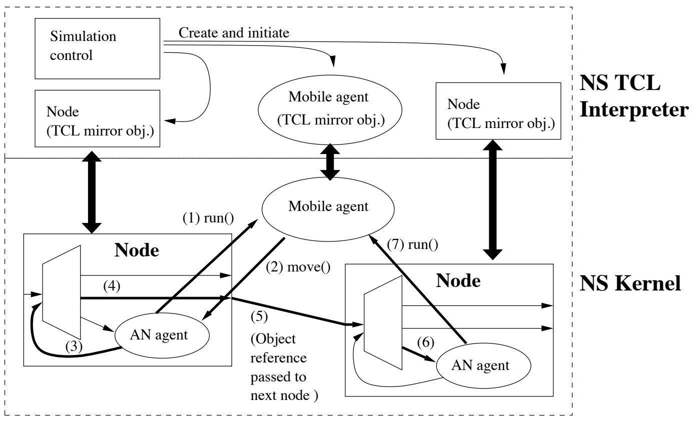


Figure 3. Schematic representation of interactions between objects in the NS simulator.

such that a comparisons between our algorithm, Rubinsteins algorithms and the Ant Colony System could be performed. Table 1 shows the results.

Topology	**No of nodes**	**No of agents**	**No of tours**	**Best tour**	**Converged average**	**Rubin–stein total no of samp–les**	**Rubin–stein best tour**	**ACS best tour**	**Best know n (TSP–LIB)**
fri26	26	1	133 267 (± 3015)	960 (1047)	1010 (± 18)	–	–	–	937
fri26	26	26	358 564 (± 8481)	940 (994)	970 (± 20)				
fri26*	26	26	149 453 (± 5562)	943 (1077)	1022 (± 34)				
ry48p	48	1	308 401 (± 12101)	15 169 (16210)	15 725 (± 340)	172 800	15509	14422	14422
ry48p	48	48	947 063 (± 16058)	14 618 (15369)	15 139 (± 212)				
ft53p	53	1	311 555 (± 9436)	7 351 (8069)	7 702 (± 249)	238 765	7111	–	6905
ft53p	53	53	1 270 621 (± 20147)	7 122 (7641)	7 487 (± 209)				
kro124p	100	1	579 618 (± 13424)	40 807 (45014)	43 128 (± 1167)	1120000	39712	36230	36230
kro124p	100	100	3 359 288 (± 69817)	38 352 (40322)	40 095 (± 2054)				

Table 1. Lists results from nine different simulation scenarios. By default all agents in our algorithm has different home nodes. Scenarios marked with * in the left column are exceptions where all agents have the same home node.

Column 2-6 (counting from the left) show parameter settings and performance results from our distributed algorithm. Column 1 and 2 give the name of the topology used in the scenario and the number of nodes of the topology. Column 3 shows the number of agents and autonomous home nodes applied in parallel during simulation.

Column 4 shows the total number of tours traversed before all agents converged towards a tour with the cost given in column 6. Column 5 shows the best tour found. Column 4 and 6 are averaged over 12 simulations with standard deviation shown in brackets while column 5 is the best of the best values found among 12 simulations with the worst of the best in brackets.

Column 7-9 show results obtained by two centralized algorithms, Rubinsteins original algorithm and the Ant Colony System version Opt-3. The last column shows the best known results listed in TSPLIB.

Empirically we found that the following parameter settings gave good results: $\beta = 0.998$, $\rho = 0.01$ and ρ reduction factor $= 0.95$. Thus they have been applied in all the simulation scenarios.

In general our algorithm finds good solution to the TSP scenarios tested, close to (and in a few occasion better than) the results reported by Rubinstein. But speed of convergence is not equally good. Our algorithm requires up to 5 times more tours to be traversed before convergence compared to the total number of samples in Rubinsteins algorithm. (Rubinstein stops his algorithm when the best tour found has not improved in a certain number of iteration. We report the number of tours required for all agents to converge towards one path. Results can still be compared since best tours for our algorithm are in general found only a short while before full convergence.) Small standard deviation values indicate stable convergence over several simulation runs.

When comparing results from scenarios run on the same topology but with a different number of agents we observe that our algorithm requires a higher number of tours for scenarios where multiple agents are searching in parallel than for single agent scenarios. Still the number of tours per agent is significantly less for the multi -agent cases, close to 10 time less for fri26, 20 times less for ry48p, 15 times less for ft53p, and 15 times less for kro124p. In a true concurrent environment this would result in the respective real time performance gains.

In the scenario named fri26* 26 agents search in parallel and share the same home node, i.e. they share the same set of autoregressive parameters required by (11). They use approx. the same total number of tours to converge as in the single agent version of fri26. Thus real time performance is improved by a factor equal to the number of agents. However the converged average is higher than for the other fri26 scenarios, i.e. premature convergence is more common. Having only a single home node also introduces a single point of failure which contradicts with our objective of a dependable distributed system.

The ACS-opt3 algorithm is implemented as a complex mix of iterations using heuristics for local optimization and iterations using global optimization by pheromone trails. Thus it is difficult to compare performance results by other means than best tour found and CPU time required. Our simulator is not implemented with the objective of solving TSPs as fast as possible. Thus CPU time is no good performance measure which leaves best tour as the only comparable result. The ACS-opt3 algorithm finds better best tours than both our algorithm and Rubinstein's.

7. Concluding remarks

In this paper we have introduced an algorithm for solving routing problems in communication networks. The algorithm is fully distributed and well suited for implementation by use of simple autonomous mobile agents encapsulated in for instance active network packets. Agents act asynchronously and independently and communicate with each other only indirectly using path quality markings (pheromone trails) and one shared search control parameters.

In contrast to other "ant-inspired" distributed stochastic routing algorithms, our algorithm has a mathematical foundation inherited from Reuven Rubinstein's cross-entropy method for combinatorial optimization [Rub99]. Rubinstein proposes an efficient search algorithm using Kullback-Leibler cross-entropy, important sampling, Markov chains and the Boltzmann distribution. However his algorithm is centralized and batch oriented. By introducing autoregressive stochastic counterparts to Rubinstein's method of shifting routing probabilities, we have removed the need of centralized control. In addition, due to the necessary approximations made, we have reduce the computational load of handling an agent in a node to a few simple arithmetic operations.

Performance wise the new algorithm shows good results when tested on a hard (NP-complete) routing problem, the Travelling Salesman Problem. Compared to Rubinstein's algorithm, up to 5 times more paths need to be tested before convergence towards a near optimal path takes place. Increasing the number of agents searching in parallel decrease significantly the number of tours per agent required to find a high quality path.

No excessive parameter tuning has so far been performed. Further investigation is required specially on the effect of adjusting the weight put on historical information (β) during the search process. Pros and cons of making more (or less) global knowledge available (i.e. let more parameter values propagated throughout the network) should also be looked into.

Currently new versions of our algorithm is under development where heuristic techniques found in algorithms like the Ant Colony System [DG97] are incorporate to improve performance. Additionally, by altering the search strategies, we expect our algorithm to find optimal tours when network topologies are far from fully meshed (as it is for TSPs) and do not allow all nodes to be visited only once.

Other ongoing work includes having several species of agents compete in finding quality paths in a network. Early results indicate that a set of disjunct high quality paths can be found efficiently. We intend to investigate the applicability of such a system to the routing problems encountered by Grover in his work on restorable network and protection cycles [GS98].

References

[Bal95a] Michael O. Ball. *Handbooks in Operation Research and Management Science, Network Models*, volume 7. North Holland, 1995.

[Bal95b] Michael O. Ball. *Handbooks in Operation Research and Management Science, Network Routing*, volume 8. North Holland, 1995.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[DG97] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computing*, 1(1), April 1997.

[Glo96] F. Glover. *Tabu Search*. Kluwer, 1996.

[Gol98] D. Goldberg. *Genetic Algorithms in Search, Optimization and MachineLearn ing*. Addison Wesley, 1998.

[GS98] W.D. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 537 –543, 7-11 June 1998.

[Hei95] Philip Heidelberger. Fast simulation of rare events in queueing and reliabili ty models. *ACM Transactions on modelling and Computer Simulation*, 5(1):43–85, January 1995.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science 220*, pages 671–680, 1983.

[Pso99] Konstantinos Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, First Quarter, 1999.

[Rei01] G. Reinelt. TSPLIB. Institut für Angewandte Mathematik, Universität Heidelberg, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, 2001.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[Rub01] Reuven Y. Rubinstein. *Stochastic Optimization: Algorithms and Applications*, chapter Combinatorial Optimization, Cross-Entropy, Ants and Rare Events - Section 7: Noisy Networks. Kluwer Academic Publishers, 2001.

[Sch00] J. Schuringa. Packet Routing with Genetically Programmed Mobile Agents. In *Proceedings of SmartNet 2000*, Wienna, September 2000.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[WH00] Otto Wittner and Bjarne E. Helvik. Simulating mobile agent based network management using network simulator. Poster in Forth International Symposium on Mobile Agent System (ASA/MA 2000), September 2000.

[WPO98] T. White, B. Pagurek, and Franz Oppacher. Connection Management using Adaptive Mobile Agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDAPTA'98)*, 1998.

PAPER B

Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks

Otto Wittner and Bjarne E. Helvik

Congress on Evolutionary Computation, CEC 2002

May 12-17th 2002, Honolulu, Hawaii.

CROSS ENTROPY GUIDED ANT-LIKE AGENTS FINDING DEPENDABLE PRIMARY/BACKUP PATH PATTERNS IN NETWORKS

Otto Wittner

Bjarne E. Helvik

Abstract Telecommunication network owners and operators have for half a century been well aware of the potential loss of revenue if a major trunk is damaged, thus dependability at high cost has been implemented. A simple, effective and common dependability scheme is 1:1 protection with 100% capacity redundancy in the network. A growing number of applications in need of dependable connections with specific requirements to bandwidth and delay have started using the internet (which only provides best effort transport) as their base communication service. In this paper we adopt the 1:1 protection scheme and incorporate it as part of a routing system applicable for internet infrastructures. 100% capacity redundancy is no longer required. A distributed stochastic path finding (routing) algorithm based on Rubinstein's Cross Entropy method for combinatorial optimisation is presented. Early results from Monte Carlo simulations indeed indicate that the algorithm is capable of finding pairs of independent primary and backup paths satisfying specific bandwidth constraints.

Keywords: Dependable Routing, Optimal Paths, Cross Entropy Optimisation, Mobile Agents, Swarm Intelligence, Network Management

1. Introduction

Telecommunication network owners and operators have for half a century been well aware of the potential loss of revenue and reputation if a trunk is damaged and a large number of subscribers are disconnected for a longer period of time. Dependability at high cost has been implemented by building physical redundancy in network topologies. A common, simple and effective but expensive scheme is 1:1 protection with 100% redundancy where a primary link always has an independent backup link standing by ready for use if a failure should occur.

The internet has since its days of design three decades ago been in operation providing a best effort communication service with no guarantees for loss of connectivity between users [Pos81]. So far many applications have found this level of QoS sufficient, but recently a growing number of applications in need of more dependable connections with specific requirements to bandwidth and delay have started using the internet as their base communication service. This trend calls for new ideas and solutions in the domain of routing, i.e. path fining in networks. In this paper we adopt the 1:1 protection scheme and incorporate it as part of a routing system such that a high level of dependability is achieved without the need of 100% redundancy in the network.

Finding paths in networks is generally considered to be hard problems even NP-complete [GJ79] in several cases (e.g. Travelling Salesman, Hamiltonian Path, Longest Path). Finding an optimal pattern of primary and backup paths between a set of source and destination nodes such that bandwidth, delay and link independence requirements are satisfied has similarities with proven NP-complete problems like “Path with Forbidden Pairs”, “Disjoint Connecting Paths” and “Shortest Weight-Constrained Path” (54, 62 and 63 are the respective problem classification codes in [GJ79]). In this paper we introduce a distributed stochastic optimisation algorithm capable of finding near optimal patterns of primary and backup paths. The agents in our algorithm have a behaviour comparable to the navigational behaviour of ants, but the overall algorithm inherits its mathematical foundation from Rubinstein’s cross entropy method for combinatorial optimisation [Rub99].

Section 2 describes the problem at hand in more detail. Section 3 introduces Rubinstein’s cross entropy method. Section 4 describes the foundations for the behaviour of the agents in our algorithm, and Section 5 continues with implementation details. Section 6 presents results from Monte Carlo simulations and finally Section 7 summarieses, concludes and indicates future work.

2. Problem Description

2.1 Motivation

A common strategy to improve dependability in any system is to introduce redundancy among the system components. 100% redundancy implies duplicating all components such that both a primary set and a backup set exist. While a primary component is active and fault free, the peer backup component can be configured to operate in different ways. *Modular redundancy* implies having the backup component running in parallel with the primary, e.g. duplicated links in a network transporting the same copy of information synchronously (1+1 protection). *Standby redundancy* implies activating a backup component only when the primary fails, e.g. a network dimensioned to have spare capacity enough on backup links to be able to reroute all traffic if primary links should fail (1:1 protection).

Modular redundancy is commonly implemented at link level in communication networks where fast fault recovery is important. Redundant links can not be use for low priority traffic in such a configuration since a backup link is always in use.

Standby redundancy is common at higher levels in communication networks, e.g. at transport level in state or country back bone networks. In an SDH protection ring high priority traffic is only transported in one direction while the ring is free of faults, and rerouted in the opposite direction when a link failure occurs (see e.g. [De95]).

Low priority traffic can thus utilise the backup/spare capacity in the opposite direction when all links in the ring are operational.

Virtual paths (VP) in ATM networks can be applied to construct patterns of primary and backup paths in a meshed network (see e.g. [PD00, FHW96]) . On the contrary to SDH rings which implement 100% capacity redundancy due to the ring structure, primary/backup patterns in meshed networks can implement link failure protection for all primary paths without requiring 100% capacity redundancy simply by letting selected primary paths share a backup path. The selected primary paths must be independent, i.e. have low probability of failing simultaneously.

As with SDH, ATM typically runs in back bone networks (city, state, country) where necessary primary/backup path patterns are configured by the network operator. Several optimisation algorithms for network planning have been developed (e.g. [MGR97]) which produce near optimal configurations as long as all network parameters (from all nodes and links) are available for analysis, i.e. the algorithms are centralised. A network operator may have enough overview and control to apply such algorithms.

The Internet Protocol (IP) has become a common protocol both in backbone and end user networks. By enabling MPLS [RVC01] in IP-networks labelled switched paths (LSPs) can be established (similar to VPs in ATM), which again enables configuration of primary/backup path patterns for paths between end user. Optimal pattern configuration can provide end users with a better tailored QoS than the best effort service provided in a standard IP-network today. Finding such an optimal pattern configuration on the other hand, is hard when the number of primary paths vary constantly (end users establish and release connections frequently) and since no central control unit with total overview exits in an IP-network, i.e. centralised optimisation algorithms cannot be applied. In the next section we introduce an algorithm for finding primary/backup path patterns. The algorithm is distributed, produces near optimal patterns and is potentially able to handle a dynamic environment, i.e. produce re-optimised patterns when new primary paths are establish and old released.

2.2 Primary/Backup Path Patterns

A meshed communication network can be represented as a directed graph G with a set of links L_G as edges and a set of nodes N_G as vertices . Let the pair $\{i, j\} \in L_G$, abbreviated ij, represent a link from node i to node j. (Pairs in general are abbreviated in the same manner in the rest of the paper). Let c_{ij} be the available capacity of link ij.

Let $\pi_r^{m(sd)} = \{si, ij, \ldots, kd\}$ be a path of rank $r \in \mathcal{R}$ providing connection $m \in \mathcal{M}$ from source node s to a destination node d via a specific set of links (s and d are implicit in m and will be omitted). $\mathcal{M}$ is the set of all connections and $\mathcal{R}$ the set of all ranks. A primary path has by definition a higher rank than its backup path, i.e. $r_{primary} < r_{backup}$. In this paper $\mathcal{R} = \{0, 1\}$, thus only two rank levels for paths are considered, i.e. $r_{primary} = 0$ and $r_{backup} = 1$.

Let $\Pi = \{\pi_r^m\}_{\forall mr}$ i.e. the set of all paths, and let

$$\Pi^\dagger = \{\Pi_x : \, \Pi_x \in \mathcal{P}(\Pi) \wedge (\{\pi_0^m, \pi_1^m\} \nsubseteq \Pi_x)_{\forall m}\}$$

i.e. a set containing subsets of every possible combination ($\mathcal{P}(\ldots)$ is the power set) of primary/backup path patterns except subsets where both the primary and backup paths for the same connection m is present. $\Pi^\dagger$ describes all the possible ways the set of connections $\mathcal{M}$ can be established such that only one of the paths in a primary/backup path pair are operational at a time.

Finally let a_m be the required capacity of connection m. An object function for our primary/backup path pattern optimisation problem can now be formulated as

$$\min_{\Pi} \left[\sum_{\Pi_l \in \Pi^\dagger} \sum_{ij \in L_G} \left[\sum_{mr \in \mathcal{X}_{ij}^l} a_m - c_{ij} \right]^+ \right] \tag{1}$$

where

$$\mathcal{X}_{ij}^l = \{\forall mr : \, \pi_r^m \in \Pi_l, \, ij \in \pi_r^m\}$$

i.e. we want to find a primary/backup path pattern Π which minimises the sum of link overload for all valid combinations of operational primary and backup paths. $[x]^+$ returns x when $x > 0$ otherwise 0. The optimisation is subject to the following constraints:

$$\sum_{ij \in L_G} \left(c_{ij} - \sum_{\forall m:\, ij \in \pi_0^m} a_m \right) \geq 0 \tag{2}$$

$$\sum_{\forall m} |\pi_0^m \cap \pi_1^m| = 0 \tag{3}$$

i.e. all primary paths must be assigned the capacity they require (2) and all links in primary and backup paths of the same connection must be disjoint (3).

3. The Cross Entropy Method

Our algorithm in this paper is based on a previously publish algorithm [HW01] which again is founded in Rubinstein's Cross Entropy method for combinatorial optimisation [Rub99]. Both [HW01] and [Rub99] focus on the search for optimal Hamiltonian cycles, known as the Travelling Salesmen Problem, however the algorithm may be used to find any optimal path with an additive cost function. The algorithms use a transition probability matrix to generate sample paths. The matrix provides probabilities for transitions (movement) from a specific node to its neighbour nodes. The event of finding an optimal path by doing a random walk (without revisiting nodes) based on the matrix is rare, and rare event theory may be applied to find such paths.

In [Rub99] Rubinstein designs an algorithm which by importance sampling in multiple iterations alters the transition matrix and amplifies probabilities in Markov chains producing near optimal Hamiltonian paths. Cross Entropy is applied to ensure optimal alteration of the matrix. To speed up the process, a performance function weights the path qualities such that high quality paths have greater influence on the alteration of the matrix. The algorithm has 4 steps

1 At iteration $t = 0$ choose an initial transition matrix $P_{t=0}$ (e.g. uniformly distributed).

2 Generate N paths from P_t. Calculate the minimum Boltzmann temperature γ_t

$$\min \gamma_t \text{ s.t. } h(P_t, \gamma_t) = \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) > \rho \tag{4}$$

where $H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}}$ is the performance function returning the quality of path π_k. $L(\pi_k)$ is the raw cost of path π_k (e.g. delay in telecommunication network). $10^{-6} \leq \rho \leq 10^{-2}$ is a search focus parameter. The minimum solution for γ_t will result in a certain amplification (controlled by ρ) of high quality paths and a minimum average $h(P_t, \gamma_t) > \rho$ of all path qualities in the current batch of N paths.

3 Using γ_t from step 2, and $H(\pi_k, \gamma_t)$ for $k = 1, 2..., N$ generate a new transition matrix P_{t+1} by solving

$$\max_{P_{t+1}} \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) \sum_{ij \in \pi_k} \ln P_{t,ij} \tag{5}$$

where $P_{t,ij}$ is the transition probability from node i to j at iteration t. The solution to (5) is shown in [Rub99] to be

$$P_{t+1,rs} = \frac{\sum_{k=1}^{N} I(\{r,s\} \in \pi_k) H(\pi_k, \gamma_t)}{\sum_{l=1}^{N} I(\{r\} \in \pi_l) H(\pi_l, \gamma_t)} \tag{6}$$

which will minimise the cross entropy between P_t and P_{t+1} and ensure an optimal shift in probabilities with respect to γ_t and the performance function.

4 Repeat steps 2-3 until $H(\widehat{\pi}, \gamma_t) \approx H(\widehat{\pi}, \gamma_{t+1})$ where $\widehat{\pi}$ is the best path found.

4. Agent Behaviour

4.1 Distributed Implementation of the Cross Entropy Method

In [HW01] (6) and (4) are replaced by the autoregressive counterparts

$$P_{t+1,rs} = \frac{\sum_{k=1}^{N} I(\{r,s\} \in \pi_k) \beta^{N-k} H(\pi_k, \gamma_t)}{\sum_{l=1}^{N} I(\{r\} \in \pi_l) \beta^{N-k} H(\pi_l, \gamma_t)} \tag{7}$$

and

$$\min \gamma_t \text{ s.t. } h_t^{'}(\gamma_t) > \rho \tag{8}$$

respectively where

$$\begin{aligned} h_t^{'}(\gamma_t) &= h_{t-1}^{'}(\gamma_t)\beta + (1-\beta)H(\pi_N, \gamma_t) \\ &\approx \frac{1-\beta}{1-\beta^N}\sum_{k=1}^{N}\beta^{N-k}H(\pi_k, \gamma_t) \end{aligned}$$

and $\beta < 1$. Hence whenever a single new path π_N is found step 2 and 3 can immediately be performed and a new probability matrix generated.

The overall algorithm can now be viewed as a distributed search system where search agents can evaluate a path found (and calculate γ_t by (8)) when they reach their destination node, and immediately start backtracking the path towards the source while updating relevant probabilities in the transition matrix by applying $H(\pi_N, \gamma_t)$ through (7). Such a search system has great similarities to how ant colonies explore their surrounding environment searching for food. Markov chains in the transition matrix can be compared to pheromone trails built by ants searching for food. Step 2 is comparable to an ant's forward search for a path to a food source and the ant realising the quality of the path found (finding γ_t). Step 3 represents an ant's adjustment of the pheromone level of the trail when returning to its nest. In the remaining of this paper we interchangeably call the components which execute the steps of the distribute algorithm for mobile agents, ant-like agents or just agents .

Several other algorithms directly inspired by the ant foraging analogy exist ([CD98] [WPO98][VS99a][DC99][Sch00]) with Schroonderwoerd et al as one of the pioneers [SHBR97]. However they lack the mathematical foundation of the algorithm in [HW01]. Problem solving inspired by ants and other swarming creatures is known generally as "Swarm Intelligence" [BDT99].

4.2 Agents Detesting each other

Our algorithm in this paper adopts all the fundamentals from [HW01] but implements a different cost function. The approach is to let each primary path and each back-up path be dealt with by a separate species of ant-like agents. Our fundamental idea is to let the different species detest each other in accordance to the primary/backup optimisation criteria in Section 2.2.

- Backup agents search for paths which are disjoint with their corresponding primary paths.
- Backup agents having overlapping corresponding primary paths search for disjoint paths.
- All agents detest other agents which represent a load which in addition to their own may incur an overload

Hence, the following additive cost function for a path is established

$$L(\pi_r^m) = \sum_{ij \in \pi_r^m} \mathcal{D}_{ij}^{mr}$$

where the link cost $\mathcal{D}$ represents how desirable the link is considering the primary/backup path optimisation criteria in Section 2.2.

$$\mathcal{D}_{ij}^{mr} = \mathcal{S}\left[a_m + \sum_{\forall ns:\, ij \in \pi_s^n} P_{ij}^{ns}\, V_i^{ns}\, \mathcal{Q}_{mr}^{ns}\, a_n - c_{ij} \right] \tag{9}$$

where

$$\mathcal{Q}_{mr}^{ns} = \begin{cases} \widehat{Q}_{m,r-1}^{n,s-1}, & n \neq m,\ s = r = 1 \\ 1, & n = m,\ s < r \\ 0.75, & n \neq m,\ s < r \\ 0, & otherwise \end{cases} \tag{10}$$

$$\mathcal{S}[c] = \begin{cases} \eta \cdot e^{\frac{c}{e \cdot \eta}}, & c < \eta \cdot e \\ c, & otherwise \end{cases} \tag{11}$$

We consider agents involved in building a path π_r^m as agents of species m with rank r (species is synonymous with connection). The components of (9)-(11) can be explained as follows:

- a_m is the capacity required by connection m which the path in question (π_r^m) is intended to carry.
- a_n is the capacity required by a competing path (π_s^n).
- The summation includes paths of all species (connections) and all ranks which have previously followed link ij (and placed pheromones).
- P_{ij}^{ns} is the probability that agent ns will choose link ij as its departure link from node i given that the agent visits node i.
- V_i^{ns} is the probability that agent ns visits node i during forward search. Thus $P_{ij}^{ns} \cdot V_i^{ns}$ weights the required capacity a_n of competing paths such that agent mr is discouraged to follow links where overload in general is likely.
- $\mathcal{Q}_{mr}^{ns}$ weights the capacity a_n even further:
  - When π_s^n is a backup path of a different connection, $\widehat{Q}_{m,r-1}^{n,s-1}$ is returned. $\widehat{Q}_{m,r}^{n,s}$ is the approximate probability of a common link failure in both π_s^n and π_r^m, hence $\widehat{Q}_{m,r-1}^{n,s-1}$ is the probability of a common link failure in both π_s^n's and π_r^m's primary paths. The return value weights the required capacity a_n of competing paths such that agent mr is discouraged to follow links where overload is likely if a link-failure should occur.
  - When π_s^n is π_r^m's primary path, a constant of 1 is returned such that agent mr is discouraged to follow links used by its primary path. This enforces the constraint in (3).

- When π_s^n is a primary path of a different connection, a constant of 0.75 is returned such that agent mr to some degree is discouraged to follow links used by alien primary paths. The two constants 1 and 0.75 are empirically chosen, and meant to increase the probability of backup paths being routed on links together with alien primary paths rather than being routed on links together with their own primary path, i.e. avoid violating the constraint in (3).
- When $\pi_r^m = \pi_s^n$, a constant 0 is returned. The required capacity of connection m is represented outside the summation.

- $\mathcal{S}[\ldots]$ is applied to smoothen the output of the cost expression and to ensure none-zero link costs. Parameter η regulates the smoothness of the transition area between no overload and overload. See fig. 1.

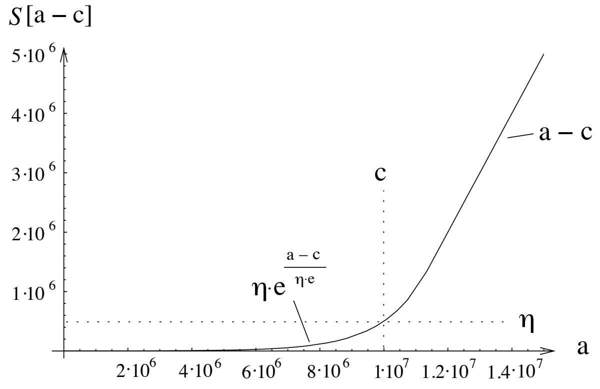


Figure 1. Cost shaping as a result of (11). The plot shows $\mathcal{S}[a - c]$ where $c = 10^7$ and $\eta = 5 \cdot 10^5$.

$\mathcal{D}_{ij}^{mr}$ returns a high cost (the expected overload) when link ij has too many candidate paths running through it which together could overload the link. A cost close to zero is returned when link ij has some or a lot of unrequested spare capacity.

In summary , the cost $L(\pi_r^m)$ of a path is increasing if an ant-like agent "smells" high levels of pheromone (observes high probabilities in the transition matrix) from agents potentially competing for the same resources along the links of the path.

5. Implementation

To verify the performance of our primary/backup path algorithm a simulator was implemented based on an Active Network (AN) [Pso99] enabled version of "Network Simulator 2", an open source simulator package capable of simulating realistic IP-based scenarios [DAR]. The AN extension makes it convenient to implement simulation scenarios where multiple mobile ant-like agents solve problems by exploring a network topology.

The implementation of an agent's behaviour is fairly straight forward. An agent in our algorithm roughly performs the following steps after being "born" at its source node:

1 Clear tabulist containing nodes already visited. Fetch a vector form node's database (DB) of (approximate) probabilities for common link failures between the agent species' primary path and alien primary paths, i.e. $\left\{\widehat{Q}^{n,0}_{m,0}\right\}_{\forall n}$.

2 Record visit to current node i in tabulist. If current node is the destination node:

   (a) Fetch agent species/rank's temperature (γ_k) from destination node DB. Calculate new temperature based on accumulated path cost $L(\pi^m_r)$. Store new temperature.

   (b) Update total no of visited nodes by agent's species/rank stored in node DB. Fetch new total.

   (c) Clear vector with probabilities of common failures.

   (d) Backtrack every hop towards sources node. At each hop:

      i Update the transition matrix (leave pheromones) by use of (7)

      ii Accumulate approximate common link failure probabilities between agent species/rank's and other species of same rank by fetching relevant V^{nr}_i and P^{nr}_{ij} from node DB and calculating $\widehat{Q}^{n,r}_{m,r}$. Store $\widehat{Q}^{n,r}_{m,r}$ in vector.

      iii Update V^{mr}_i for agent's species/rank by dividing total no of visits to current node (available from node DB) by total no of nodes visited by agent's species/rank (carried from destination node). Store new V^{mr}_i in node DB.

   (e) When source node has been reached, store vector of common link failure probabilities in node DB.

   (f) Goto step 1.

3 Build a next-hop probability table based on the transition matrix (pheromones) for the agent's species and the agent's rank as well as the tabulist (to avoid revisiting nodes).

4 Select next node j to visit using the next-hop probability table.

5 Calculate cost $\mathcal{D}^{mr}_{ij}$ of link ij towards next node. V^{ns}_i and P^{ns}_{ij} for relevant ns are fetched from node DB. $\widehat{Q}^{n,s-1}_{m,r-1}$ is available from vector carried by agent. Accumulate path cost $L(\pi^m_r)$.

6 Move to next node and goto step 2.

Minimum one agent of a specific species/rank is required for every path to be searched for. When the search converges, the path found by a species/rank of agents will appear as a chain of high values in the transition matrix (an intense track of pheromone).

6. Simulation Results

Figure 2 shows the topology of the example network we chose for our test scenario. All links in the topology are duplex with a capacity of 10 Mb/s.

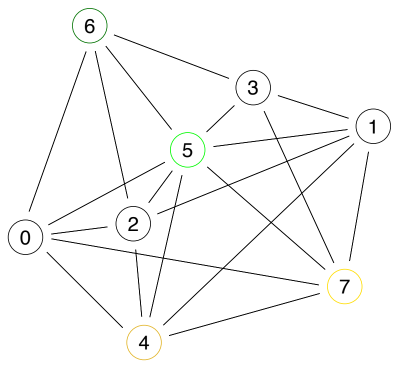


Figure 2. Network topology with 8 nodes and 21 links. All links are duplex with a capacity of 10 Mb/s.

All algorithm parameter values (see [HW01] for details) applied during the simulations were empirically selected: $\rho = 0.01$, $\beta = 0.998$, $noise = 10^{-60}$, $D = 100$, $\rho reduction = 1$ (i.e. no reduction) and $\eta = 5 \cdot 10^5$.

In our test scenario eight 5Mb/s connections between node 0 and 1 in the test network were requested, and a pattern of 8 independent primary paths and 8 backup paths was expected to be established. Such a pattern is realisable since the network provides resources for 10 5Mb/s paths from node 0 to node 1, i.e. 8 5Mb/s primary paths can be operational simultaneously and share the last 2 paths for backup purposes.

Table 1. Summarised results from 15 simulations.

Primary path causes overload	0 links out of 21 for all sim.
Primary/Backup path not disjoint	0 links out of 21 for all sim.
Single link failure handled by backup paths without overload	6 out of 15 sim.
Average (stdev) no of links which may cause overload due to suboptimal backup paths	9 out of 15 sim., 6.0 ($\pm$2.06) links out of 21
Loss given link failure when backup paths are suboptimal	Maximum 5 Mb/s

Table 1 shows summarised results from 15 simulation of our test scenario. None of the patterns of primary paths established caused any link to be overloaded, i.e. the constraint in (2) from Section 2.2. was satisfied in every simulation. None of the patterns established primary/backup path pairs with common links, i.e. the constraint in (3) was also satisfied in every simulation. In 6 out of the 15 simulations the backup path patterns established where able to handle a single link failure in any link of the network without overloading links, i.e. a good candidate Π for the object function (1) was found. For the remaining 9 simulations an average of 6.0 (with standard deviation 2.06) specific single link failures would cause an overload situation due to suboptimal patterns of backup paths. But even if such an overload situation should occur, no greater loss of traffic than 5 Mb/s (one connection) would be experienced. Hence even the primary/backup path patterns where the backup paths are suboptimal can be considered relatively good candidate Πs for the object function.

7. Conclusion

A common way of improving dependability in communication networks is by configuring pairs of independent primary and backup paths. A backup path is meant to carry primary traffic if the primary path should fail. In the internet today no such level of dependability is provided to end users. MPLS may be a tool to enable backup/primary path configurations in IP-networks.

In this paper we present a distribute algorithm capable of finding good primary/-backup path patterns in meshed networks subject to given link capacity constraints. The algorithm is suitable for IP-networks where no centralised control is desired. The algorithm is based on a previously publish algorithm [HW01] which again inherits its foundation from Rubinstein's work on cross entropy optimisation.

An advanced link cost function is introduced in the algorithm described. The cost function takes into consideration path conflicts and tries to minimise link overload and traffic loss both when all primary paths are free of faults and when link failures occur.

Simulation results indicate that the algorithm manages to establish close to optimal primary/backup path patterns. However parameter tuning and further testing are required.

Future work includes testing the algorithm in larger and more realistic network environments with dynamic traffic conditions, and investigating its performance as a routing system in ad hoc networks.

References

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[DC99] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137–172, 1999.

[De95] M. Decina and T. Plevyak (editors). Special Issue: Self-Healing Networks for SDH and ATM. *IEEE Communications Magazine*, 33(9), September 1995.

[FHW96] V. J. Friesen, J. J. Harms, and J. W. Wong. Resource Management with VPs in ATM Networks. *IEEE Network*, 10(5), Sep/Oct 1996.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[HW01] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[MGR97] M. H. MacGregor, W. D. Gover, and K. Ryhorchuk. Optimal spare capacity preconfiguration for faster restoration of mesh networks. *Journal of Network and System Management*, 5(2):159–170, Jun 1997.

[PD00] Larry L. Peterson and Bruce S. Davie. *Computer Networks - A Systems Approach*. Morgan Kaufmann, 2000.

[Pos81] Jon Postel. RFC 791: Internet Protocol. IETF, September 1981.

[Pso99] Konstantinos Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, First Quarter, 1999.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. RFC3031: Multiprotocol Label Switching Architecture. IEFT, January 2001.

[Sch00] J. Schuringa. Packet Routing with Genetically Programmed Mobile Agents. In *Proceedings of SmartNet 2000*, Wienna, September 2000.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[VS99] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[WPO98] T. White, B. Pagurek, and Franz Oppacher. Connection Management using Adaptive Mobile Agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDAPTA'98)*, 1998.

PAPER C

Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks

Otto Wittner and Bjarne E. Helvik

The Third International Workshop on Ant Algorithms, ANTS'2002

September 12-14th, Brussels, Belgium.

CROSS-ENTROPY GUIDED ANT-LIKE AGENTS FINDING CYCLIC PATHS IN SCARCELY MESHED NETWORKS

Otto Wittner

Bjarne E. Helvik

Abstract Finding paths in networks is a well exercised activity both in theory and practice but still remains a challenge when the search domain is a dynamic communication network environment with changing traffic patterns and network topology. To enforce dependability in such network environments new routing techniques are called upon. In this paper we describe a distributed algorithm capable of finding cyclic paths in scarcely meshed networks using ant-like agents. Cyclic paths are especially interesting in the context of protection switching, and scarce meshing is typical in real world telecommunication networks. Two new next-node-selection strategies for the ant-like agents are introduced to better handle low degrees of meshing. Performance results from Monte Carlo Simulations of systems implementing the strategies are presented indicating a promising behavior of the second strategy.

1. Introduction

Finding paths in networks is a well exercised activity both in theory and practice. Still it remains a challenge especially when the search domain is a dynamic communication network environment with changing traffic patterns and network topology. The internet is such an environment, and as an increasing number of applications demanding QoS guarantees is beginning to use internet as their major communication service, efficient and dependable routing in the network becomes more important than ever.

Protection switching [AP94] is a well known technique for improving dependability in communication networks and commonly used in larger SDH- and ATM-networks. To enable fast recovery from link or network element failures two (or more) disjunct independent paths from source to destination are defined, one primary and one (or more) backup path. Loss of connectivity in the primary path triggers switching of traffic to the backup path. Good dependability is achieved by allocating required resources for the backup path prior to the occurrence of failures in the primary path. However maintaining the necessary mesh of backup paths in a dynamic network with a large number of active sources and destinations is a complex task [WH02b]. Grover & al. [GS98, SG99] propose to use simple cyclic paths ("p-cycles") as a means for dependable routing in meshed networks. Protection rings are common in SDH based transport networks and guarantee protection against single link failures in the ring (assuming the ring has duplex links)[De95]. All network elements on the ring can continue to communicate with each other after a single link failure by routing all traffic over the "healthy" curve-section of the ring (Figure 1). Thus a cyclic path can provide a dependable communication service for a set of

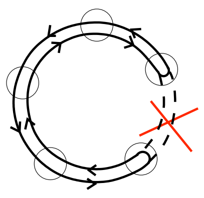


Figure 1. Protection switching in a ring network.

sources and destinations. Assuming p-cycles can be found, the number of necessary cycles to be maintained in a network providing a dependable communication service to a set of network elements, is likely to be far less than the number of traditional backup paths required to provide the same service.

In this paper we describe an algorithm, founded in rare event theory and cross entropy, that is able to find cyclic paths in networks. The fundamentals of the algorithm has previously be published in [HW01]. This paper enhances the original algorithm by enabling it to find cyclic paths in networks with low degrees of meshing, a common property of real world telecommunication networks. The algorithm is fully distributed with no centralized control, two desirable properties when dependability is concerned. The algorithm can conveniently be implemented using simple (ant-like) mobile agents [PK98]. Section 2 introduces the foundations of the original algorithm and motivates the use of it. Section 3 describes the original next-node-selection strategy for the mobile agents as well as two new strategies. Emphasis is put upon search performance when searching for cycles in scarcely meshed networks. Section 4 presents results from Monte Carlo simulations of systems based on the strategies described in section 3. Finally section 5 summarizes, concludes and indicates future work.

2. Agent Behavior Foundations

The concept of using multiple mobile agents with a behavior inspired by foraging ants to solve routing problems in telecommunication networks was introduced by Schoonderwoerd & al. in [SHBR97] and further developed in [CD98][WPO98] [Sch00]. Schoonderwoerd & al.'s work again builds on Dorigo & al.'s work on Ant Colony Optimization (ACO) [DC99]. The overall idea is to have a number of simple ant-like mobile agents search for paths between a given source and destination node. While moving from node to node in a network an agent leaves markings imitating the pheromone left by real ants during ant trail development. This results in nodes holding a distribution of pheromone markings pointing to their different neighbor nodes. An agent visiting a node uses the distribution of pheromone markings to select which node to visit next. A high number of markings pointing towards a node (high pheromone level) implies a high probability for an agent to continue its itinerary towards that node. Using trail marking agents together with a constant evaporation of all pheromone markings, Schoonderwoerd and Dorigo show that after a relatively short period of time the overall process converges towards having the majority of the agents following a single trail. The trail tends to be a near optimal path from the source to the destination.

2.1 The Cross Entropy Method

In [Rub99] Rubinstein develops a search algorithm with similarities to Ant Colony Optimization [DC99, ZBMD00]. The collection of pheromone markings is represented by a probability matrix and the agents' search for paths is a Markov Chain selection process generating sample paths in the network. ("Path" and "trail" are equivalent in this paper and will be used interchangeably.)

In a large network with a high number of feasible paths with different qualities, the event of finding an optimal path by doing a random walk (using a uniformly distributed probability matrix) is rare, i.e. the probability of finding the shortest Hamiltonian cyclic path (the Traveling Salesman Problem) in a 26 node network is $\frac{1}{25!} \approx 10^{-26}$. Thus Rubinstein develops his algorithm by founding it in rare event theory.

By importance sampling in multiple iterations Rubinstein alters the transition matrix and amplifies probabilities in Markov chains producing near optimal paths. Cross entropy (CE) is applied to ensure efficient alteration of the matrix. To speed up the process, a performance function weights the path qualities (two stage CE algorithm [Rubar]) such that high quality paths have greater influence on the alteration of the matrix. Rubinstein's CE algorithm has 4 steps:

1 At the first iteration $t = 0$, select a start transition matrix $P_{t=0}$ (e.g. uniformly distributed).

2 Generate N paths from P_t using some selection strategy (i.e. avoid revisiting nodes, see section 3). Calculate the minimum Boltzmann temperature γ_t to fulfill average path performance constraints, i.e.

$$\min \gamma_t \text{ s.t. } h(P_t, \gamma_t) = \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) > \rho \tag{1}$$

where $H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}}$ is the performance function returning the quality of path π_k. $L(\pi_k)$ is the raw cost of path π_k (e.g. delay in a telecommunication network). $10^{-6} \leq \rho \leq 10^{-2}$ is a search focus parameter. The minimum solution for γ_t will result in a certain amplification (controlled by ρ) of high quality paths and a minimum average $h(P_t, \gamma_t) > \rho$ of all path qualities in the current batch of N paths.

3 Using γ_t from step 2 and $H(\pi_k, \gamma_t)$ for $k = 1, 2..., N$, generate a new transition matrix P_{t+1} which maximizes the "closeness" to the optimal matrix, by solving

$$\max_{P_{t+1}} \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) \sum_{ij \in \pi_k} \ln P_{t,ij} \tag{2}$$

where $P_{t,ij}$ is the transition probability from node i to j at iteration t. The solution to (2) is shown in [Rub99] to be

$$P_{t+1,rs} = \frac{\sum_{k=1}^{N} I(\{r,s\} \in \pi_k) H(\pi_k, \gamma_t)}{\sum_{l=1}^{N} I(\{r\} \in \pi_l) H(\pi_l, \gamma_t)} \tag{3}$$

which will minimize the cross entropy between P_t and P_{t+1} and ensure an optimal shift in probabilities with respect to γ_t and the performance function.

4 Repeat steps 2-3 until $H(\widehat{\pi}, \gamma_t) \approx H(\widehat{\pi}, \gamma_{t+1})$ where $\widehat{\pi}$ is the best path found.

2.2 Distributed Implementation of the Cross Entropy Method

Rubinstein's CE algorithm is centralized, synchronous and batch oriented. All results output from each step of the algorithm must be collected before the next step can be executed. In [HW01] a distributed and asynchronous version of Rubinstein's CE algorithm is developed. A few approximations let (3) and (1) be replaced by the autoregressive counterparts

$$P_{t+1,rs} = \frac{\sum_{k=1}^{t} I(\{r,s\} \in \pi_k) \beta^{t-k} H(\pi_k, \gamma_t)}{\sum_{l=1}^{t} I(\{r\} \in \pi_l) \beta^{t-k} H(\pi_l, \gamma_t)} \tag{4}$$

and

$$\min \gamma_t \text{ s.t. } h_t^{'}(\gamma_t) > \rho \tag{5}$$

respectively where

$$\begin{aligned} h_t^{'}(\gamma_t) &= h_{t-1}^{'}(\gamma_t)\beta + (1-\beta) H(\pi_t, \gamma_t) \\ &\approx \frac{1-\beta}{1-\beta^t} \sum_{k=1}^{t} \beta^{t-k} H(\pi_k, \gamma_t) \end{aligned}$$

and $\beta < 1$, step 2 and 3 can immediately be performed when a single new path π_t is found and a new probability matrix P_{t+1} can be generated.

The distributed CE algorithm may be viewed as an algorithm where search agents evaluate a path found (and calculate γ_t by (5)) right after they reach their destination node and then immediately return to their source node backtracking along the path. During backtracking relevant probabilities in the transition matrix are updated by applying $H(\pi_t, \gamma_t)$ through (4).

The distributed CE algorithm resembles Schoonderwoerd & al.'s original system. However Schoonderwoerd's ants update probabilities during their forward search.

Dorigo & al. realized early in their work on ACO that compared to other updating schemes, updating while backtracking results in significantly quicker convergence towards high quality paths. Dorigo & al.'s AntNet system [CD98] implements updating while backtracking, thus is more similar to the distributed CE algorithm than Schoonderwoerd & al.'s system. However none of the earlier systems implements a search focus stage (the adjustment of γ_t) as in the CE algorithms.

2.3 P-cycles, Hamiltonian Cyclic Paths and CE algorithms

Grover's "p-cycles" [GS98] provide protection against a single link failure on any link connecting the nodes which are on the path defined by the p-cycle. This includes both on-cycle links (links traversed by the path) as well as straddling links (links not traversed but having their end nodes on the path). Intuitively a Hamiltonian cyclic path, which by definition visits all nodes once in a network, would provide a cycle potentially able to protect against any single link failure. This is also argued in [SG00].

The CE algorithms from both [Rub99] and [HW01] show good performance when tested on optimal Hamiltonian cyclic path search problems as long as the network environment is fully meshed (all nodes have direct duplex connections). Real world telecommunication networks are seldom fully meshed. An average node degree much lager than 5 is uncommon. Finding a single Hamiltonian cyclic path in a large network with such scarce meshing can itself be considered a rare event.

In the section 3.1 we describe the selection strategy (used in CE algorithm step 2) implemented in the original CE algorithms (both [Rub99] and [HW01]). They strategy struggles to find Hamiltonian cyclic paths in our 26 node test network shown in Figure 2. In section 3.2 and 3.3 we suggest two new selection strategies intended to better cope with a network topology with scarce meshing.

3. Selection Strategies

3.1 Markov Chain Without Replacement

The CE algorithms in [Rub99] and [HW01] implement a strict next-hop selection strategy termed *Markov Chain Without Replacement* (MCWR) in [Rub99]. No nodes are allowed to be revisited, except for the home node when completing a Hamiltonian cyclic path.

Let

$$X^i_{t,r}(s) = I(s \notin \mathbf{V}^i_t \vee ((\mathbf{G}_{t,r} \subseteq \mathbf{V}^i_t) \wedge s = hn^i))$$

where $I(\ldots)$ is the indicator function, $\mathbf{V}^i_t$ is agent i's list of already visited nodes, $\mathbf{G}_{t,r}$ is the set of neighbor nodes to node r and hn^i is agent i's home node. Thus $X^i_{t,r}(s)$ is 1 if node s has not already been visited by agent i, or if all neighbor nodes of r have been visited by agent i and s is agent i's home node.

When MCWR is applied $P_{t,rs}$ from (4) is weighted by $X^i_{t,r}(s)$ and renormalized giving a new next-hop probability distribution

$$Q^i_{t,rs} = \frac{[I(t > D)P_{t,rs}(1-\epsilon) + \epsilon]\, X^i_{t,r}(s)}{\sum_{\forall k} [I(t > D)P_{t,rs}(1-\epsilon) + \epsilon]\, X^i_{t,r}(k)}$$

where D is the number of path samples required to be found to complete the initialization phase of the system (step 1). The random noise factor ϵ is set to a small value, e.g. 10^{-60}. During the initialization phase agents are forced to explore since the next-hop probability vector $\mathbf{Q}^i_{t,r}$ will have a uniform distribution over the qualified ($X^i_{t,r}(s) = 1$) neighbor nodes. See [HW01] for more details about the initialization phase.

If $\sum_{s \in \mathbf{G}_{t,r}} X^i_{t,r}(s) = 0$, agent i has reach a dead end and in the MCWR strategy it is terminated. When the event of finding a Hamiltonian cyclic path is rare due to scarce meshing in a network, most agents will reach such dead ends. Thus only a few "lucky" agent will be able to contribute with a path in step 2 of the CE algorithm. This will slow down the search process significantly since CE algorithms require a "smooth" search space, i.e. many suboptimal solutions should exist in addition to the optimal solutions.

3.2 Markov Chain Depth First

Instead of immediately terminating agents when dead ends are reached a "retry mechanism" can be implemented. We have tested what we call the *Markov Chain Depth First* (MCDF) strategy which allows agents to backtrack and retry searching. An MCDF-agent performs a depth first search [RW88] from its home node, i.e. it tries to visit nodes in such an order that when a dead end is met (a leaf node is found) all nodes have been visited only once and the home node is a neighbor of the leaf node. If a dead end is reached and either all nodes has not been visited or the home node is not a neighbor node, the agent backtracks along its path one step before continuing the search.

Let

$$X^{i*}_{t,r}(s) = I(s \notin (\mathbf{V}^i_t \cup \mathbf{D}^i_{t,r}) \;\vee\; ((\mathbf{G}_{t,r} \subseteq \mathbf{V}^i_t) \;\wedge\; s = hn^i))$$

where $\mathbf{D}^i_{t,r}$ is the set of neighbor nodes of r leading to dead ends for agent i. Thus $X^{i*}_{t,r}(s)$ is 1 if node s has not already been visited by agent i and s does not lead to a dead end, or (as for $X^i_{t,r}(s)$) if all neighbor nodes of r have been visited by agent i and s is agent i's home node.

All $\mathbf{D}^i_{t,r}$ (for $\forall_r$) are stored in a stack managed by agent i. When a fresh next node r_{+1} is chosen $\mathbf{D}^i_{t,r_{+1}} \equiv \emptyset$ is pushed onto the stack. If a dead end is reached at node r_{+1} agent i backtracks to the previously visited node r, removes (pops) $\mathbf{D}^i_{t,r_{+1}}$from the stack and adds r_{+1} to $\mathbf{D}^i_{t,r}$ (which is now on the top of the stack).

When MCDF is applied $P_{t,rs}$ is weighted by $X^{i*}_{t,r}(s)$ in the same way $X^i_{t,r}(s)$ is for MCWR. Results in section 4 show simulation scenarios for MCDF-agents both with unlimited and limited backtracking. Unlimited backtracking implies never terminating agents but letting them search (in depth first fashion) until they find Hamiltonian cyclic paths. Limited backtracking implements a quota of backtracking steps in each agent, i.e. a certain no of “second chances” or “retries” are allow for an agent before termination.

3.3 Markov Chain with Restricted Replacement

By relaxing the agent termination condition even more, we arrive at what we call the *Markov Chain with Restricted Replacement* (MCRR) strategy. The fundamental difference between this strategy and both MCWR and MCDF is a less strict condition concerning revisits to nodes. Revisits are simply allowed, but only when dead ends are reached. To ensure completion of cycles the home node is given priority when a dead end is reached and the home node is a neighbor node.

Let

$$\begin{aligned} X^{i**}_{t,r}(s) \;=\;& \overline{I}(((\mathbf{G}_{t,r} \not\subseteq \mathbf{V}^i_t) \;\wedge\; s \in \mathbf{V}^i_t) \vee \\ & ((\mathbf{G}_{t,r} \subseteq \mathbf{V}^i_t) \;\wedge\; hn^i \in \mathbf{G}_{t,r} \;\wedge\; s \neq hn^i)) \end{aligned}$$

where $\overline{I}(\ldots)$ is the inverse indicator function. Thus $X^{i*}_{t,r}(s)$ is zero if an unvisited neighbor node to r exists and s has already been visited, or if all neighbor nodes have been visited and the home node is a neighbor node but s is not the home node. As for MCWR and MCDF $X^{i**}_{t,r}(s)$ weights $P_{t,rs}$.

In our simulations we consider only paths found by MCRR-agents which have visited all nodes when they return to their home node. Several of these agents will find closed acyclic paths (with loops), i.e. none Hamiltonian cyclic paths. The search space is now “smoother” and a range of suboptimal solutions exists (most none Hamiltonian). This enables step 2 in the CE algorithm to be executed with close to the same efficiently as for a fully meshed network.

However when using MCRR-agents there is no longer guaranteed that the best path found when the system converges is a Hamiltonian cyclic path. Since the agents visit all nodes, the length of acyclic closed paths where nodes have been revisited, are likely to be longer than Hamiltonian cyclic paths. Thus finding the shortest Hamiltonian cyclic path (minimization) may still be achievable.

We realize that the above statement does not hold in general since a network topology may be constructed having its shortest Hamiltonian cyclic path longer than one or more closed acyclic paths visiting all nodes. However in the context of p-cycle design closed acyclic paths may still provide protection against single link failures.

Results in section 4 are promising. MCRR outperforms both MCWR and MCDF when it comes to speed of convergence, and do in all simulation scenarios converge to a Hamiltonian cyclic path.

4. Strategy Performance

As for the simulation scenarios in [HW01] we have used an active network enabled version of the *Network Simulator* version 2 [DAR] to test the three selection strategies. Our test network topology is show in Figure 2, a 26 node network with an average number of outgoing links (degree) per node equal to 5. The low av-

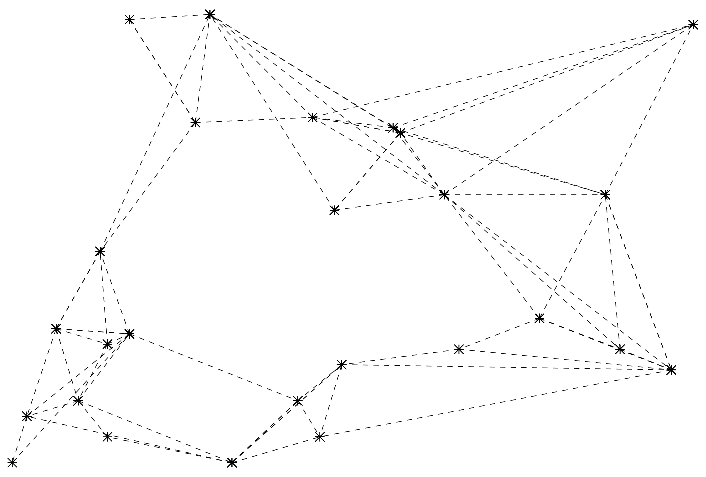


Figure 2. Network topology used in simulation scenarios. The topology is generated by the *Tiers 1.1* topology generator [CDZ97].

erage degree implies existence of far less Hamiltonian cyclic paths compared to a fully meshed 26 node network with the number of Hamiltonian cyclic paths equal to $25! \approx 10^{26}$. The network topology was generated by the *Tier 1.1* topology generator [CDZ97] with the parameter vector "1 0 0 26 0 0 9 1 1 1 1 9".

All scenarios had equal parameter settings (except for different selection strategies): $D = 19, \mu = 26, \beta = 0.998, \rho = 0.01, \rho^* = 0.95$, where μ is the number of agents operating concurrently and ρ^* the ρ reduction factor. See [HW01] for further descriptions of the parameters.

Table 1 and 2 compare results form simulation scenarios for the different selection strategies. Results shown are values recorded after 100, 1500 and 10 000 seconds of simulation time. Time has been chosen as the scale of progress rather than number of iterations since the time spent per iteration (per search) vary significantly between the different strategies. Simulation time is expected to be approximately proportional to real time in a network.

Table 1. Number of paths found in simulation scenarios using the three different selection strategies. Values are averaged over 10 simulations and reported with standard deviations (prefixed by $\pm$). Numbers in bracket are the number of agents which have finished their initialization phase.

Test	*No of Hamiltonian cyclic paths found*			*No of none Hamiltonian cyclic paths*		
scenarios	*100s*	*1500s*	*10 000s*	*100s*	*1500s*	*10 000s*
MCWR	1.0 $\pm$0 (0)	16.0 $\pm$2.3 (0)	109 $\pm$8.7 (0)	16.0e3 $\pm$52.1	246e3$\pm$213	1.6e6$\pm$533
MCDF unlimited	0.0$\pm$0 (0)	1.0 $\pm$0.0 (0)	2.57 $\pm$1.3 (0)	201$\pm$30.6	1.5e3$\pm$79.1	9.3e3$\pm$190
MCDF quota=10	1.0 $\pm$0 (0)	14.0 $\pm$4.5 (0)	85.0 $\pm$12.4 (0)	8.8e3$\pm$24.3	134e3$\pm$156	901e3$\pm$345
MCDF quota=5	2.0 $\pm$0.9 (0)	16.0 $\pm$4.4 (0)	95.0 $\pm$10.8 (0)	10.1e3$\pm$34.1	155e3$\pm$151	1.0e6$\pm$402
MCDF quota=2	1.0 $\pm$0 (0)	14.0 $\pm$3.7 (0)	99.0 $\pm$12.1 (0)	11.5e3$\pm$40.9	176e3$\pm$126	1.2e6$\pm$316
MCDF quota=1	2.0 $\pm$1.0 (0)	16.0 $\pm$5.5 (0)	105$\pm$12.9 (0)	12.3$\pm$35.5	188e3$\pm$101	1.3e6$\pm$367
MCRR	7.0 $\pm$2.8 (26)	**52 800$\pm$4620 (26)**	(converged)	4.9e3$\pm$56.0	**15.7e3$\pm$1580**	(converged)

Table 2. Quality of paths found in simulation scenarios using the three different selection strategies. Values are based on 10 simulation. Standard deviations are prefixed by ± and the worst of the best values are given in brackets.

Test	*Best path found (Worst of best paths found)*			*Average path cost*		
scenarios	*100s*	*1500s*	*10 000s*	*100s*	*1500s*	*10 000s*
MCWR	0.199 (0.251)	0.197 (0.214)	0.193 (0.207)	0.231±0.015	0.232±0.004	0.231±0.004
MCDF unlimited	(no paths found)	0.250 (3.273)	0.215 (0.750)	(no paths found)	0.923±1.023	254.5±670.5
MCDF quota=10	0.202 (0.371)	0.202 (0.222)	0.194 (0.207)	0.260±0.058	0.254±0.008	0.254±0.008
MCDF quota=5	0.201 (0.240)	0.201 (0.221)	0.196 (0.209)	0.228±0.014	0.240±0.011	0.243±0.008
MCDF quota=2	0.198 (0.293)	0.194 (0.224)	0.194 (0.206)	0.244±0.028	0.241±0.006	0.241±0.005
MCDF quota=1	0.204 (0.258)	0.201 (0.210)	0.193 (0.204)	0.231±0.016	0.235±0.003	0.238±0.003
MCRR	0.203 (0.230)	**0.194 (0.202)**	(converged)	0.514±0.037	**0.206±0.015**	(converged)

Columns 2, 3 and 4 in Table 1 show the total number of Hamiltonian cyclic paths found including re-discoveries. Values are averaged over 10 simulations and reported with standard deviations (prefixed by ±). Numbers in bracket are the number of agents which have finished their initialization phase, i.e. changed search behavior from doing random walk guided only by a selection strategy to being guided both by a selection strategy and cross entropy adjusted probabilities (pheromones). Column 5, 6 and 7 show the total number of none Hamiltonian cyclic paths found, including dead ends and paths not visiting all nodes.

Column 2, 3 and 4 in Table 2 show the best of the best paths found (lowest cost) in 10 simulations with the worst of the best in brackets. And finally columns 5, 6 and 7 show path cost averaged over 10 simulations and reported with standard deviations (prefixed by ±).

4.1 Markov Chain without Replacement

As expected in the MCWR scenario few Hamiltonian cyclic paths are found even after 1500 simulation seconds. The scarce meshing in the test network results in many agent reaching dead ends. The fraction of feasible to infeasible paths found is as low as $6.5 \cdot 10^{-5}$ after 1500 seconds. The paths found are of relatively good quality, but still after 10 000 seconds none of the agents have managed to collect enough paths samples to finish their initialization phases.

These results indicate the need for a different selection strategy when searching for Hamiltonian cyclic paths in networks with scarcely meshed topologies.

4.2 Markov Chain Depth First

The results for the MCDF scenarios are not promising. When unlimited backtracking is enabled even fewer path are found than for the MCWR scenario. The path quality is low because the total search time (including backtracking) is registered as path cost. This is not surprising since unlimited backtracking implies that every agent is doing an exhaustive search for Hamiltonian cyclic paths. The very reason for introducing stochastic search techniques in the first place is to avoid the need for such exhaustive searches.

When the MCDF strategy is limited by quotas of 10, 5, 2 or 1 retry the performance improves compared to unlimited backtracking but is not better than the original MCWR strategy. Also with this strategy not enough valid paths have been found after 10 000 seconds to enable any agents to complete their initialization phase. Best and average path costs are similar to the values for MCWR and no convergence is observed due to the overall low number of valid paths found.

4.3 Markov Chain with Restricted Replacement

The last row in Table 1 and 2 which presents results from MCRR scenarios, stand out from the other results. Already after 100 simulation seconds all 26 agents have completed their initialization phases, and after 1500 seconds the average path cost has converged to a value close to the best path found (see bold values). Since full convergence is experienced already around 1500 seconds, no values are given for 10 000 seconds. For all 10 simulations the best path found is a true Hamiltonian cyclic path. Figure 3 show how the ratio of Hamiltonian to none Hamiltonian cyclic paths found increases during the search process. Not long after 800 seconds most agents start finding Hamiltonian cyclic paths.

We believe these results indicates the value of using infeasible paths (none Hamiltonian cyclic paths) as intermediate solutions during the iterative search process of the CE algorithm. The question of accepting or rejecting infeasible solutions is well known to designers of evolutionary systems [Gol98, Mic96]. When feasible solutions are located far from each other in the search domain, a significant speedup in the search process can be achieved by letting infeasible solutions act as "stepping stones" between suboptimal and optimal feasible solutions.

5. Concluding Remarks

As an increasing number of applications demanding QoS guarantees are accessing internet, dependable routing is becoming more important than ever. This paper examines and enhances a distributed cross entropy founded algorithm designed to find cyclic paths (Hamiltonian cyclic paths) in networks. Such paths are argued to be good candidate paths when protection switching is to be implemented in meshed networks using p-cycles [SG00]. The algorithm presented is well suited for distributed implementation, for instance using mobile agents or active networks technology.

Previous versions of the algorithm [Rub99, HW01] struggle to find Hamiltonian cyclic paths in scarcely meshed networks, very much because of the strict selection strategy (Markov Chain without Replacement) in operations during the search process. In this paper we compare the original selection strategy with two new strategies. The first new strategy, *Markov Chain Depth First*, proves to be as inefficient as the original, while the second, *Markov Chain with Restricted Replacement*, outperforms both the other strategies. However, using the second strategy convergence towards feasible solutions (i.e. Hamiltonian cyclic paths) is not guaranteed. Even so results

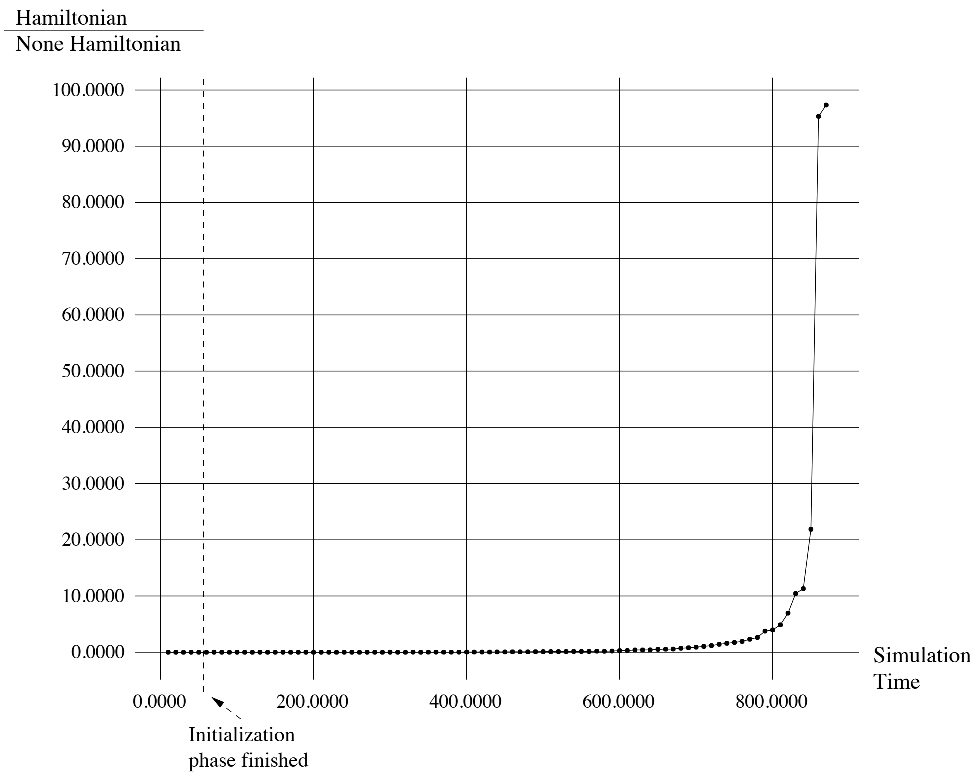


Figure 3. Ratio of Hamiltonian to none Hamiltonian cyclic paths found when using the MCRR selection strategy.

from simulation scenarios indicate a high probability of converging towards near optimal Hamiltonian cyclic paths.

More excessive parameter tuning of the algorithm is required. Also the possibility of including heuristics to speed up the convergence process even more should be investigated.

Currently a new version of the algorithm is under development which allows several species of agents compete in finding quality paths in a network. By this a set of cyclic paths providing an overall high quality p-cycle design may potentially be found.

Appendix: Additional Results

This appendix is not part of the original paper, however the results described here were included in the presentation of the paper at the ANTS 2002 workshop in Brussels. The results have some significance and are considered important for the comprehensiveness of the paper.

As mention in Section 4 of this paper, the numbers given in Table 1 for Hamiltonian cyclic paths include re-discoveries. The number of re-discoveries of paths will increase significantly when the search process begins to converge. Hence comparing the numbers for the MCRR scenario with the other numbers in Table 1 has limited value since MCRR is the only scenario which enters a convergence phase before simulation is terminated. In this appendix we compare the number of *unique* Hamiltonian cyclic paths found by the scenarios. The number of unique paths found indicate how efficiently the different strategies explore the search space to find feasible solutions. Table A.1 is similar to the first four columns of Table 1 except that only unique Hamiltonian cyclic paths are counted.

Table A.1. Number of unique paths found in simulation scenarios using the three different selection strategies. Values are averaged over 10 simulations and reported with standard deviations (prefixed by ±). Numbers in bracket are the number of agents which have finished their initialization phase.

Test	*No of unique Hamiltonian cyclic paths found*		
scenarios	*100s*	*1500s*	*10 000s*
MCWR	0.7 ±0.8 (0)	15.8 ±2.6 (0)	108 ±8.8 (0)
MCDF unlimited	0.0±0 (0)	0.3 ±0.7(0)	1.6 ±1.2 (0)
MCDF quota=10	1.0 ±0.9 (0)	13.9 ±5.2 (0)	84.3 ±12.6 (0)
MCDF quota=5	1.3 ±0.9 (0)	15.3 ±4.7 (0)	95.0 ±11.3 (0)
MCDF quota=2	0.9 ±1.0 (0)	13.8 ±3.7 (0)	98.3 ±12.1 (0)
MCDF quota=1	1.3 ±0.8 (0)	16.0 ±6.3 (0)	105±13.0 (0)
MCRR	0.3 ±0.5 (26)	**115±40.5 (26)**	(converged)

The number of unique paths for the first six strategies, MCWR and all variants of MCDF, do not differ significantly from the total number of paths given in Table 1. Hence the first six strategies seldom re-discover a path. However the last strategy, MCRR, start re-discovering paths early. Already after 100 seconds of simulation time only on average 0.3 out of 7.0 paths found are unique. Even so, MCRR manages to find on average 115 unique paths already after 1500 seconds. 115 paths is a greater average number of unique paths than any of the other strategies ever find before simulation is terminated. Hence the results in Table A.1 presents further evidence in favor of allowing infeasible solutions during the search process.

References

[AP94] S. Aidarous and T. Plevyak, editors. *Telecommunications Network Management into the 21st Century*. IEEE Press, 1994.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[CDZ97] K. I. Calvert, M. B. Doar, and E. W. Zegura. Modeling Internet Topology . *IEEE Communications Magazine*, 35(6):160 –163, June 1997.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[DC99] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137–172, 1999.

[De95] M. Decina and T. Plevyak (editors). Special Issue: Self-Healing Networks for SDH and ATM. *IEEE Communications Magazine*, 33(9), September 1995.

[Gol98] D. Goldberg. *Genetic Algorithms in Search, Optimization and MachineLearn ing*. Addison Wesley, 1998.

[GS98] W.D. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 537 –543, 7-11 June 1998.

[HW01] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[Mic96] Zbigniew Michalewicz. *Genetic algorithms + Data Stuctures = Evolution Programs*. Springer Verlag, second edition, 1996.

[PK98] Vu Anh Pham and A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine*, 36(7):26–37, July 1998.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[Rubar] Reuven Y. Rubinstein. The Cross-Entropy and Rare Events for Maximum Cut and Bipartition Problems - Section 4.4. *Transactions on Modeling and Computer Simulation*, To appear.

[RW88] Kenneth A. Ross and Charles R. B. Wright. *Discrete Mathematics*. Prentice Hall, 2nd edition, 1988.

[Sch00] J. Schuringa. Packet Routing with Genetically Programmed Mobile Agents. In *Proceedings of SmartNet 2000*, Wienna, September 2000.

[SG99] D. Stamatelakis and W.D. Grover. Rapid Span or Node Restoration in IP Networks using Virtual Protection Cycles. In *Proceedings of 3rd Canadian Conferance on Broadband Research (CCBR'99)*, Ottawa, 7 November 1999.

[SG00] D. Stamatelakis and W.D. Grover. Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles ("p-cycles"). *IEEE Transactions on Communications*, 48(8):1262–1265, August 2000.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[WH02] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.

[WPO98] T. White, B. Pagurek, and Franz Oppacher. Connection Management using Adaptive Mobile Agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDAPTA'98)*, 1998.

[ZBMD00] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based Search for Combinatorial Optimization. IRIDIA IRIDIA/2001-15, Universite Libre de Bruxelles, Belgium, 2000.

PAPER D

Robust Implementation of Policies Using Ant-like Agents

Otto Wittner and Bjarne E. Helvik

The Network Control and Engineering for QoS, Security and Mobility with focus on Policy- based Networking IFIP and IEEE Conference, Net-Con'2002

October 23-25th, Paris, France.

NOTE: A slightly extended version of this paper will be published in the *Annals of Telecommunications Journal, Special Issue on "Policy-based Control"* in January/February 2004.

ROBUST IMPLEMENTATION OF POLICIES USING ANT-LIKE AGENTS

Otto Wittner

Bjarne E. Helvik

Abstract Policy based management is a powerful means for dealing with complex heterogeneous systems. However, the policies are commonly strictly interpreted, and it is tempting to resort to centralized decisions to resolve conflicts. At the same time, swarm intelligence based on “ant like” mobile agents has been shown to be able to deal with challenging optimization and trade-off problems. This paper discusses and demonstrates how policies may be used to govern the behavior of mobile agents to find near optimal solutions for the implementation of a set of potentially conflicting policies in a truly distributed manner. A more dependable/robust system is obtained. The enforcement of the policies is soft in the sense that it is probabilistic and yields a kind of “best effort” implementation. A case study illustrating how ant like mobile agents may implement load distribution and conflict free back-up policies, is presented.

Keywords: Policy enforcement, dependability, swarm intelligence, ant-like agents, mobile agents

1. Introduction

Today’s computer and telecommunication network environments consists of a heterogeneous mix of network technologies, computing devices and applications. Policy based management [Slo94b] is a recognized concept for organizing management operations in such complex network environments. Results form research activities the last decade include standardization on policy management architectures, information models and protocols [MESW01, BCD$^+$00] as well as a range of policy specification languages, cf. for instance Chapter 2.2 of [Dam02] for a survey.

The traditional approach to policy enforcement yields basically an implementation of deterministic rules. Furthermore, in order to resolve potential conflicts between the different policies, there is a tendency to resort to methods requiring centralized decision making [LS99d]. Avoiding centralization is one reason (among others) for not always eliminating policy conflicts. Strategies for resolving policy conflicts have been studied by several researchers [LS99d, CLN00].

In this paper we present a technology, known as *swarm intelligence* [BDT99], applicable for enforcement of process policies in a distributed and robust manner. The fundamental components of swarm intelligence systems are simple, autonomous agents with a stochastic behavior similar to "agents" found in biological systems in nature, e.g. ant colonies. A high number of such agents are in action simultaneously communicating indirectly by changing the environment. The overall concept of swarm intelligence has several desirable properties from a dependability point of view: High redundancy, true distribution and adaptability. Section 2 of this paper introduces the fundamentals of swarm intelligence and agents with ant-like behavior. Such agents are proposed to implement policy management systems. The stochastic behavior of the agents is interpreted as *soft policy enforcement,* i.e. the policies specifying an agent's behavior will be enforced with some probability which yields a kind of "best effort" enforcement of conflicting policies and policies not fully enforceable due to resource limitations in the network. Section 3 describes how a swarm based system may implement a set of policies, where subsection 3.3 describes how swarm based systems can resolve policy conflicts.

To demonstrate the potential of swarm intelligence implementations of policies, section 4 presents a case study where policies for connection management in a network are considered. Finally in section 5 we summarize, conclude and indicate future work.

2. Fundamental Agent Behavior

Swarm intelligence may be defined as "... *any attempt to design algorithms or distributed problem-solving devices inspired by collective behavior of social insect colonies and other animal societies*" [BDT99]. Pioneering work in this field of research was performed decades ago, however the first attempts of applying swarm intelligence in a telecommunication management context was done by Steward and Appleby [SA94] and has later been developed by several other researchers [SHBR97] [CD98][HW01].

2.1 Informal Behavior Outline

In this paper we concentrate on the Cross Entropy founded algorithm (CE algorithm) from [HW01] which is inspired by the foraging behavior of ants. The algorithm has some fundamental elements in common with most swarm intelligence systems, with the extension that the search is successively focused by lowering a temperature parameter as better solutions are found. In addition, fundamental to our approach is the interaction between different types of ant-like agents as introduced in [WH02b].

Many asynchronous, autonomous agents. A large number of autonomous simple agents move around in a given environment represented as a graph $\mathcal{G}$ with set $V(\mathcal{G})$ of vertices and $E(\mathcal{G})$ of edges. A unique type of agent represents a policy combined with the target for enforcement of this policy. Enforcement is implemented by having agents utilize resources connected to vertices and edges, see section 4.2 for exemplification.

Stochastic search for solutions. Each agent search in a stochastic manner for a solution to a given problem, i.e. a policy enforcement, similar to ants' behavior when searching for a food source. A solution is represented as a path $\pi_t = \{si, ij, \ldots, kl, ld\}$ where $s, i, j, k, l, d \in V(\mathcal{G})$. An agent finds a path by doing a random walk combined with a search strategy (e.g. avoid revisiting vertices). Each step in the random walk is controlled by a probability vector $P_{t,i}$ where i is the currently visited vertex[1] .

Indirect, asynchronous communication. Agents communicate indirectly and affect each others behaviors by changing the environment. When a solution path π_t is found, an agent backtracks along the path and leaves virtual *pheromone* on every edge traversed. Real pheromones are chemical substances released by an ant to signal some message to other ants. Virtual pheromones are weights which influence the probability distributions $P_{t,i}$ which again influence the search behavior of the agents (cf. previous item).

Iterative with positive feedback. The search process is iterative and accelerated by positive feedback. When a solution is found and backtracking completed, an agent restarts forward search. Edges in high quality paths receive high pheromone weights. A high weight on an edge increase the probability for an agent to traverse the edge in successive searches. Pheromones placed as a result of successive searches reinforce the high weights even more. Over time high quality solutions emerge as sequences of highly weighted edges, resembling the emergence of ant trails. The CE algorithm speeds up this process further by making the agents more selective with respect to what a high quality solution is as better solutions are found [HW01, Rub99].

Together these components have been shown to enable design of systems capable of finding optimal or near optimal solutions to complex optimization problems classified as NP-hard without any need of centralized control. Network management of large heterogeneous telecommunication networks involve solving different optimization problems with NP-hard complexity, e.g. routing and network planning. Decentralized techniques for performing such optimization are indeed of interest and may be worth incorporating into policy based network management systems.

2.2 Agent Algorithm

The actual steps in the CE algorithm that will be used in the case study of section 4, may be described as follows. Keep in mind that the algorithm describes the behavior of one type of agent. In multi-type agent systems, each type will have its separate set of parameters N, variables $P_{t,i}$, γ_t, functions L_{ij} and hence, $H(\pi_t, \gamma_t)$.

1 At iteration $t = 0$ initialize all probability vectors ($P_{t=0,i} \mid_{\forall i \in V(\mathcal{G})}$) to uniform distributions.

[1]Each type of agent have their individual probabilities. For the sake of simplifying the introduction, the sub and superscripts necessary to identify the types are not shown.

2 Create N agents. For each agent perform the step 3-5.

3 Generate a path π_t by performing a random walk based on the probability vectors $P_{t,i} \mid_{\forall i \in V(\mathcal{G})}$ and some search strategy ([WH02a]).

4 At the destination vertex calculate the optimal Boltzmann temperature γ_t which controls $H(\pi_t, \gamma_t) = e^{-\frac{L(\pi_t)}{\gamma_t}}$, the performance function returning the quality of path π_t. $L(\pi_t) = \sum_{ij \in \pi_t} L_{ij}$ is the raw cost of path π_k which again is the sum of edge costs for all edges in the path. The edge cost is related to the success of the policy enforcement on the edges as will be discussed in section 4. The optimal solution for γ_t will result in a certain amplification of high quality paths and a minimum weighted average of all path qualities for paths found so far, i.e. search focus is directed towards good candidate solution paths. The enforcement of potentially conflicting policies are "signaled" through L_{ij} which may include calculations based on pheromones from alien species of ant-like agents, see section 3.3.

5 Using γ_t and $H(\pi_t, \gamma_t)$ from step 4, generate new probability vectors $P_{t+\delta,i}$ by minimizing the cross entropy between $P_{t,i} \mid_{\forall i}$ and $P_{t+\delta,i} \mid_{\forall i}$, where δ is the incremental time to the next agent starts its search. Minimizing the cross entropy ensures an optimal shift in the probabilities with respect to γ_t and the performance function. The generation of $P_{t+\delta,i} \mid_{\forall i}$ is achieved by adjusting relevant probabilities of $P_{t,i} \mid_{\forall i}$ during re-traversal of path π_t.

6 Repeat steps 3-5 until $H(\widehat{\pi}, \gamma_t) \approx H(\widehat{\pi}, \gamma_{t+\Delta})$ where $\widehat{\pi}$ is the best path found and Δ is an appropriately chosen time.

For details on the CE algorithm and the formal foundations the reader is referred to [Rub99, HW01]

3. From Policies to Agent Behavior

The agent behavior, described in the previous section, has hereto been adapted and used to handle specific optimization problems, either as stand alone "optimization engines", e.g. [Rub99][KGV83][DG97], or as an integrated part of network O&M, [HW01][WH02b][WH02a]. However we suggest to take a policy view point of the problem at hand and base the agent behavior on a set of policies. Soft enforcement of policies may be regarded as minimizing the deviation from the policies under constraints set by resource limitations in the network, network topology and operation. In the following subsections we describe the steps from policy specification to policy implementation by ant-like agents. We create policies at two levels, and in accordance with [MC93] we apply the term *functional policies* for upper level policies and *process policies* for lower level policies.

3.1 Problem Fundamentals: Functional Policies

Elements connecting the agent behavior to a specific problem are the *graph* ($\mathcal{G}$) and the *edge costs* (L_{ij}).

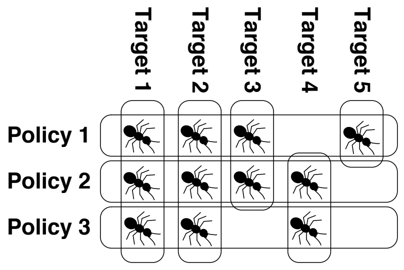


Figure 1. Policies and target organization. Each policy target instance is allocated one unique type of ant-like agent.

The *graph* requires a mapping onto the solution space for the problem at hand. A one-to-one mapping is desirable such that all possible solutions from the solution space are represented in the graph and no solutions being none-existing in the solution space can be found in the graph. In section 4.2 nodes and link resources in an IP backbone network are mapped to graph vertices and edges respectively.

Given a graph representation, the edge costs and the path performance function control the core behavior of an agent. As the first step in the design process for defining these elements, we suggest to *establish a set of policies,* $\{Fr\}_{r=0,\ldots,R}$*, which identify requirements for relevant targets fundamental in the management problem.* It is assumed that these policies are ordered according to their importance, hence, r is referred to as the rank of the policy.

Targets of the management problem are the managed entities in the network which shall comply with one or more of the policies. The sub- or superscript m refers to a target. Examples of targets are virtual paths, cf. the case study, and databases. In general, targets are associated with vertices, edges as well as patterns of vertices and/or edges in the graph. Figure 1 illustrates how policies and targets may be organized. Some targets may be treated by several policies while others only by one or a few. In section 4.1 two functional policies, F1 and F2, are established each treating 12 different target paths which again are sequences of link resources.

As a step in the enforcement implementation of the policies we allocate one unique type of ant-like agent to each policy-target instance, mr, as illustrated in Figure 1. Thus a type of ant-like agent is intended to be designed specifically to enforce a certain policy for a certain target.

3.2 Behavior Details: Process Policies

To better understand what behavior details will be required for our agents, we refine the functional policies from the previous section and establish a set of process policies, i.e., each functional policy is mapped to a set of process policies, $Fr \to \{Pr1, \ldots, Prn_r\}$.

The next step is to establish a measure of fulfillment of the requirements stated by the process polices. The measure will indicate how desirable the execution of a specific action is and will typically be related to vertices and/or edges in the graph.

For the sake of simplicity, we regard edges only. The fulfillment with respect to policy-target instance mr on edge ij is a function given by the process policies, i.e., $L_{ij}^{mr} = f_{P1,\dots,Pn}(\{a_{ij}\},\{c_{ij}\})$ where $\{a_{ij}\}$ is the set of requirements/demands associated with the edge and $\{c_{ij}\}$ is the set of resources available on the edge to meet these requirements.

3.3 Resolving Policy Conflicts by Detestation

During enforcement of several policy-target instances (Figure 1) conflicts may arise. If limitations in the resources required by several policy target instances exists, one or more policy target instances may not be enforced. Hence some sort of conflict resolution mechanism is required.

An ant-like agent allocated to a policy target instance uses a unique pheromone type, i.e. the agent maintains a reserved set of variables on each vertex and/or edge in the graph which indicate if the specific part of the graph is likely to be part of the final solution enabling enforcement of the policy-target instance. Agents update only their own pheromones but may still read alien agents' pheromones.

As a mechanism for conflict resolution we include pheromone levels of alien agents as weights in the cost function. If alien pheromones indicate that a required resource is likely to be allocated by one or more alien agents, the weights will increase the cost of accessing the resource and eventually result in a reduced probability for the agent in question to revisit the resource in the next search for a policy enforcement solution. This effect can be seen as having *agents with conflicting interests detest each other* [VS99b, WH02b]. In section 4.2 agents detest each other with different degrees depending on what policy-target instance they are allocated to. Semi-formally, we extend our edge cost function to $L_{ij}^{mr} = g_{P1,\dots,Pn}(\{a_{ij}\},\{c_{ij}\},\{P_{t,i}^{ns}\})$ where $\{P_{t,i}^{ns}\}$ is the pheromone trail left by all other types of agents, and it is implicit that the policies may cause conflicts.

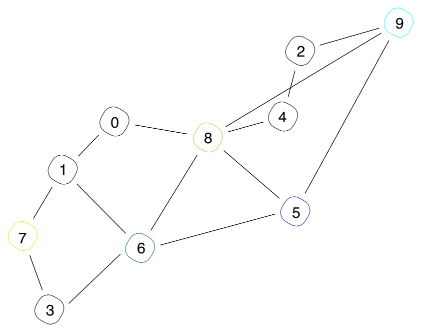
3.4 Soft Policy Enforcement

Two aspects of the agent behavior result in a less than 100% guaranty of finding enforcement solution for all policy target instances.

Firstly, the fundamental stochastic behavior of the agents combined with the size of the problem solution space renders it impossible to guaranty all potential solutions evaluated. However in [Rub99] clear indications are given that CE algorithms with high probability converge towards a near optimal solution.

Secondly, in complex networks with multiple conflicting policies, a solution complying with all policies may not exist. The detestation weights introduced to handle policy conflicts will, however, force agents to search in parts of the solution space where the policies are partially enforced. This is likely to result in solutions enforcing most policies to such a degree that the overall system behavior is satisfactory, i.e., a best effort policy enforcement.

The stochastic behavior of the agents as well as the detestation properties adds a potential ability of adaptation to the environment [Rub01]. Should the target network



Connection	Primary Path	Backup Path
1–6	1–6	1–7–3–6
1–8	1–0–8	1–6–8
1–9	1–0–8–9	1–7–3–6–5–9
6–1	6–1	6–3–7–1
6–8	6–8	6–5–8
6–9	6–5–9	6–8–4–2–9
8–1	8–0–1	8–6–1
8–6	8–6	8–5–6
8–9	8–9	8–4–2–9
9–1	9–8–0–1	9–5–6–3–7–1
9–6	9–5–6	9–2–4–8–6
9–8	9–8	9–2–4–8

Figure 2. Left: The Norwegian university IP backbone network consisting of 10 nodes and 14 duplex links (i.e. 28 simplex links). Right: A pattern of paths for an optimal policy enforcement.

environment change and require changes in the policy specification, the agents may still be able to find satisfactory sets of policy enforcement solutions. We look at this effect as *soft policy enforcement* which we believe may provide better control over complex network environments than traditional static policy enforcement implementations.

4. Case Study: Primary and Backup Path Reservation

In this section we perform a policy-to-agent-behavior case study of a primary and backup path reservation scenario. The scenario presented is constructed but still realistic and may in the near future become highly relevant.

Figure 2 shows a graph illustrating the Norwegian university IP backbone infrastructure. All links have 155 Mb/s capacity. Node 6, 1, 8 and 9 are the university cities Oslo, Bergen, Trondheim and Tromsø respectively. A team of physicians from the faculties of medicine at the four different universities have decided to establish a virtual research environment based on high quality multimedia conferencing over the IP network. Such multimedia streams requires up to 70 Mb/s capacity thus 70 Mb/s connections from each university to all others are required, i.e. 12 simplex connections with both primary and backup paths. A backup path is used only if the corresponding primary path has a failure.

4.1 Establishing Policies

The following functional policies are specified with the intention of enabling the desired transport service. Policy targets are emphasized:

Policy F1 Reserve required bandwidth capacity in the network for *primary paths for all specified connections* such that no links are overloaded (zero traffic loss).

Policy F2 Reserve required bandwidth capacity in the network for *backup paths for all specified connections* such that when a single link failure occurs, traffic loss due to link overload is minimized .

The functional policies are further refined resulting in a set of process policies with the mappings $F1 \rightarrow \{Pr1\}$ and $F2 \rightarrow \{Pr2, Pr3, Pr4\}$. In the policy descriptions below *contend for capacity* should be interpreted as causing a link overload if all parties sharing a link transmit simultaneously. However, sharing a link does not necessary imply contention for capacity given the link in question has enough total capacity. Further, π_r^m represent a path of rank r for connection m. Rank is either primary (0) or backup (1), i.e. $r \in \{0, 1\}$.

Policy Pr1 Reserve link capacity for connection m's primary path π_0^m such that π_0^m *do not contend for capacity against other primary paths* $\pi_0^n \mid_{\forall n}$ *on any of the links in* π_0^m (i.e. each link in π_0^m must have enough capacity to carry connection m in addition to all other primary paths carried).

Policy Pr2 Reserve link capacity for connection m's backup path π_1^m such that π_1^m *do not contend for capacity against primary paths* $\pi_0^n \mid_{\forall n}$ *on any of the links in* π_1^m (i.e. each link in π_1^m must have enough capacity to carry connection m in addition to all other primary paths carried).

Policy Pr3 Reserve link capacity for connection m's backup path π_1^m such that all links in π_1^m are *disjoint from* links in connection m's *primary path* π_0^m (i.e. no single link failure causes both a primary and backup path to fail).

Policy Pr4 Reserve link capacity for connection m's backup path π_1^m such that π_1^m *do not contend for capacity against connection n's backup path* π_1^n when connection m and n's primary paths π_0^m and π_0^n are not disjoint (i.e. each link shared by π_1^m and π_1^n must have enough capacity to carry both connection m and n should a link shared by π_0^m and π_0^n fail).

Policy Pr1 ensures no overload (zero traffic loss) when primary paths are in use. Policy Pr2, Pr3 and Pr4 ensures minimum traffic loss due to overload when a single link failure occurs. In Figure 2 a pattern of paths is given which enforces all policies, i.e. an optimal policy implementation.

4.2 Implementing Policies

To implement soft policy enforcement the network environment is mapped onto a graph structure such that vertices represent network nodes and edges represent link resources. Unique types of ant-like agents are allocated to the policy-target instances. Two classes of ant-like agents are created, one class to search for primary paths, and one for backup paths. Primary agents/paths are of rank 0, and backup agents/paths of rank 1. Each of the 12 connections are allocated one species. Every species has both primary and backup agents, thus a total of 24 different types of ant-like agents are implemented each with a unique pheromone type and a specific policy target for which to find an enforcement solution.

Taken from policy F1 and F2, link overload is chosen as the cost measure for a link. Policies Pr1-Pr4 require link capacity to be reserved. Due to limited resources, reservations initiated by two policies may result in an overloaded link, i.e. a policy conflict is experienced. As an attempt to resolve such conflicts agents are made to detest pheromones related to alien paths. Combining the cost measure and pheromone detestation the following link cost expression is derived:

$$L_{ij}^{mr} = \mathcal{S}\left[a_m + \sum_{\forall ns:\, ij \in \pi_s^n} P_{ij}^{ns}\, V_i^{ns}\, Q_{mr}^{ns}\, a_n - c_{ij}\right]$$

where $L_{ij}^{mr}$is the expected potential link overload on link ij if the link is included in connection m's path of rank r. The function, terms and factors of the expression can be explained and related to policies as follows:

- To avoid negative cost values and smoothen the transmission between no loss and loss, a shaping function

$$\mathcal{S}[c] = \begin{cases} \eta \cdot e^{\frac{c}{e \cdot \eta}}, & c < \eta \cdot e \\ c, & otherwise \end{cases}$$

  is applied to the link cost expression. η is a parameter ($\mathcal{S}[0] = \eta$).

- a_m, a_n and c_{ij} represent capacities required by connection m and n, and capacity available on link ij respectively. The sum of all required capacities less available capacity $[a_m + \sum_n a_n - c_{ij}]^+$ equals link overload, thus implements the quality measure specified in policies F1 and F2.

- P_{ij}^{ns} and V_i^{ns} are approximate probabilities. P_{ij}^{ns} indicates the probability that agent of rank s for connection n (agent ns) will include link ij in its solution (i.e. follow ij during search) given that it visits node i. V_i^{ns} indicates the probability that agent ns will include node i in its solution, i.e. visit i during search. Hence P_{ij}^{ns} and V_i^{ns} represent pheromones for agent ns and weight the alien capacity request a_n according to how likely it is to be used when the search process converges. P_{ij}^{ns} and V_i^{ns} implement detestation and enforcement of policies Pr1-Pr3.

- Q^{ns}_{mr} is a weight function controlling the level of detestation

$$Q^{ns}_{mr} = \begin{cases} \widehat{Q}^{n,s-1}_{m,r-1}, & n \neq m,\ s = r = 1 & \text{– the likelihood that path } ns \text{ and } mr \text{ do not have disjont primary paths, i.e. enforcement of Pr4} \\ 1, & n \neq m,\ s = r = 0 & \text{– i.e. no influence by } Q^{ns}_{mr} \\ 20, & n = m,\ s < r = 1 & \text{– strong detestation and enforcement of Pr3} \\ 5, & n \neq m,\ s < r = 1 & \text{– medium detestation and enforcement of Pr2} \\ 0, & otherwise & \end{cases}$$

Finally the cost $L(\pi^m_r)$ of a path π^m_r is made the sum of the costs of all links in the path

$$L(\pi^m_r) = \sum_{ij \in \pi^m_r} L^{mr}_{ij}$$

The sum of all link cost is an approximation of the desirable path cost measure indicated by policies F1 and F2. Optimally only the cost of the link in the path responsible for the largest loss should represent the path cost. However such maximization violates the requirement of additivity in the cost function for CE algorithms (chap. 5 [Rub99]). The sum of link cost is a conservative approximation thus no infeasible solutions are included in the search space by this.

4.3 Simulation results

Table 1 shows simulation results for the scenario. All values are averaged over 12 simulations and standard deviations are given in brackets. The first column indicates which policy the results in the last three columns are related to. The second column gives a short description of the policy. The third column shows the average number of incidents causing policy enforcement failures. The fourth column shows expected relative permanent loss (in percent of total traffic load) due to failed policy enforcements, and the last column shows expected relative loss due to failed policy enforcements given a single link failure. The last row of Table 1 shows totals, both overall expected relative loss including and excluding permanent loss.

In total 12 primary and 12 backup paths are established in each simulation. Solutions for enforcement of policy Pr1 are found in 11 out of 12 simulations, i.e. only one incident over all simulations is observed where 3 primary paths share a link and traffic loss of 55 Mb/s is experienced. In the case of policy Pr2, 3 incidents of enforcement failures are on average observed, and given a single link failure the expected relative loss due to such incidents is on average $1.179\,\%$. For the two last process policies, Pr3 and Pr4, 0.9 incidents of failed enforcement is observed on average, and a relative loss of around $0.2\,\%$ is expected given a single link failure.

Summarized, the results from Table 1 give encouraging indications that ant-like agents can produce policy enforcement implementations of high quality. In our exam-

Table 1. Results averaged over 12 simulations.

Policy	Description	No of incidents of failed enforcement	Expected relative loss (%)	
			Permanent	On single link failure
Pr1	Primary v.s. primary path	0.08 ($\pm$0.29)	0.504 ($\pm$1.816)	0.504 ($\pm$1.816)
Pr2	Primary v.s. alien backup path	3.0 ($\pm$1.15)	0	1.179 ($\pm$0.773)
Pr3	Primary v.s. own backup path	0.9 ($\pm$1.25)	0	0.229 ($\pm$0.347)
Pr4	Backup v.s. none-disjoint primary paths	0.9 ($\pm$1.14)	0	0.162 ($\pm$0.241)

Totals, given a single link failure:	Incl. permanent loss	Excl. permanent loss
Expected relative loss (%)	2.074 ($\pm$1.560)	1.570 ($\pm$0.770)

ple, implementations are produced enabling close to full enforcement of all policies. Given a single link failure an overall loss of 2% of total traffic is expected. However there is still room for improvement especially considering that at least one optimal enforcement implementation exist (no loss given a single link failure) as given in Figure 2.

The poorest results are experienced for policy Pr2. We suspect one reason could be the lack of an additional policy which specifies what strategy the allocation process for primary paths should have when encountering well established alien backup paths, i.e. a policy similar to Pr2 but with opposite roles for primary and backup path. Introducing such a policy would imply having primary agents detest backup agents as well as opposite, which again should result in overall less contention for capacity.

5. Conclusion

In this paper we have presented a distributed management approach which is based on cooperating (and competing) simple mobile agents forming a swarm intelligent policy management system. We have described a design process where a policy specification is transformed into an ant-like agent optimization system, capable of finding enforcement solutions for the policies. The agent system's ability to resolve policy conflicts is termed *soft policy enforcement*. Results from a simulation scenario indicate that near optimal enforcement solutions can by found by the agent system. However no guaranty for finding optimal solutions can be given.

Ongoing work include large scale pheromone management. When embedding swarm based policy enforcement in a large network environment, care must be taken to avoid overloading nodes with pheromone data. Future work should include formalizing the design process as well as further testing of the soft policy enforcement scheme in dynamic network environments.

References

[BCD$^+$00] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. RFC2748: The COPS (Common Open Policy Service) Protocol. IEFT, January 2000.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[CLN00] Jan Chomicki, Jorge Lobo, and Shamin Naqvi. A logic programming approach to conflict resolution in policy management. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 121–132, San Francisco, 2000. Morgan Kaufmann.

[Dam02] Nicodemos C. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Departement of Computing, February 2002.

[DG97] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computing*, 1(1), April 1997.

[HW01] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science 220*, pages 671–680, 1983.

[LS99] E. Lupu and M. Sloman. Conflicts in Policy-based Distributed Systems Management. *IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management*, 25(6):852–869, Nov. 1999.

[MC93] M. J. Maullo and S. B. Calo. Policy Management: An Architecture and Approach. In *Proceedings of the IEEE First International Workshop on Systems Management, 1993*, pages 13 –26, UCLA, California, April 1993.

[MESW01] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. RFC3060: Policy Core Information Model - Version 1 Specification. IETF, February 2001.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[Rub01] Reuven Y. Rubinstein. *Stochastic Optimization: Algorithms and Applications*, chapter Combinatorial Optimization, Cross-Entropy, Ants and Rare Events - Section 7: Noisy Networks. Kluwer Academic Publishers, 2001.

[SA94] Simon Steward and Steve Appleby. Mobile Software Agents for Control of Distributed Systems Based on Principles of Social Insect Behavior. *BT Technology Journal*, 12(2):104–113, April 1994.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[Slo94] Morris S. Sloman. Policy Driven Management for Distributed Systems. *Journal of Network and Systems Management*, 2(4):333–360, 1994.

[VS99] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[WH02a] Otto Wittner and Bjarne E. Helvik. Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks. In *The Third International Workshop on Ant Algorithms, ANTS'2002*, Brussels, Belgium, Sept 2002.

[WH02b] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.

PAPER E

Scalable Distributed Discovery of Resource Paths in Telecommunication Networks using Cooperative Ant-like Agents

Otto Wittner, Poul E. Heegaard and Bjarne E. Helvik

Congress on Evolutionary Computation, CEC 2003

December 8 - 12th, Canberra, Australia.

SCALABLE DISTRIBUTED DISCOVERY OF RESOURCE PATHS IN TELECOMMUNICATION NETWORKS USING COOPERATIVE ANT-LIKE AGENTS

Otto Wittner

Poul E. Heegaard

Bjarne E. Helvik

Abstract Future user controlled development of telecommunication services combined with powerful terminal equipment results in many heterogenous services running in a peer-to-peer execution environment. Locating a desired service in such an environment is challenging. In this paper a swarm based optimization algorithm is presented which is capable of finding paths of resources in a complex network environment. The algorithm is fully distributed and may be implemented using simple ant-like mobile agents. On the contrary to existing localization mechanisms for peer-to-peer systems the algorithm considers all accessed resources between (and including) the client side and server side when a resource path is evaluated. Scalability is achieved by making agents cooperate during search when they have overlapping search profiles. Results from simulations are promising. The expected cooperative behavior is shown to be present, i.e. a set of near optimal resource paths conforming to a set of different but overlapping search profiles may be found with improved performance.

Keywords: Telecommunications, distributed multi-criteria optimization, resource paths, swarm intelligence, ant-like agents, peer-to-peer.

1. Introduction

Recent development in terminal and core network technologies have opened for realization of a range of new telecommunication services.[1] One new service category

[1]To enable rapid realization of new heterogenous services, Telenor, Ericsson and NTNU have chosen to move from the traditional call centric approaches to a more service centric approach as well as involve users and user innovation in the service development process [San02]. Peer-to-peer is adopted as a potential environment for distributed service execution. The overall initiative has resulted in the AVANTEL project [San03]. This paper presents results form research sponsored by the AVANTEL project. This work was also partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).

is peer-to-peer systems where users develop and provide services to other users with little or no centralized management. Locating specific services in such systems is challenging.

Many directory systems for peer-to-peer environments have been developed during the last decade [MKL$^+$02]. Common for these systems are limited functionality for specifying quality of service (QoS) parameters in service lookup requests, which often result in uninteresting service offers. For instance access to a high quality multi-media stream may be offered but due to lack of network bandwidth the stream becomes uninteresting.

A class of bio-inspired algorithms known as *swarm intelligence* systems [BDT99] are potentially robust and may scale well due to their use of distributed autonomous components known as agents. Swarm intelligence has successfully been applied to a range of optimization problems [DC99], some in the domain of telecommunications [SHBR97, CD98, WBP98, VS99b]. In this paper we present a *swarm intelligence* based algorithm which enables implementation of improved QoS controlled service lookup and access. The algorithm seeks to find a *path of resources* from a client terminal to a service providing server such that all resources in the path conforms (as well as possible) with constraints and preferences of a request profile specified by the user.

Our algorithm is based on work previously published in [HW01, WH02a, WH02b]. Scalability in terms of number of parallel tasks handled by agents has so far only been addressed to a limited extent. In this paper we introduce mechanisms to manage large scale use of the algorithm. On the assumption that service request profiles in many cases will be overlapping, we let agents share information about the overlapping parts of the profiles. Assuming a limited total number of possible constraints and preferences the new sharing strategy reduces the total amount of storage space required for pheromones to a manageable level, and increases the search efficiency.

The remainder of this paper has four sections. Section 2 presents background information, terminology and formalisms. Section 3 introduces the behavior foundations for the agents, describes cost functions, presents reformulations and additions required to realize extended pheromone sharing between agents, and describes the new agent algorithm. Section 4 describes our experimental setup and reports and discusses simulation results. Finally, Section 5 summarizes and indicates future work.

2. Resource Paths and Profiles

In this paper we view all components in a network environment as resources with individual *profiles*, i.e. service components (created by users or operators) as well as links and network nodes for network transport are viewed as resources with a related profile. An ordered sequence of resources are denoted a *resource path*.

2.1 AMIGOS

The motivation for adopting a resource view is the heterogeneity of the expected network environment where one of the AVANTEL project's root services, Advanced

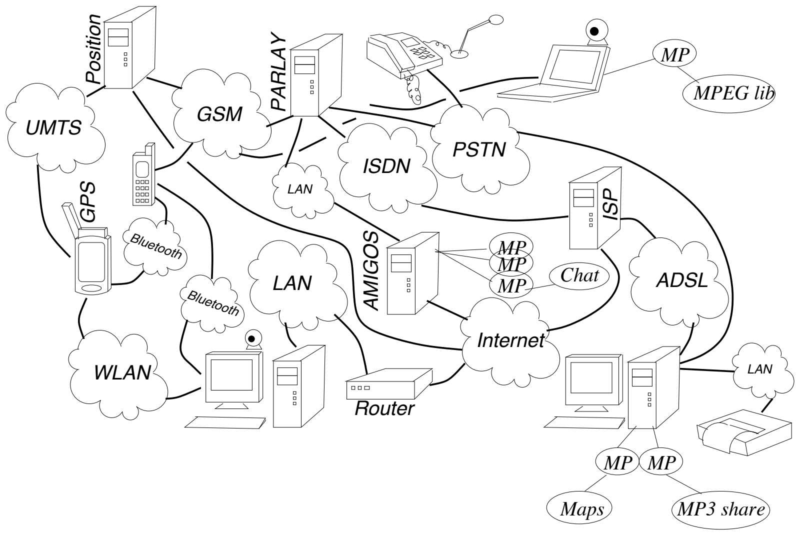


Figure 1. A example network environment where *AMIGOS* is expected to run.

Multimedia In Group Organized Services (AMIGOS) [San02], is running. AMIGOS provides the basic functionality required for users to manage and visit a *Meeting Place* (MP). An MP is a user (or operator) composed and configured telecommunication service providing a connection point between a specific set of users. A meeting place may also act as a repository for multimedia objects to be shared between the users visiting the meeting place.

Figure 1 illustrates the expected heterogeneity of the environment where the AMIGOS service will be running. Software processes provide MP service resources which again provide access to multimedia objects. A variety of terminals controlled by different users and a mix of servers own by different operators provide processing power. And finally, a range of transmission technologies provide transmission links interconnecting terminals and servers.

2.2 Resource Paths

In the AMIGOS environment a resource is denoted ω. When users are active they will access a set of resources. We denote such an ordered set of resources a *resource path*

$$\pi = \{\omega_0, \omega_1, \cdots, \omega_{N_\pi - 2}, \omega_{N_\pi - 1}\} \tag{1}$$

where N_π is the number of resources in the path.

We classify resources into two categories: *Transport* and *Peripheral* resources. A *peripheral resource* is the last resource $\omega_{N_\pi - 1} \in R_p$ in a path and provides some value added service. Users will normally desire to access a specific type of peripheral resource, e.g. an MP, multimedia libraries etc. R_p is the set of all peripheral resources. A *transport resource* is an intermediate resource $\omega_i \in R_t$, where

$i = 0 \dots N_\pi - 2$. Such a resource, e.g. a link, provides transport which contributes in enabling access to a peripheral resource. R_t is the set of all transport resources.

In general a resource path π will contain a sequence of transport resources and a single peripheral resource. Thus $\pi \in \Omega$

$$\Omega = \{\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}, \omega_{N_\pi-1}\} : \begin{array}{l} \omega_{0\dots N_\pi-2} \in R_t, \\ \omega_{N_\pi-1} \in R_p\} \end{array}$$

is the set of all resource paths, which we denote the *search space* for user request profiles (see next section).

A common limitation of today's lookup services for peer-to-peer systems is that transport resource limitations are not taken into account during search, i.e. $\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}\}$ is ignored. Our algorithm takes into account the complete path of resources.

2.3 Profiles and QoS Objectives

In the AMIGOS environment, users, terminals and services are expected to have individual *profiles* containing QoS parameters. There are two classes of profiles.

2.3.1 User Request Profile

A user request profile $\bar{\zeta}_r(k)$ refer to a set of QoS parameters specifying constraints and preferences, i.e. *QoS objectives,* for a request r from user k. In general, a user request profile may refer to a large set of QoS parameters. To maintain scalability, we have defined a finite and ordered set Ξ of QoS parameters from which a specific request profile may be constructed. Each parameter ξ_i in Ξ, where $i = 1, ..., |\Xi|$, is a specific QoS requirement. A user request profile may now be expressed as $\bar{\zeta}_r(k) = \{\alpha_1, \dots, \alpha_{|\Xi|}\}$. In this investigation, a binary user request profile is used, i.e.,

$$\alpha_i = \left\{ \begin{array}{lcl} 1 & : & \text{when requirement } i \text{ should be met} \\ 0 & : & \text{otherwise} \end{array} \right.$$

however in general (Section 3.3) arbitrary values $\alpha_i \geq 0$ may be used to balance the importance of the various requirements.

In the cases where we have a range of QoS parameters of the same kind to choose from, e.g. delays and bandwidths, only one α-value in the range should be set.

2.3.2 Resource Profile

When used, a resource ω_x may introduce QoS impairments with respect to the QoS requirements of a user request r_k. These impairments may for instance be excessive delays, limited bandwidth, processing or storage capacity, or lack of required peripheral equipment, services or information. Impairments are denoted the *loss* $\ell_i(\omega_x)$ introduced by resource ω_x with respect to a QoS requirement i. Hence, the resource profile $\bar{\zeta}(\omega_x)$ associated with resource x is represented as a loss vector

$$\bar{\zeta}(\omega_x) \equiv \bar{\ell}(\omega_x) = \{\ell_1(\omega_x), \ell_2(\omega_x), \dots, \ell_{|\Xi|}(\omega_x)\}$$

where $\ell_i(\omega_x) \in \mathbb{R}^+$. *Resource profile* and *loss vector* are used interchangeably throughout the rest of this paper. Examples of how the various loss elements $\ell_i(\omega_x)$ may be determined are presented in Section 4.

3. Agent Behavior

Our search algorithm is based on *swarm intelligence* [BDT99] and mimics the foraging behavior of ants. It uses a high number of agents with simple behaviors, and generates one *species* of agents (i.e. one type of agent) for every user request. Agents of the same species have the only mission of searching for resource paths conforming with the criteria given by the profile of a specific user request. Multiple species of agents may search in parallel.

Our algorithm provides a general method for generating solutions to *combinatorial multi-criteria optimization problems* (CMCO problems). A few CMCO systems based on swam intelligence [MM99, GGP02, IMM01] exist. Most of these build on Dorigo & al.'s *Ant Colony Optimization* system [DC99] which requires centralization and batch oriented operations to generate solutions efficiently. Our algorithm however, is fully distributed with no central control component.

3.1 Foundations

The concept of using multiple agents with a behavior inspired by foraging ants to solve problems in telecommunication networks was introduced by Schoonderwoerd & al. in [SHBR97] and further developed in [CD98, WPO98, Sch00]. Schoonderwoerd & al.'s relates to Dorigo & al.'s work on Ant Colony Optimization (ACO) [DG97]. The overall idea is to have a number of simple ant-like mobile agents search for paths between source and destination nodes. While moving from node to node in a network, an agent leaves markings resembling the pheromones left by real ants during ant trail development. This results in nodes holding a distribution of pheromone markings pointing to their different neighbor nodes. An agent visiting a node uses the distribution of pheromone markings to select which node to visit next. A high number of markings pointing towards a node (high pheromone level) implies a high probability for an agent to continue its itinerary towards that node. Using trail marking agents together with a constant evaporation of all pheromone markings, Schoonderwoerd and Dorigo show that after a relatively short period of time the overall process converges towards having the majority of the agents follow a single trail. The trail tends to be a near optimal path from the source to the destination.

3.1.1 The Cross Entropy Method

In [Rub99] Rubinstein develops a centralized search algorithm with similarities to Ant Colony Optimization [DC99, ZBMD00]. The total collection of pheromone markings in a network at time t is represented by a probability matrix P_t where an element $P_{t,rs}$ (at row r and column s of the matrix) reflects the normalized intensity of pheromones pointing from node r towards node s. An agent's stochastic search for a sample path resembles a Markov Chain selection process based on P_t.

In a large network with a high number of feasible paths with different qualities, the event of finding an optimal path by doing a random walk (using a uniformly distributed probability matrix) is rare, e.g. the probability of finding the shortest Hamiltonian cyclic path (the Traveling Salesman Problem) in a 26 node network is $\frac{1}{25!} \approx 10^{-26}$. Thus Rubinstein develops his algorithm by founding it in rare event theory.

By importance sampling in multiple iterations Rubinstein alters the transition matrix ($P_t \rightarrow P_{t+1}$) and amplifies certain probabilities such that agents eventually find near optimal paths with high probabilities. Cross entropy (CE) is applied to ensure efficient alteration of the matrix. To speed up the process further, a performance function weights the path qualities (two stage CE algorithm [Rubar]) such that high quality paths have greater influence on the alteration of the matrix. Rubinstein's CE algorithm has 4 steps:

1 At the first iteration $t = 0$, select a start transition matrix $P_{t=0}$ (e.g. uniformly distributed).

2 Generate N paths from P_t. Calculate the minimum Boltzmann temperature γ_t to fulfill average path performance constraints, i.e.

$$\min \gamma_t \text{ s.t. } h(P_t, \gamma_t) = \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) > \rho \tag{2}$$

where

$$H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}}$$

is the performance function returning the quality of path π_k. $L(\pi_k)$ is the cost of using path π_k (see Section 3.2 and 3.3). $10^{-6} \leq \rho \leq 10^{-2}$ is a search focus parameter. The minimum solution for γ_t will result in a certain amplification (controlled by ρ) of high quality paths and a minimum average $h(P_t, \gamma_t) > \rho$ of all path qualities in the current batch of N paths.

3 Using γ_t from step 2 and $H(\pi_k, \gamma_t)$ for $k = 1, 2..., N$, generate a new transition matrix P_{t+1} which maximizes the "closeness" (i.e. minimizes distance) to the optimal matrix, by solving

$$\max_{P_{t+1}} \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) \sum_{ij \in \pi_k} \ln P_{t,ij} \tag{3}$$

where $P_{t,ij}$ is the transition probability from node i to j at iteration t. The solution of (3) is shown in [Rub99] to be

$$P_{t+1,rs} = \frac{\sum_{k=1}^{N} I(\{r, s\} \in \pi_k) H(\pi_k, \gamma_t)}{\sum_{l=1}^{N} I(\{r\} \in \pi_l) H(\pi_l, \gamma_t)} \tag{4}$$

which will minimize the cross entropy between P_t and P_{t+1} and ensure an optimal shift in probabilities with respect to γ_t and the performance function.

4 Repeat steps 2-3 until $H(\widehat{\pi}, \gamma_t) \approx H(\widehat{\pi}, \gamma_{t+1})$ where $\widehat{\pi}$ is the best path found.

3.1.2 Distributed Cross Entropy Method

In [HW01] a distributed and asynchronous version of Rubinstein's CE algorithm is developed. By a few approximations, (4) and (2) may be replaced by autoregressive counterparts based on

$$P_{t+1,rs} = \frac{\sum_{k=1}^{t} I(\{r,s\} \in \pi_k)\beta^{t-k} H(\pi_k, \gamma_t)}{\sum_{l=1}^{t} I(\{r\} \in \pi_l)\beta^{t-l} H(\pi_l, \gamma_t)} \tag{5}$$

and

$$\min \gamma_t \text{ s.t. } h_t^{'}(\gamma_t) > \rho \tag{6}$$

where

$$\begin{aligned} h_t^{'}(\gamma_t) &= h_{t-1}^{'}(\gamma_t)\beta + (1-\beta)H(\pi_t, \gamma_t) \\ &\approx \frac{1-\beta}{1-\beta^t} \sum_{k=1}^{t} \beta^{t-k} H(\pi_k, \gamma_t) \end{aligned}$$

and where $\beta \in \langle 0, 1 \rangle$ controls the history of paths remembered by the system (i.e. replaces N in step 2). Step 2 and 3 in the algorithm can now be performed immediately after a single new path π_t is found, and a new probability matrix P_{t+1} can be generated.

The distributed CE algorithm may be viewed as an algorithm where search agents evaluate a path found (and calculate γ_t by (6)) right after they reach their destination node, and then immediately return to their source node backtracking along the path. During backtracking pheromones are placed by updating the relevant probabilities in the transition matrix, i.e applying $H(\pi_t, \gamma_t)$ through (5).

The distributed CE algorithm resembles Schoonderwoerd & al.'s original system as well as Dorigo & al.'s AntNet system [CD98]. However, none of the earlier systems implements a search focus stage (the adjustment of γ_t) as in the CE algorithms [Rubar]. The search focus stage ensures fast and accurate convergence without having to introduce search focus heuristics as is typically required in ACO systems.

3.2 Cost Functions

Cost functions applied in this paper output a measure for the level of QoS loss introduced by non-conformance between a specific QoS parameter in a user request and service capabilities of a sequence of resources in a resource path.

Recall that Ω is denoted the *search space* of user requests. In this section we use *solution* and *resource path* interchangeably. Both indicate elements in a relevant search space.

3.2.1 Constraints and Ordering

The set of valid QoS parameters Ξ is divided into two subsets denoted *Constraint* and *best-value* parameters. Constraint parameters, of the set denoted Ξ_C, require a service to have a level of quality within a specific range, i.e. a minimum and/or a maximum acceptable QoS level is specified. Best-value parameters, of the set denoted Ξ_B, indicate only that better QoS levels are preferred (no upper or lower limits are given).

To be able to compose an overall cost function which measure impairments to constraint and best-value QoS parameters, we require two classes of terms in the function: *Implicit constraint checks* and *solution ordering* terms. Implicit constraint checks handle constraint QoS parameters and provide a rough sorting of the search solutions in feasible and infeasible solutions. Solution ordering terms handle best-value QoS parameters and enable a detailed ordering of the candidate solutions found.

To realize the two classes of terms, we define two support functions. Both functions perform rough normalization by mapping values onto the $[0.5,\ 1]$. Thus values of very different original scale may be compared and/or summarized. To enable *solution ordering* on a common scale we define

$$h(y) = \frac{1}{1 + e^{-\eta \cdot y}}, \quad y \geq 0 \tag{7}$$

which normalize any positive real value y to the range $\langle 0.5,\ 1]$ where η is a general scaling parameter. Further we define

$$u(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0.5 & \text{otherwise} \end{cases} \tag{8}$$

which maps any real value z into 1 or 0.5, i.e. the upper and lower limits of the range of $h(y)$. Hence $u(z)$ may work as an *implicit constraint check* by introducing a normalized penalty when undesirable QoS loss is experienced.

3.2.2 Search Space Smoothness and Overall Cost

Since our algorithm is of a stochastic nature and is based on cross entropy, good performance is ensured by making the search space Ω “smooth”, i.e. ensure that Ω contains a wide range of resource paths of different qualities. To realize smoothness we enforce additivity (as shown efficient in Section 5.1 of [Rub99]) when deriving an overall quality measure for a resource path.

Hence, during a search for a resource path π, we accumulated an *overall* loss vector $\bar{L}(\pi)$ with one cost value for each QoS parameter specified.

$$\bar{L}(\pi) = \{L_1(\pi), L_2(\pi), \ldots, L_{|\Xi|}(\pi)\} \tag{9}$$

where additivity is preserved by having

$$L_i(\pi) = \sum_{\omega_x \in \pi} \ell_i(\omega_x)$$

where $i = 1, \dots, |\Xi|$, and hence

$$\bar{L}(\pi) = \sum_{\omega_x \in \pi} \bar{\ell}(\omega_x) \tag{10}$$

Further to create an overall cost measure for the QoS loss with respect to the requirements, we summarize all elements in $\bar{L}(\pi)$ and produce a scalar cost $L(\pi)$. Since the different elements in $\bar{L}(\pi)$ may relate to very different QoS parameters, we apply our support functions (7) and (8) appropriately to normalize and give correct focus to the different elements. Let

$$L_i^*(\pi) = \begin{cases} u(L_i(\pi)), & \xi_i \in \Xi_C \\ h(L_i(\pi)), & \xi_i \in \Xi_B \end{cases} \tag{11}$$

where $i = 1, \dots, |\Xi|$ and

$$\bar{L}^*(\pi) = \{L_1^*(\pi), L_2^*(\pi), \dots, L_{|\Xi|}^*(\pi)\}$$

The overall scalar cost of a resource path becomes

$$\begin{array}{rcl} L(\pi) & = & \bar{L}^*(\pi) \cdot \bar{\zeta}_r(k) = \sum_{i=1}^{|\Xi|} \alpha_i L_i^*(\pi) \\ & = & \sum_{i \in \Xi_C} \alpha_i u(L_i(\pi)) + \sum_{j \in \Xi_B} \alpha_j h(L_j(\pi)) \end{array} \tag{12}$$

where, as presented in Section 2.3, the various α_i specify the QoS parameters of the user request profile $\bar{\zeta}_r(k)$.

The rationale for accumulating all loss values in the vector $\bar{L}(\pi)$ during a search and not use (12) directly, is to enable efficient collection and dissemination of QoS information through pheromone values in the nodes.

3.3 Path Quality Vectors

As mention in Section 2.3.1, our algorithm ensures scalability by taking advantage of the assumption that only a limited set of unique QoS parameters are available for building request profiles. In earlier published work [HW01, WH02a, WH02b] the distribute CE algorithm allocated one unique pheromone type to every agent species in operation. Reapplying this allocation strategy would mean one unique pheromone type for every user request. In the AMIGOS environment the number of unique user requests to be managed may be very large. Allocating unique pheromones will result in a large amount of pheromone data to be managed by each resource in the network and a need for a large number of agents per species to ensure convergence towards good solutions in reasonable time.

To manage scalability we make different agent species cooperate (on the contrary to work in [WH02b]) in updating a shared set of pheromone values. Instead of a unique identity for each user request (and a corresponding agent species), we construct a vector containing an element for each QoS parameter in the user request profile.

By controlling the total number of unique QoS parameters $|\Xi|$ available, we can limit the number of unique pheromones required. Note that the total number of possible unique profiles $N_{\bar{\zeta}}$ will still be large,

$$N_{\bar{\zeta}} = 2^{|\Xi|} - 1$$

e.g. to enable a total of $N_{\bar{\zeta}} = 10^{100}$ different profiles only $|\Xi| \approx 333$ unique pheromones are required[2]. In reality less than $N_{\bar{\zeta}}$ profiles will be valid since (as mentioned in Section 2.3.1) several QoS parameters will be mutually exclusive, e.g. "max. delay = 70 ms" excludes "max. delay = 80 ms".

The basis for generating pheromones is cost values output from the cost functions described in the previous section. Now recall the algorithmic step calculating the temperature from (6). The existence of a unique pheromone for each QoS parameter implies that a separate temperature parameter γ_t must be calculated for each cost value. Thus two vectors, one with temperatures and one with performance values, are generated by applying (6) for each cost value $L_i^*(\pi_k)$ found in (11) :

$$\begin{aligned} \bar{\gamma}_t &= \left\{\gamma_{t,1},\, \gamma_{t,2}, \dots,\, \gamma_{t,|\Xi|}\right\} \\ \bar{H}(\pi_k, \bar{\gamma}_t) &= \begin{array}{l} \{H_1(\pi_k, \gamma_{t,1}),\, H_2(\pi_k, \gamma_{t,2}), \\ \dots,\, H_{|\Xi|}(\pi_k, \gamma_{t,|\Xi|})\} \end{array} \end{aligned} \tag{13}$$

where

$$H_i(\pi_k, \gamma_{t,i}) = e^{-\frac{L_i^*(\pi_k)}{\gamma_{t,i}}}$$

We also calculate γ_t by (6) where

$$H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}} \tag{14}$$

for reasons described below.

Path backtracking and pheromone placing activities performed by agents, i.e. step 3 of the algorithm from Section 3.1.1, now generate an updated vector of probability distributions, i.e. (5) becomes

$$\bar{P}_{t+1,rs} = \frac{\sum_{k=1}^{t} I(\{r,s\} \in \pi_k)\beta^{t-k}\bar{H}(\pi_k, \bar{\gamma}_t)}{\sum_{l=1}^{t} I(\{r\} \in \pi_l)\beta^{t-l}\bar{H}(\pi_l, \bar{\gamma}_t)} \tag{15}$$

During forward search however the agents require P_{t+1} to select which node to visit next (step 2 from Section 3.1.1). P_{t+1} is the probability matrix built from the over all scalar cost measure $L(\pi_k)$ applied in (14). By (14) and (12) we have

$$H(\pi_k, \gamma_t) = \prod_{i=1}^{|\Xi|} e^{-\frac{\alpha_i L_i^*(\pi_k)}{\gamma_t}} = \prod_{i=1}^{|\Xi|} H_i(\pi_k, \gamma_{t,i})^{\alpha_i \frac{\gamma_{t,i}}{\gamma_t}} \tag{16}$$

[2]By introducing non-deterministic requirements, i.e. weighting of alternatives by having $\alpha_i \in \langle 0, 1\rangle$, the profile space becomes even richer/larger.

By carrying $\bar{\gamma}_t, \gamma_t$ and the request profile $\bar{\zeta}_r(k) = \{\alpha_1, \ldots, \alpha_{|\Xi|}\}$ agents can generate P_{t+1} during forward search. For each node r visited, an agent requests (13) from the node, calculates (16) and produces $P_{t+1,r}$ for all neighbor nodes s by (15).

3.4 Discovery of a Destination Node

Recall that an agent builds a path by a stochastic process. After adding one resource r to the path, the next resource to be included is selected (using the probability vector $P_{t,r}$) from the set of neighbor resources of r. Since a user request profile $\bar{\zeta}_r(k)$ do not specify a specific destination resource, we have introduced loopback links as an aid to identify potential destination nodes. All peripheral resources are connected to themselves by a link resource ω_{LB} where $\bar{\zeta}(\omega_{LB}) \equiv \bar{0}$. If an agent traverses such a loopback link resource, and visits the same peripheral resource twice in a row, the peripheral resource is selected as the agents destination resource, and the path found is considered complete.

Introducing loopback link resources is equivalent to adding one unique destination resource ω_D to the environment and connecting all peripheral resources to ω_D by zero cost link resources (similar to ω_{LB}).

3.5 Agent Algorithm

The most intuitive implementation of our algorithm requires two types of components: *mobile agents* and *network nodes running mobile agent platforms* [PK98]. The nodes are only required to provide storage for pheromone values $\bar{H}(\pi_t, \bar{\gamma}_t)$ with their related autoregressive history variables (see [HW01] for details on autoregression), and basic arrival, departure and execution functionality for the mobile agents. The rest of the algorithm is implemented in agents. An agent roughly performs the following steps after being created at its home resource ω_0 with user request profile $\bar{\zeta}_r(k)$ describing its search mission:

Forward search

1 Clear tabu list containing resources already visited. Clear $\bar{L}(\pi_t)$. Fetch $\bar{\gamma}_t, \gamma_t$, and $\bar{\zeta}_r(k) = \{\alpha_i : \forall i\}$ from the database in resource ω_0.

2 Record visits to current resource ω_x in tabu list. If current resource has been visited twice in a row, proceed with step 1 of *path evaluation and backtracking* (see below).

3 Reconstruct relevant row of P_{t+1} by (16) using $\bar{\gamma}_t, \gamma_t$, and $\bar{\zeta}_r(k)$. Build a next-hop probability table and use the tabu list to avoid revisiting non-peripheral resources.

4 Select next resource ω_{x+1} to visit using the next-hop probability table.

5 Update $\bar{L}(\pi_t)$ by adding $\bar{\zeta}(\omega_{x+1})$, i.e. implement (10).

6 Move to resource ω_{x+1} and loop back to step 2 of *forward search*.

Path evaluation and backtracking

1 Calculate finale path cost values $\bar{L}^*(\pi_t)$ by (11).

2 Increase search focus if required (see [WHH03b] for details on search focus adjustment).

3 Calculate new temperatures $\bar{\gamma}_{t+1}$ and γ_{t+1} using (6), cost values in $\bar{L}^*(\pi_{t+1})$ and (12).

4 Backtrack every hop towards resource ω_0. At each hop update the transition matrix (leave pheromones) by use of (13) and (15)

5 When backtracking has completed, fetch and recalculate the temperatures $\bar{\gamma}_{t+1}$ and γ_{t+1} (in case other agents have updated the database in resource ω_0 while this agent was doing forward search). Store the new values of $\bar{\gamma}_{t+1}$ and γ_{t+1}.

6 Goto step 1 of *forward search* and start a new search.

Minimum one agent is required to complete a search mission specified by a certain user request profile. However, the algorithm allows for many agents to search in parallel.

When the search mission converges, paths found by the agents driven by the same user request profile will appear as paths of high probability values in the transition matrix (i.e. an intense track of pheromones).

4. Experiments

To investigate the performance of our algorithm a simulator has been implemented based on "Network Simulator 2" (NS2)[WH00]. NS2 is an open source simulator package capable of simulating realistic IP-based scenarios [DAR].

4.1 The Test Environment

All our test scenarios described below where run in a test environment based on the illustration in Figure 1. In Figure 2 the same environment is shown as a graph where link bandwidths, link delays and node RPI/RPQs (see next section) of links and nodes are indicated. All nodes with connection degree greater than one were enabled as routing nodes, i.e. they forward traffic destined for other nodes. We have also enabled all nodes to act as peripheral resources, i.e. all nodes have loopback link resources (not shown in Figure 2) as described in Section 3.4.

4.2 QoS Parameters in the Test Environment

In our simulations the user request profiles includes six sets of QoS parameters:

- $\Xi_{RPI} \subseteq \Xi_C$ is the set of all resource profile index (RPI) parameters. An RPI acts as a summary index for some set of resource capabilities. RPIs are relevant only for peripheral resources.

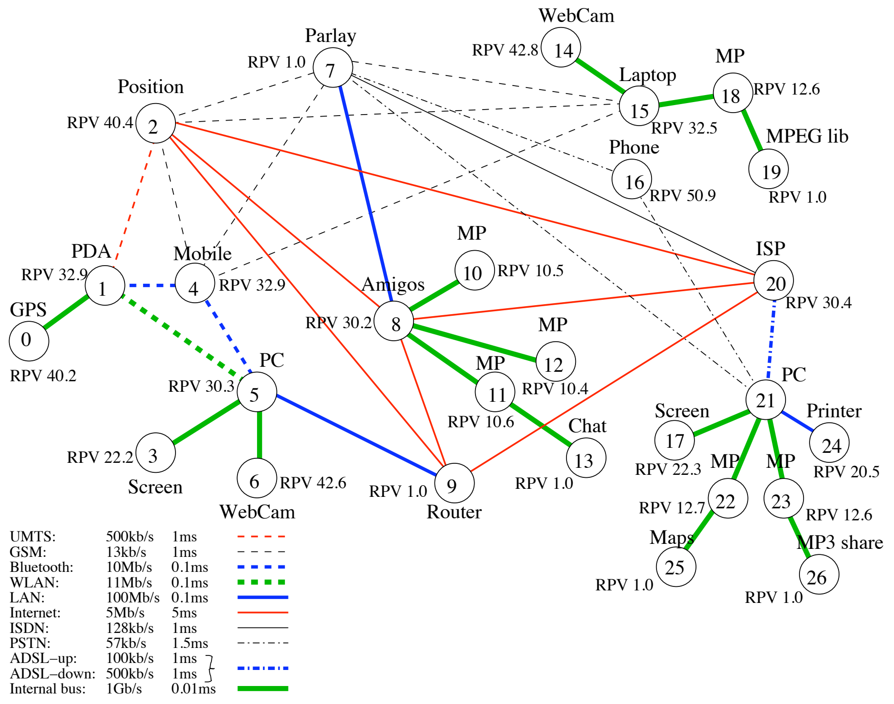


Figure 2. The test environment which is a graph representation of the environment illustrated in Figure 1. An RPV (resource profile value) holds both a node's RPI (resource profile index) and RPQ (resource profile quality). See [WHH03b] for details.

- $\Xi_\delta \subseteq \Xi_C$ is the set of all maximum delay parameters. Only link resources induce delay.
- $\Xi_\beta \subseteq \Xi_C$ is the set of all minimum accepted bandwidth parameters. Only link resources have limited bandwidth.
- $\Xi_{RPQ} \subseteq \Xi_B$ is the set of all resource profile quality (RPQ) parameters. RPQ is the quality index of an RPI.
- $\Xi_{\delta*} \subseteq \Xi_B$ is the set of all expected delay parameters.
- $\Xi_{\beta*} \subseteq \Xi_B$ is the set of all bandwidth utilization parameters.

Three QoS parameters in each of the above parameter sets were used in our simulation scenarios, i.e. $|\Xi| = 18$. See [WHH03b] for a comprehensive description of Ξ.

4.3 Cost functions

By (10) and (11) we may derive the general cost vector applied on a path π in our test scenarios. For all best value QoS parameters we set $\eta = 1$ in (7). The 18 elements of the cost vector $\bar{L}^*(\pi)$ are the following,

$$L_i^*(\pi) = \begin{cases} u(\ell_i(\omega_{N_\pi - 1})) & i \in \{1,2,3\},\ \xi_i \in \Xi_{RPI} \\ u(\sum_{x=0}^{N_\pi - 2} \ell_i(\omega_x)) & i \in \{4,...,9\},\ \xi_i \in (\Xi_\beta \cap \Xi_\delta) \\ h(\ell_i(\omega_{N_\pi - 1})) & i \in \{10,11,12\},\ \xi_i \in \Xi_{RPQ} \\ h(\sum_{x=0}^{N_\pi - 2} \ell_i(\omega_x)) & i \in \{13,...,18\}, \\ & \xi_i \in (\Xi_{\beta^*} \cap \Xi_{\delta^*}) \end{cases}$$

The calculations required to derive a loss value $\ell_i(\omega_x)$ for the resource profile of a resource ω_x implement an interpretation of QoS parameter ξ_i. In our test scenarios we interpret and implement the different QoS parameters described above ($\xi_i \in \Xi$) as follows:

- $\xi_i \in \Xi_{RPI}$, *resource correctness*: Absolute distance between ξ_i and the RPI for a peripheral resource ω_x (RPI_{ω_x}) is returned as the loss value.

$$\ell_i(\omega_x) = |\xi_i - RPI_{\omega_x}| \tag{17}$$

- $\xi_i \in \Xi_{RPQ}$, *resource quality*: The ratio of the distance between requested RPQ value ξ_i and $RPQ_{\omega_x}$of resource ω_x, and the decimal part of ξ_i is returned.

$$\ell_i(\omega_x) = \frac{|\xi_i - RPQ_{\omega_x}|}{\xi_i - \lfloor \xi_i \rfloor} \tag{18}$$

- $\xi_i \in \Xi_\beta$, *bandwidth constraints*: If the capacity B_x of a router-link resource ω_x is below ξ_i, a non-zero contribution equal to the exceeded capacity is returned as loss. Otherwise, zero loss is returned.

$$\ell_i(\omega_x) = [\xi_i - B_x]^+ \tag{19}$$

- $\xi_i \in \Xi_{\beta^*}$, *bandwidth utilization*: The ratio between the minimum bandwidth requirement ξ_i and the router-link capacity B_x is returned.

$$\ell_i(\omega_x) = \xi_i / B_x \tag{20}$$

- $\xi_i \in \Xi_\delta$, *delay constraints*: The delay induced by resource ω_x is denoted Δ_x. We want the cost element $L_i(\pi)$ to represent the difference between the sum of all induced delays and the maximum delay requirement given by ξ_i, thus

$$L_i(\pi) = \left[\sum_{\omega_x \in \pi} \Delta_x - \xi_i \right]^+ \tag{21}$$

The support function (8) normalizes $L_i(\pi)$ (when $\xi_i \in \Xi_\delta$), thus the truncation operator $[\ldots]^+$is redundant. The loss returned may now be expressed by

$$\ell_i(\omega_x) = \Delta_x - \frac{\xi_i}{N_\pi} \tag{22}$$

Table 1. Simulation scenario parameters

Scenarios	No of species	No of ants / species	Client resource (node #)	User request profile $\xi_r(k)$
A	1	12	4	{1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0}
B	1	12	1	———-“———
C	1	12	5	———-“———
D_4		4	4	———-“———
D	3	4	1	———-“———
		4	5	———-“———
E_4		6	4	{**1**,0,0,**1**,0,0,**1**,0,0,**1**,0,0,**1**,0,0,**1**,0,0}
		6	1^*	{0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0}
E	6	6	5^*	{0,1,0,0,0,1,**1**,0,0,0,1,0,0,0,1,**1**,0,0}
		6	21^*	{0,0,1,**1**,0,0,0,1,0,0,0,1,**1**,0,0,0,1,0}
		6	16^*	{**1**,0,0,0,0,1,0,0,1,**1**,0,0,0,0,1,0,0,1}
		6	15^*	{0,0,1,0,1,0,0,0,1,0,0,1,0,1,0,0,0,1}

where N_π is the number of resources in path π. Note that knowledge of the complete path π is required to calculate (22), hence in practice $\ell_i(\omega_x)$ will return only Δ_x and further calculations, i.e. (21), are postponed until the agent reaches the peripheral resource of path π.

- $\xi_i \in \Xi_\delta$, *expected delay*: The ratio between the sum of induced delays by all resources in the path and ξ_i is returned.

$$\ell_i(\omega_x) = \frac{\Delta_x}{\xi_i} \tag{23}$$

4.4 Scenarios

To verify that pheromone sharing results in cooperative behavior (and not interference or disturbance) between agent species, we created five simulation scenario denoted A,B,C,D and E. The following sections describe two test cases using the five scenarios.

4.4.1 Full Overlap in Profiles

The first four scenarios (A-D) test cooperation between three agent species. Table 1 shows the parameters in use for the scenarios. Scenario A, B and C are similar. One agent species search for resource paths by applying a specific user request profile. The same request profile is applied by A, B and C, i.e. full overlap in profiles, however search is initiated from three different client resources. In scenario D three species search simultaneously. They all still apply the same user request profile as in A, B and C. For all scenarios the total of agents reading and updating a relevant QoS parameter is 12.

Table 2. Simulation results

Scenario	Client resource	Average convergence time		Average final path cost		Best final path cost		Final equals preferable path	
	(node #)	Mean	Stdev	Mean	Stdev	Cost	# sim	# sim	$\lvert final - preferable \rvert$
A	4	205.3	(166.7)	3.5237	(0.0017)	3.5229	15	17	0.0006
B	1	249.9	(346.7)	3.5571	(0.1047)	3.5228	2	2	0.0155
C	5	133.0	(138.8)	3.5477	(0.1066)	3.5221	10	14	0.0241
D D_4	4	309.9	(221.1)	3.5252	(0.0040)	3.5229	12	13	0.0021
D	1	452.4	(258.6)	3.5264	(0.0048)	3.5228	9	9	0.0032
	5	291.1	(235.7)	3.5238	(0.0021)	3.5221	12	13	0.0015
E_4	4	389.4	(386.5)	3.6196	(0.1962)	3.5229	9	11	0.0732

If full cooperation between the agent species exists, results from scenario A, B, and C should be comparable with results from D.

4.4.2 Partial Overlap in Profiles

The last scenario, scenario E, tests how the algorithm performs when only a partial overlap in user request profiles exist. The last row of Table 1 shows the parameters used in the scenario. Six agent species search in parallel applying a mix of user request profiles. The first species of scenarios E, which we denote E_4, has the same client resource (node 4) and request profile as the species in scenario A and the first species in scenario D, which we denote D_4. The other five species of scenario E differ from E_4, as well as among themselves, in both client resources and request profiles. However for every QoS parameter relevant for species E_4, one of the five other species has a profile containing that same parameter (see bold face profile bits in Table 1), i.e. for all QoS parameters relevant for E_4 there are in total two agent species reading and updating pheromones related to the parameter. To ensure that the comparison of the species of scenario A, D_4 and species E_4 is as correct as possible, 6 agents per species are created, i.e. again 12 agents will be reading and updating relevant QoS parameters. Further, an effort was made to ensure that the results obtained for E_4 are at least to some degree independent of which client resources the five last species of scenario E use. The "order" of the client resources (marked with a * in column 4 of Table 1) were shuffled for every simulation initiated while the related request profiles (column 5 in Table 1) are kept in the same order. The results were averaged over 20 simulations based on 20 different orders of the client resources.

If cooperation between the species takes place even when only a partial overlap in request profiles exists, performance results for species E_4 should be comparable with results for the species in scenario A and D_4.

4.5 Results

Table 2 summarizes results from the simulation scenarios described above. Results are based on 20 simulation runs for each scenario. We denote the last path found by a species the *final path*. The path with lowest cost in a simulation is denoted the *preferable path*.

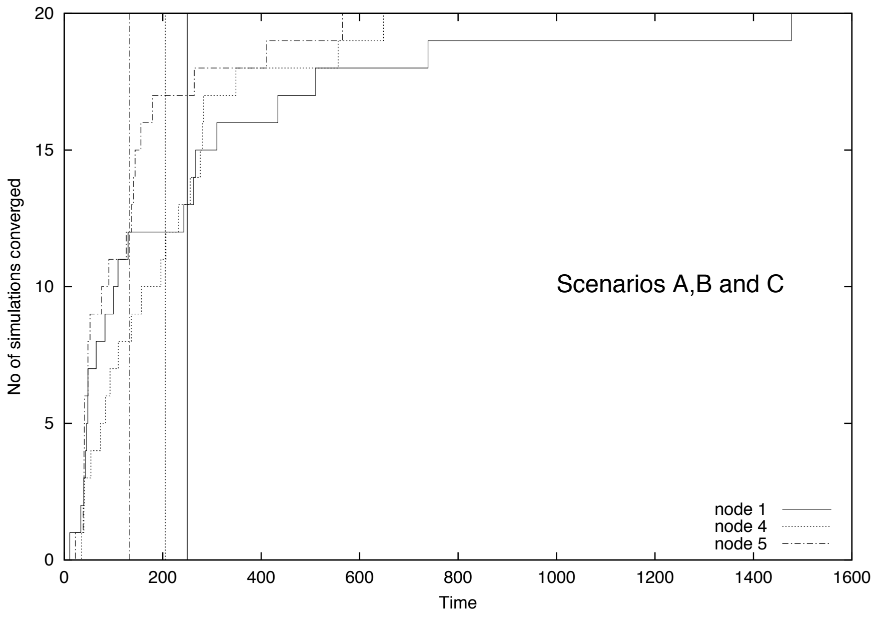


Figure 3. Convergence progress for scenario A (node 4), B (node 1) and C (node 5) .

The two first columns of Table 2 identify the scenario and the relevant client resource. Columns 3 - 6 presents average convergence time in seconds (simulated real time) with standard deviation and average path cost of final path found with standard deviation. Column 7 presents the cost of the overall best final path found during all simulations, and column 8 the number of simulations that converged to this path. Column 9 presents the number of simulations having final path equal to the preferable path found during the simulation. The average difference in cost between final paths reported and the preferable path is given in column 10.

4.5.1 Full Overlap in Profiles

Results for scenarios A, B and C are comparable with results for scenario D. Average path costs differ only to a little extent and they are all close to the best values found. Low standard deviations indicate limited spread among the solutions. For most simulations the final path found after convergence is also the preferable path found during simulation (second last column). For all simulations the average difference between final and preferable paths are very small, i.e. final paths in general tend to be good solutions.

Average values for convergence times differ more than the cost values, and large standard deviations indicate significant spread. Figures 3 and 4 show the convergence progress of scenarios A, B and C, and scenario D respectively.

Both diagrams show average values as straight vertical lines, and the accumulative convergence as lines increasing by steps. In both diagrams it can be observed that 60-65% of the simulations have converged before the average convergence times. The

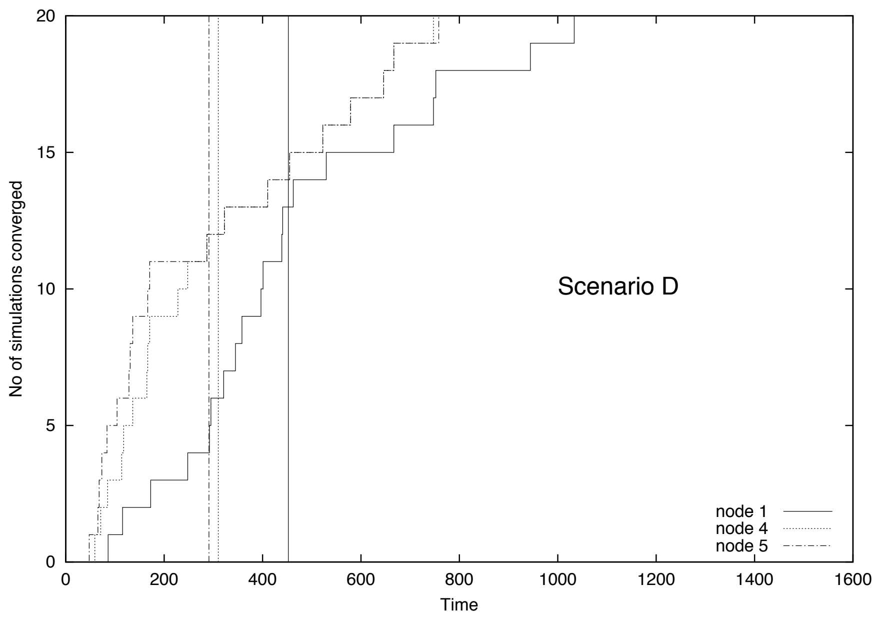


Figure 4. Convergence progress for scenario D.

last 10-20% of the simulations produce a long tail in the distribution of convergence times. This is more significant for the scenarios A, B and C than for scenario D. The long tails are much due to the simple convergence criterion we have chosen. Convergence is considered complete when the probability of re-traversing the last path found is greater than 0.9. Thus when two very similar near optimal solutions exists in the search space, agents can oscillate between finding the one or the other solution for many iteration before one solution is chosen.

Average convergence times increase by less than 100% when we compare A, B and C with D, i.e. they double. However considering that there are in total *three times* as many agents in operation in scenario A, B and C together compared to D, our simulations indicate that efficiency is preserved, and even improved, when unique pheromones per species are replaces by a pheromone per QoS parameter. We observe a 33% reduction in "agent seconds" (number of agents in operation multiplied by convergence time).

4.5.2 Partial Overlap in Profiles

Only results for species E_4, the species using node 4 as client resource, from scenario E are show in Table 2. Again results are comparable. Average path cost is only slightly higher for E_4 than for scenario A and D_4, and as many as 45% of the simulations for E_4 converges to the same best solution as found in A and D_4.

Figure 5 shows a comparison of the convergence progress for scenario A, D_4 and E_4. Similar to what we observed in the previous section, 60% of the simulations have converged before average convergences times, and the slowest 10-30% of the

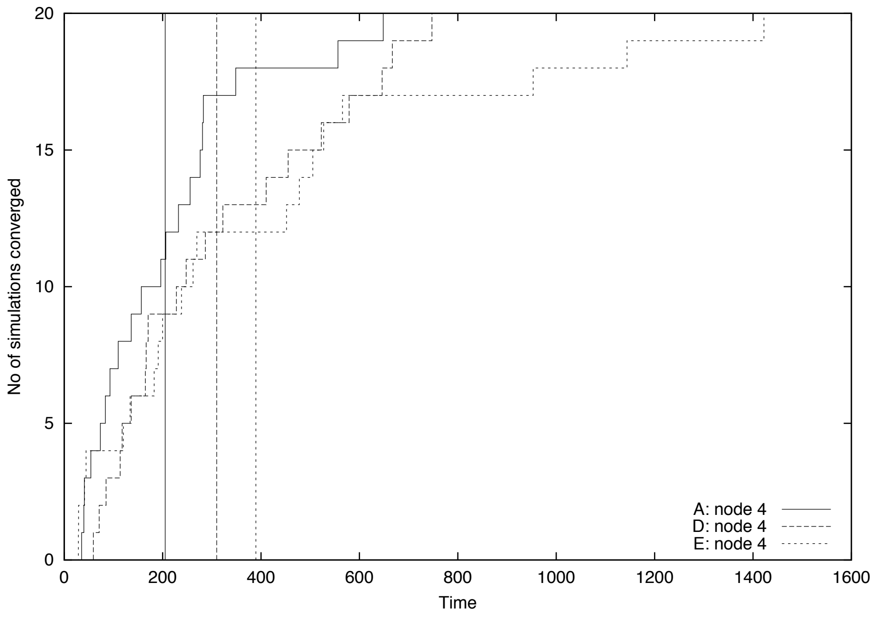


Figure 5. Convergence progress for scenario A, D_4, and E_4 .

simulations create a long tail in the distribution of convergence times. As in the previous section, long tails results from the simple convergence criterion.

Examining the ratios between the averages convergence times, again it can be observed that a less than 100% increase exist when scenario A is compared with D_4 and E_4. Again this can be interpreted as preservation of performance considering that in A all *twelve* agents contribute in finding solutions to *one* request while in D *twelve* agents contribute to *three* profiles and in E *eighteen* agents contribute to *six* profiles. While performance is less than halved, the number of profiles covered is increased by a factor of 3 and 4, implying a reduction in “agents seconds” by 33% and 50% respectively.

Hence we can with reasonable confidence conclude that cooperation between species take place both when there is full overlap and partial overlap in user request profiles. For firm conclusions more tests are required. However, our simulation scenarios indicate that pheromone sharing may contribute in realizing a fully distributed and scalable resource location system.

5. Summary

In this paper we propose a swarm based distributed multi-criteria optimization algorithm which is capable of searching, in an efficient manner, for paths of resources in a complex network environment. The algorithm is QoS aware and ensures to identify resource paths where all resources conform (as much as possible) to a given set of QoS criteria, i.e. the algorithm can implement a QoS aware resource location service.

The algorithm inherits its formal foundations from Rubinstein's work on cross-entropy and combinatorial optimization, and from extensions of Rubinstein's work introduced by Helvik and Wittner. In this paper a new pheromone sharing scheme is introduced to improve scalability. On the contrary to earlier version of the algorithm, the proposed version lets agents share the knowledge stored in pheromones across the network to a greater extent. Care is take not to invalidate the formal foundations, and to construct cost functions providing an efficient search space.

Results from a set of test scenarios show that pheromone sharing enables cooperation between agents. Compared to a non-pheromone-sharing system, a lower total number of unique pheromones can be used without loss of performance, i.e. scalability is improved. Indications exists that cooperation even lead to increased performance.

The test scenarios in the paper only evaluate the algorithm to a limited extend, thus further testing is required. Firstly, larger network environments must be constructed to enable a better evaluation of scalability. Secondly, scenarios testing search in dynamic networks where resources come and go should be implemented. Injecting simulated user traffic into the network is also relevant when examining the algorithms adaptability.

Finally, taking the step from simulations to a real world implementation of the algorithm is also future work.

References

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[DC99] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137–172, 1999.

[DG97] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computing*, 1(1), April 1997.

[GGP02] C. Gagné, M. Gravel, and W. Price. Scheduling a single machine where setup times are sequence dependent using an ant-colony heuristic. In *Abstract Proceedings of ANTS'2000*, pages 157–160, Brussels, Belgium, 7.-9. September 2002.

[HW01] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[IMM01] Steffen Iredi, Daniel Merkle, and Martin Middendorf. Bi-criterion optimization with multi colony ant algorithms. In *Proceedings of the First International Conference* *on Evolutionary Multi-Criterion Optimization (EMO 2001)*, number 1993 in LNCS, Zurich, Switzerland, March 2001. Springer.

[MKL$^+$02] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, March 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf.

[MM99] C. E. Mariano and E. Morales. Moaq an ant-q algorithm for multiple objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 894–901, Orlando, Florida, USA, 13-17 July 1999.

[PK98] Vu Anh Pham and A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine*, 36(7):26–37, July 1998.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[Rubar] Reuven Y. Rubinstein. The Cross-Entropy and Rare Events for Maximum Cut and Bipartition Problems - Section 4.4. *Transactions on Modeling and Computer Simulation*, To appear.

[San02] Richard Torbjørn Sanders. Service-Centered Approach to Telecom Service Development. In *Proceedings of IFIP WG6.7 Workshop IFIP WG6.7 Workshop and EUNICE Summer School on Adaptable Networks and Teleservices, (EUNICE 2002)*, NTNU, Trondheim, Norway, september 2002.

[San03] Richard Sanders. Avantel - advanced telecom services. http://www.item.ntnu.no/avantel/, Visited August 2003.

[Sch00] J. Schuringa. Packet Routing with Genetically Programmed Mobile Agents. In *Proceedings of SmartNet 2000*, Wienna, September 2000.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[VS99] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[WBP98] T. White, A. Bieszczad, and B. Pagurek. Distributed Fault Location in Networks Using Mobile Agents. In *Proceedings of the 3rd International Workshop on Agents in Telecommunication Applications IATA'98*, Paris, France, July 1998.

[WH00] Otto Wittner and Bjarne E. Helvik. Simulating mobile agent based network management using network simulator. Poster in Forth International Symposium on Mobile Agent System (ASA/MA 2000), September 2000.

[WH02a] Otto Wittner and Bjarne E. Helvik. Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks. In *The Third International Workshop on Ant Algorithms, ANTS'2002*, Brussels, Belgium, Sept 2002.

[WH02b] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.

[WHH03] Otto Wittner, Poul E. Heegaard, and Bjarne E. Helvik. Swarm based distributed search in the amigos environment. AVANTEL Technical Report ISSN 1503-4097, Department of Telematics, Norwegian University of Science and Technology, December 2003.

[WPO98] T. White, B. Pagurek, and Franz Oppacher. Connection Management using Adaptive Mobile Agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDAPTA'98)*, 1998.

[ZBMD00] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based Search for Combinatorial Optimization. IRIDIA IRIDIA/2001-15, Universite Libre de Bruxelles, Belgium, 2000.

III

APPENDICES

APPENDIX A: AUTOREGRESSION DETAILS

In this appendix the transition from Reuven Rubinstein's cross entropy method for continuous and combinatorial optimization [Rub99] to Helvik and Wittner's distribute cross entropy algorithm [HW01] is presented in some more detail than in paper A. A few unpublished adjustments in the autoregressive expressions, performed after the publishing date of paper A, are also presented.

1. Fundamentals

As described in paper A the following set of expressions represents the "engine" of Rubinstein's stochastic cross entropy driven optimization algorithm. π_k, denoted *a path,* represents a solution in the relevant solution space. To estimate the temperature γ_t for iteration t of the algorithm

$$\min \gamma_t \text{ s.t. } h(P_t, \gamma_t) \geq \rho \tag{1}$$

is applied where

$$h(P_t, \gamma_t) = \frac{1}{N} \sum_{k=1}^{N} H(\pi_k, \gamma_t) \tag{2}$$

i.e. the lowest temperature γ_t is found which brings the average quality of all paths $\{\pi_k : \forall_k\}$ in a batch of N paths (generated using matrix P_t) above or equal to a certain level of quality given by ρ. Path quality is given by

$$H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}} \tag{3}$$

where $L(\pi_k)$ is the cost of a path. The probability matrix P_{t+1} to be used in the next iteration of the algorithm is generated by

$$P_{t+1,rs} = \frac{\sum_{k=1}^{N} I(\{r,s\} \in \pi_k) H(\pi_k, \gamma_t)}{\sum_{l=1}^{N} I(\{r\} \in \pi_l) H(\pi_l, \gamma_t)} \tag{4}$$

which will minimize the cross entropy between P_t and P_{t+1} and ensure an optimal shift in probabilities with respect to γ_t and the path quality function $H(\pi_k, \gamma_t)$.

For further information on the origin of (1), (3) and (4) the reader is referred to [Rub99] and Paper A [HW01].

2. Transition to a Distributed Algorithm

In this section information about paths stored by an algorithm is denoted the algorithm's *memory of paths*, e.g. the batch on N path described above represents the memory of paths in Rubinstein's centralized algorithm.

2.1 Autoregressive Temperature Calculation

For a new temperature to be calculated in (1) and a new probability matrix generated by (4), all cost values for all N paths, i.e. $\{L(\pi_k) : k = 1, ..., N\}$, must be collected and available. This requirement introduces a synchronization point in Rubinstein's algorithm and hence makes it centralized. To remove the need for synchronization, one assumption is required:

- The optimal temperature γ_t do not change radically when one path is added to the algorithm's memory of paths.

This assumption opens for estimating a new temperature γ_{t+1} by auto-regression.

Auto-regression is introduced by replacing $h(P_t, \gamma_t)$ in (2) with

$$\begin{aligned} h_t^*(\gamma_t) &= h_{t-1}^*(\gamma_t)\beta + (1-\beta)H(\pi_t, \gamma_t) && (5) \\ &\approx \frac{1-\beta}{1-\beta^t}\sum_{k=1}^{t}\beta^{t-k}H(\pi_k, \gamma_t) && (6) \end{aligned}$$

where $h_t^*(\gamma_t)$ represents a geometrically weighted average of the quality of all paths visited so fare. $\beta \in [0, 1]$ controls the level of weighting, i.e. how "clear" paths of different ages are in the algorithm's memory. For instance $\beta = 0.995$ weights the last 240 paths collected with 0.3 or greater ($0.995^{240} = 0.3003$).

The introduction of $(1 - \beta^t)$ as denominator in (6) ensures that $h_t^*(\gamma_t)$ becomes an unbiased estimator for the expected path quality $\mu(\gamma_t)$ of all paths (solutions) in the solution space given a specific temperature γ_t. By viewing all $H(\pi_k, \gamma_t)$ (for all paths) as independent random variables, we have $E[h_t^*(\gamma_t)] = \mu(\gamma_t)$ by

$$\begin{aligned} E\left[\frac{1-\beta}{1-\beta^t}\sum_{k=1}^{t}\beta^{t-k}H(\pi_k, \gamma_t)\right] &= \frac{1-\beta}{1-\beta^t}\sum_{k=1}^{t}\beta^{t-k}E\left[H(\pi_k, \gamma_t)\right] \\ &= \mu(\gamma_t)\cdot\frac{1-\beta}{1-\beta^t}\sum_{k=1}^{t}\beta^{t-k} \\ &= \mu(\gamma_t) \end{aligned}$$

The introduction of $(1 - \beta^t)$ also ensures proper initialization of the path memory. The down scaling effect of the numerator factor $(1 - \beta)$ is reduced due to the denominator $(1 - \beta^t)$ especially for the first 10-20 paths found ($t = 1...20$). Hence "pioneering" paths are remembered "clearly" for some extra iterations creating a better memory-basis for generating probability distributions. Leaving $h_t^*(\gamma_t)$ biased tends to increase the likelihood of too early and premature convergence.

Note that, on the contrary to Rubinstein's algorithm, one iteration ($t \rightarrow t+1$) now means adding only a single path to the algorithm's memory (instead of replacing all N paths in memory with N new paths). Hence a new temperature is to be calculated for every new path found.

Replacing $h(P_t, \gamma_t)$ by $h_t^*(\gamma_t)$ in (1) produces

$$\min \gamma_t \text{ s.t. } \frac{1-\beta}{1-\beta^t} \sum_{k=1}^{t} \beta^{t-k} H(\pi_k, \gamma_t) \geq \rho$$

and since $H(\pi_k, \gamma_t)$ is monotone nondecreasing in γ_t, minimum for γ_t is at equality. Solving

$$\frac{1-\beta}{1-\beta^t} \sum_{k=1}^{t} \beta^{t-k} H(\pi_k, \gamma_t) = \rho \tag{7}$$

for γ_t implies keeping a storage of cost values ($L(\pi_k)$) for all t paths found. To avoid this, based on the assumption stated above (i.e. the temperature does not change radically when a single path is added to the path memory) an autoregressive scheme can be developed for calculating a new temperature. $H(\pi_k, \gamma_t)$ can be estimated by a first order Taylor series expansion, i.e. by $f(x) \approx f(a) + f'(a)(x-a)$, around the inverse of the temperature $\omega_t = \frac{1}{\gamma_t}$. Let

$$\begin{aligned} H(\pi_k, \frac{1}{\omega_t}) &\approx H(\pi_k, \frac{1}{\omega_k}) + H'(\pi_k, \frac{1}{\omega_k})(\omega_t - \omega_k) \\ &= e^{-L(\pi_k)\omega_k} - e^{-L(\pi_k)\omega_k} L(\pi_k)(\omega_t - \omega_k) \\ &= e^{-L(\pi_k)\omega_k}(1 + L(\pi_k)\omega_k - L(\pi_k)\omega_t) \end{aligned}$$

which inserted in (7) gives

$$\begin{aligned} \rho\frac{1-\beta^t}{1-\beta} &= \sum_{k=1}^{t} \beta^{t-k} e^{-L(\pi_k)\omega_k}(1 + L(\pi_k)\omega_k - L(\pi_k)\omega_t) \\ &= \overbrace{e^{-L(\pi_t)\omega_t}}^{H(\pi_t,\frac{1}{\omega_t})} + \overbrace{\sum_{k=1}^{t-1} \beta^{t-k} e^{-L(\pi_k)\omega_k}(1 + L(\pi_k)\omega_k)}^{A_{t-1}} \\ &\quad - \omega_t \overbrace{\sum_{k=1}^{t-1} L(\pi_k)\beta^{t-k} e^{-L(\pi_k)\omega_k}}^{B_{t-1}} \end{aligned}$$

Applying Taylor series expansion again, let

$$\begin{aligned} H(\pi_t, \frac{1}{\omega_t}) &\approx H(\pi_t, \frac{1}{\omega_{t-1}}) + H'(\pi_t, \frac{1}{\omega_{t-1}})(\omega_t - \omega_{t-1}) \\ &= e^{-L(\pi_t)\omega_{t-1}}(1 + L(\pi_t)\omega_{t-1} - L(\pi_t)\omega_t) \end{aligned}$$

and hence

$$\begin{aligned}\rho\frac{1-\beta^t}{1-\beta} &= e^{-L(\pi_t)\omega_{t-1}}(1+L(\pi_t)(\omega_{t-1}-\omega_t))+A_{t-1}-\omega_t B_{t-1}\\ \omega_t &= \frac{A_{t-1}+e^{-L(\pi_t)\omega_{t-1}}(1+L(\pi_t)\omega_{t-1})-\rho\frac{1-\beta^t}{1-\beta}}{B_{t-1}+L(\pi_t)e^{-L(\pi_t)\omega_{t-1}}}\end{aligned}$$

Returning from ω_t to γ_t results in

$$\gamma_t = \frac{B_{t-1}+L(\pi_t)e^{-\frac{L(\pi_t)}{\gamma_{t-1}}}}{A_{t-1}+e^{-\frac{L(\pi_t)}{\gamma_{t-1}}}(1+\frac{L(\pi_t)}{\gamma_{t-1}})-\rho\frac{1-\beta^t}{1-\beta}} \tag{8}$$

A_t and B_t may be expressed as autoregressive functions

$$A_t \leftarrow \beta A_{t-1}+e^{-\frac{L(\pi_t)}{\gamma_t}}(1+\frac{L(\pi_t)}{\gamma_t}) \tag{9}$$

$$B_t \leftarrow \beta B_{t-1}+L(\pi_t)e^{-\frac{L(\pi_t)}{\gamma_t}} \tag{10}$$

where initial values are $A_0 = B_0 = 0$ and $\gamma_0 = -L(\pi_0)/\ln(\rho)$. Hence (8), (9) and (10) now provide the basis for autoregressive calculation of temperatures where only the values for γ_{t-1}, A_{t-1} and B_{t-1} together with the cost of the latest path found $L(\pi_t)$ are necessary to calculate the new temperature γ_t.

Note that since γ_t is recalculated and updated for every new path found, it is not obvious that $h_t^*(\gamma_t)$ remains to be an unbiased estimator for the expected path quality during the first phase of the search process. However, during the final phase of the search process, i.e. convergence, γ_t stabilizes and $h_t^*(\gamma_t)$ again represents the desired unbiased estimation of the expected path quality.

2.2 Autoregressive Generation of Probabilities

The geometrical weighted memory of paths introduced above must be taken into consideration when the transition probability matrix P_{t+1} is to be generated. Hence (4) becomes

$$P^*_{t+1,rs} = \frac{\sum_{k=1}^{t} I(\{r,s\}\in\pi_k)\beta^{t-k}H(\pi_k,\gamma_t)}{\sum_{l=1}^{t} I(\{r\}\in\pi_l)\beta^{t-l}H(\pi_l,\gamma_t)} = \frac{T_{t,rs}}{\sum_{\forall s} T_{t,rs}} \tag{11}$$

where

$$T_{t,rs} = \sum_{k=1}^{t} I(\{r,s\}\in\pi_k)\beta^{t-k}H(\pi_k,\gamma_t)$$

and $I(\ldots)$ is the indicator function. Again to avoid the need to store cost values of all paths, Taylor series expansion of $H(\pi_k,\gamma_t)$ is performed. To ensure better approximation of the hyper exponential denominator of $P^*_{t+1,rs}$ and to guarantee that no non-physical negative values for $H(\pi_k, \gamma_t)$ are produced, a second order Taylor expansion is performed, i.e.

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2$$

is applied. Again expansion is around the inverse of the temperature $\omega_t = \frac{1}{\gamma_t}$. Hence let

$$\begin{aligned}
H(\pi_k, \frac{1}{\omega_t}) &\approx H(\pi_k, \frac{1}{\omega_k}) + H'(\pi_k, \frac{1}{\omega_k})(\omega_t - \omega_k) + \frac{H''(\pi_k, \frac{1}{\omega_k})}{2}(\omega_t - \omega_k)^2 \\
&= e^{-L(\pi_k)\omega_k} - e^{-L(\pi_k)\omega_k} L(\pi_k)(\omega_t - \omega_k) \\
&\quad + \frac{e^{-L(\pi_k)\omega_k} L(\pi_k)^2}{2}(\omega_t - \omega_k)^2 \\
&= e^{-L(\pi_k)\omega_k}\left(1 + L(\pi_k)\omega_k\left(1 + \frac{L(\pi_k)\omega_k}{2}\right) - \right. \\
&\quad \left. -\omega_t\left(L(\pi_k) + L(\pi_k)^2\omega_k\right) + \omega_t^2 \frac{L(\pi_k)^2}{2}\right)
\end{aligned}$$

An approximation of the numerator of (11), $T_{t,rs}$ may now be expressed by

$$\begin{aligned}
\widehat{T}_{t,rs} &= \sum_{k=1}^{t-1} I(\{r,s\} \in \pi_k)\beta^{t-k} \overbrace{e^{-L(\pi_k)\omega_t}}^{H(\pi_k, \frac{1}{\omega_t})} + I(\{r,s\} \in \pi_t) \overbrace{e^{-L(\pi_t)\omega_t}}^{H(\pi_k, \frac{1}{\omega_t})} \\
&= \overbrace{\sum_{k=1}^{t-1} I(\{r,s\} \in \pi_k)\beta^{t-k} e^{-L(\pi_k)\omega_k}\left(1 + L(\pi_k)\omega_k\left(1 + \frac{L(\pi_k)\omega_k}{2}\right)\right)}^{A_{t-1,rs}} \\
&\quad -\omega_t \overbrace{\sum_{k=1}^{t-1} I(\{r,s\} \in \pi_k)\beta^{t-k} e^{-L(\pi_k)\omega_k}\left(L(\pi_k) + L(\pi_k)^2\omega_k\right)}^{B_{t-1,rs}} \\
&\quad +\omega_t^2 \overbrace{\sum_{k=1}^{t-1} I(\{r,s\} \in \pi_k)\beta^{t-k} e^{-L(\pi_k)\omega_k} \frac{L(\pi_k)^2}{2}}^{C_{t-1,rs}} \\
&\quad + I(\{r,s\} \in \pi_t) e^{-L(\pi_t)\omega_t} \\
&= A_{t-1,rs} - \omega_t B_{t-1,rs} + \omega_t^2 C_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-L(\pi_t)\omega_t} \\
&= \overbrace{A_{t-1,rs} - \frac{B_{t-1,rs}}{\gamma_t} + \frac{C_{t-1,rs}}{\gamma_t^2}}^{\widehat{T}_{t-1,rs}} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}}
\end{aligned}$$

where $A_{t,rs}$, $B_{t,rs}$ and $C_{t,rs}$ can be expressed as autoregressive functions

$$\begin{aligned} A_{t,rs} &\leftarrow \beta A_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} \left(1 + \frac{L(\pi_t)}{\gamma_t}\left(1 + \frac{L(\pi_t)}{2\gamma_t}\right)\right) \\ B_{t,rs} &\leftarrow \beta B_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} \left(L(\pi_t) + \frac{L(\pi_t)^2}{\gamma_t}\right) \\ C_{t,rs} &\leftarrow \beta C_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} \frac{L(\pi_t)^2}{2} \end{aligned}$$

The initial values are $A_{0,rs} = B_{0,rs} = C_{0,rs} = 0$. As mentioned above, the second order expansion ensures that the approximation of $H(\pi_k, \gamma_t)$ never produces negative values, however none-physical positive values may occur when γ_t is small and $L(\pi_t)$ is large (due to the parabolic shape of the second order approximation). Two variants of compensation has been applied. In paper A when the history part $\widehat{T}_{t-1,rs}$ (the path memory) in the approximated numerator produces a none-physical positive value, i.e. when $\frac{d\widehat{T}_{t-1,rs}}{d\gamma_t} < 0$, $\widehat{T}_{t-1,rs}$ is replaced by its minimum, which yields

$$\widehat{T}^{*}_{t,rs} = I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} + A_{t-1,rs} + \begin{cases} -\frac{B_{t-1,rs}}{\gamma_t} + \frac{C_{t-1,rs}}{\gamma_t^2}, & \gamma_t > \frac{2C_{t-1,rs}}{B_{t-1,rs}} \\ -\frac{B^2_{t-1,rs}}{4C_{t-1,rs}}, & \text{otherwise} \end{cases}$$

However this compensation is not absorbed by the autoregressive memory functions $A_{t,rs}$, $B_{t,rs}$ and $C_{t,rs}$ which may provoke the need of another compensation in future iteration. To improve upon this a different compensation method has been applied in the papers B, C, D and E. Instead of only replacing $\widehat{T}_{t-1,rs}$ by its minimum the temperature γ_t is replaced by the root of $\frac{d\widehat{T}_{t-1,rs}}{d\gamma_t} = 0$, i.e.

$$\gamma_t^{**} = \begin{cases} \frac{2C_{t-1,rs}}{B_{t-1,rs}}, & \frac{B_{t-1,rs}}{\gamma_t^2} - \frac{C_{t-1,rs}}{\gamma_t^3} < 0 \\ \gamma_t, & \text{otherwise} \end{cases}$$

Now replacing the temperature with γ_t^{**} in the approximation of the numerator and the autoregressive memory functions yields

$$\widehat{T}^{**}_{t,rs} = A_{t-1,rs} - \frac{B_{t-1,rs}}{\gamma_t^{**}} + \frac{C_{t-1,rs}}{\gamma_t^{**2}} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}} \tag{12}$$

and

$$\begin{aligned} A_{t,rs} &\leftarrow \beta A_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}} \left(1 + \frac{L(\pi_t)}{\gamma_t^{**}}\left(1 + \frac{L(\pi_t)}{2\gamma_t^{**}}\right)\right) &(13) \\ B_{t,rs} &\leftarrow \beta B_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}} \left(L(\pi_t) + \frac{L(\pi_t)^2}{\gamma_t^{**}}\right) &(14) \\ C_{t,rs} &\leftarrow \beta C_{t-1,rs} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}} \frac{L(\pi_t)^2}{2} &(15) \end{aligned}$$

Finally (11) is replaced by

$$\widehat{P}^{**}_{t+1,rs} = \frac{\widehat{T}^{**}_{t,rs}}{\sum_{\forall s} \widehat{T}^{**}_{t,rs}} = \frac{A_{t-1,rs} - \frac{B_{t-1,rs}}{\gamma_t^{**}} + \frac{C_{t-1,rs}}{\gamma_t^{**2}} + I(\{r,s\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}}}{\sum_{\forall s} A_{t-1,rs} - \frac{B_{t-1,rs}}{\gamma_t^{**}} + \frac{C_{t-1,rs}}{\gamma_t^{**2}} + I(\{r\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t^{**}}}} \tag{16}$$

Results from simulation scenarios comparing the compensation methods do not give clear indications that the latter is better than the former (or opposite). However, compensation is better than no compensation at all.

To summaries, only three autoregressive variables per outgoing link (rs) are now required to be stored in a node (r). Whenever a new path is found and a new temperature has been calculated, numerator values (which may be claimed to mimic pheromones in natural system) for all links in the new path are updated using (12) followed by an update of the autoregressive memory variables using (13-15). During the next (forward) search for a path (i.e. beginning of iteration $t + 1$), an ant-like agent will generate a next-node-probability-distribution using (16) before selecting which node to visit next.

APPENDIX B: UPDATE OF GLOBAL VALUES

As mentioned in paper A under future work, an investigation of different updating schemes for global values is appropriate. Two global values are of special interest, the *search focus parameter* ρ and the *best cost* (lowest) found so far. In paper A, C and E these values are updated during the search process to speed up convergence. When no new best cost value has been found (and hence no update of best cost values has been perfomed) for a specific number of iterations, ρ is updated to tighten the search focus and by that "push" the search process out of a potentially oscillating state.

This appendix compares four different update schemes for ρ and *best cost*. Table 1 lists the schemes. In the first scheme, denoted *simple,* no probagation of updates of neither ρ nor *best cost* are performed. When an agent realizes an update is required, e.g. when a new best path with best cost is found, only a copy of the relevant value in the agents homenode is updated. Hence only agents sharing a specific homenode cooperate in updating ρ and *best cost*.

The second and third schemes, denoted *asymetric A* and *asymetric B*, perform probagation of updates for one out of the two values. Probagation is realized by making all agents check values stored in all nodes they visit. If a node contains an older value than the value a visiting agent carries, the agent updates the node. Otherwise the agent replaces/updates the value it carries with the value read from the node.

The fourth scheme, denoted *full*, performs probagation of both ρ and best cost found. Probagation is realized as described above for asymetric A and asymetric B.

As indicated by the name, the *simple* scheme is simple and do not introduce new dependencies between agents with different homenodes. Probagation of a new value happens in a passive and indirect manner. After an agent has found a new path with new best cost value, the probability that other agents will find and follow the new path will increase (due to increased pheromone levels, see Paper A). Hence other agents

Table 1. Four update schemes for ρ and *best cost*.

	ρ	*best cost*
Simple	No probagation	No probagation
Asymetric A	No probagation	Probagation
Asymetric B	Probagation	No probagation
Full	Probagation	Probagation

Table 2. Results from comparasions of updating schemes

Topology	Schemes compared	Wilcoxon Rank-Sum		N(0,1)
		Conv. time	Conv. value	$\alpha = 2.5\%$
B26	Simple v.s. Asymetric A	-0.659	-0.306	1.96
	Simple v.s. Asymetric B	-1.136	-0.419	1.96
	Simple v.s. Full	0.324	-2.354	1.96
fri26	Simple v.s. Asymetric A	2.380	-0.203	1.96
	Simple v.s. Asymetric B	1.298	1.388	1.96
	Simple v.s. Full	-0.243	-1.317	1.96

with different homenodes will soon discover the new best path and eventually their homenodes will be updated with the same new best cost value.

The other schemes probagate values actively and potentially faster than the simple scheme. However new dependencies are introduced which degrade the overall robustness of the system. The following example illustrates this. If one agent of a species sharing a homenode should “go bananas” due to an internal failure and start updating ρ and *best cost* with crazy values, these faulty values would probagate to other homenodes and soon influence the behavior of other species of agents. This would not happen if the simple schemes was applied.

Maintaining a robust system is desirable, hence the simple scheme is preferrable given that the system does not perform significantly worse with the simple scheme than with the other schemes.

To investigate differences in perfomance, a statistical interference procedure for comparing pairs of treatments has been applied. Perfomance results from simulations using the simple scheme have been compared to performance results from simulations using each of the other schemes. Table 2 presents the results. In total 12 inteference test have been perfomed, six on convergence times and six on convergence values. Half of the tests were performed using a 26 node fully mesh network topology, denoted *fri26* in Table 2. The other half of the tests used a 26 node scarcely meshed network, denoted *B26* in Table 2, with average connection degree ≈ 5. *B26* is identical to the topology presented in paper C (Figure 2). In all the tests, the systems search for optimal Hamiltonian cycles, i.e. travelling salesman tours (see paper A and C). Results for each interference test in Table 2 are based on 20 simulations.

Since there is little knowledge available about the distributions for the convergence times and convergence values, assuming they follow a normal distribution is not without risk. Hence a distribution-free alternative to a standard ANOVA test was chosen. Results in Table 2 present the output of Wilcoxon Rank-Sum tests for comparing pairs of treatments [BJ77]. The hypotheses relevant for the version of the Wilcoxon tests applied are

$\mathbf{H}_0$ The two distributions are identical

$\mathbf{H}_1$ The distribution of population A is shifted to the right of (i.e. has worse perfomance than) the distribution of population B.

Since the simple scheme (population A in H_1) is preferable, a tight rejection region for H_0 has been chosen. A probability of a type II error greater than 2.5% is viewed as inacceptable. An inacceptable error level implies that the simple scheme will be considere to give a system performance not sigificantly worse than the scheme it is compare with, i.e. H_0 is not rejected. The last column in Table 2 presents the relevant value of the statistic for a rejection region at $\alpha = 2.5\%$, i.e. for H_0 to be rejected the output from a Wilcoxon test must be larger than 1.96.

As column three and four in Table 2 show, all except one test output value are less than 1.96, i.e. H_0 is rejected in only 1 out of 12 test cases. Hence, it is reasonable to claim that overall there is not enough evidence indicating that the simple scheme degrades performance significantly compared to the other schemes.

Based on this conclusion the *asymetric A* scheme applied in paper A has been replaced by the *simple* scheme in papers C and E.

APPENDIX C: PAPER F

This appendix reproduces an extended abstract describing early work on simulating ant-like agents doing network managment using the *Network Simulator* software package. The abstract was publised as a poster presentation at

> *The second international symposium on agent system and applications and the fourth international symposium on mobile agents* (ASA/MA 2000), Zurich, Switzerland, September 2000.

SIMULATING MOBILE AGENT BASED NETWORK MANAGEMENT USING *NETWORK SIMULATOR*

Otto Wittner

Bjarne E. Helvik

Abstract

Large, heterogeneous computer and telecommunication networks with ever changing topology and size are typical environments todays network management (NM) systems struggle to control. Unexpected events occur frequently in these complex networks, and many of the events result in failure situations where the NM system is required to intervene and restabilize the network. By distributing the NM system throughout the network efficiency and dependability can be improved. This is indeed what todays NM system providers and operators are focusing on. Efforts are currently being made to increase the level of distribution of the most popular management architectures in use today (SNMP, OSI, TMN).

Swarm intelligence (i.e. simple mobile software agents with collective behavior) is an alternative concept for implementing distributed applications. Several promising examples of NM applications based on swarm intelligence are under development [CD98, WBP98, VS99a]. For most of these examples a simulator has been used as the major tool to configure system parameters and verify system performance.

This paper describes how *Network Simulator (NS)* has been extended to support simulation of mobile code, and how a network node surveillance system based on swarm intelligence is being developed using *NS*.

NS is an open source simulator package. Development efforts are currently funded by DARPA through the VINT project [DAR]. *NS* is capable of running realistic simulation scenarios of traffic patterns in IP-based networks. The package is implemented as a neat mix of OTcl and C++ classes.

Simulations involving mobile code can not easily be run using the standard *NS* package. An extension is required to include the necessary features.

DARPAs Active Network (AN) architecture enables mobile code in network environments. AN packets and AN enabled network nodes are very similar to mobile agents and mobile agent systems respectively. AN packets can only contains small units of software. This fits nicely with the concept of swarm intelligence (a large number of small and simple agents).

The PANAMA project (TASC and the University of Massachusetts) has developed an AN extension to *NS*. In the original extension AN packets do not contain any executable code. A reduction in packet size is used to simulate execution of a packet. To enable simulation of swarm based applications we have enhanced the AN extension to handle mobile code by allowing a reference to an OTcl object be inserted as packet data in an AN packet. AN enabled nodes has been extended to initiate a "bring back to life" method (called *run()*) in such referenced objects.

Using the AN enabled *NS* we have started developing and testing a network surveillance system . The system is inspired by how ants performed foraging and is based on work done by Dorigo et al on ant colony optimization (ACO) [DC99]. The goal of an ant in our system is to visit all nodes in the network and return to its home node, the quicker the better, i.e. the traveling salesman problem (TSP). Dorigo has developed an efficient ACO-algorithm, Ant Colony System, for solving TSPs but the algorithm requires global access to all nodes which is impossible in a real network. Our system relies only on local information. Figure 1 shows some preliminary results. Our ants do indeed learn more efficient traversal itineraries over time but they struggle to find the optimal itinerary.

More work is required to tune the surveillance system. A genetic algorithm is currently being applied for parameter tuning.

Work is also in progress on testing the performance of the surveillance system in a more realistic environment where several other traffic sources generate traffic in the

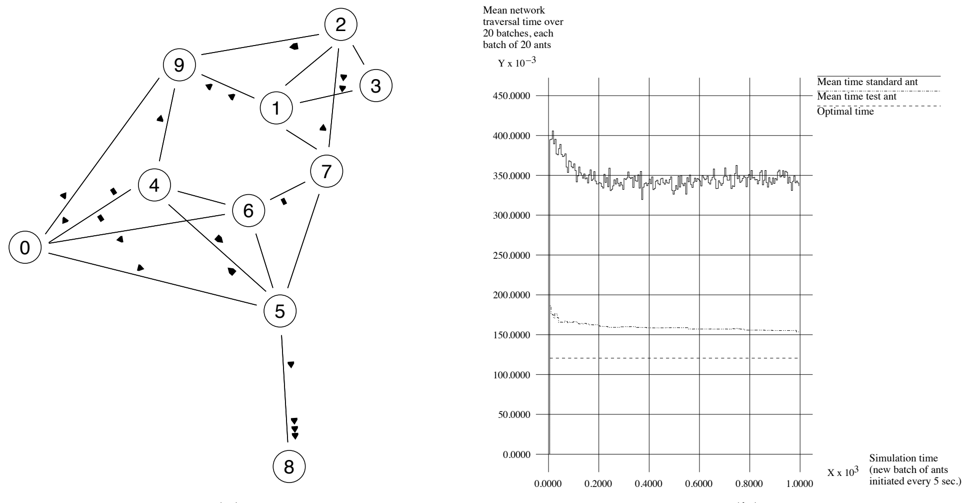


Figure 1. (a) Surveillance ants traversing a network in active network packets. The illustration is a screen dump from *Network Animator*, an animation tool included in the *Network Simulator* suit. (b) Preliminary results showing how the ants learn to traverse the network by "smelling" each others pheromone tracks. Standard ants move stochastically while test ants always choose the most desirable next hop. The lower line is the optimal traversal time for visiting all nodes and returning home.

network . *NS* provides a collection of traffic sources which can easily be added to a simulation setup.

Work has just been initiate on a backup-path reservation system based on swarm intelligence. Ants allocate resources to form backup paths for end-to-end connections. Multiprotocol Labeled Switching (MPLS) is chosen as the underlying routing mechanism. *NS* provides support for MPLS simulation.

Our first experiences with *Networks Simulator* as a simulation tool for swarm intelligence based application are promising. Speed of simulation is reasonable if care is taken when deciding which objects to implement in OTcl and C++. *NS* provides functionality for convenient creation of realistic simulation environments.

References

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[DC99] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137–172, 1999.

[VS99] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[WBP98] T. White, A. Bieszczad, and B. Pagurek. Distributed Fault Location in Networks Using Mobile Agents. In *Proceedings of the 3rd International Workshop on Agents in Telecommunication Applications IATA'98*, Paris, France, July 1998.

APPENDIX D: IMPLEMENTING TECHNOLOGIES

In this appendix a short survey is presented describing potential technologies for implementing emergent behavior based management tools in network management systems. Since a management system may utilize several independent or interacting emergent behavior based tools ([WP98]), the different potential technologies' capabilities for providing *fine grained dynamic distribution* is in focus. Fine grained dynamic distribution should be understood as the concept of having a high number of autonomous components (*fine grained*) which are distributed logically and/or physically (*distribution*), and may be redistributed/relocated frequently logically and/or physically (*dynamic*).

Dynamic distribution can enable efficient introduction of new management tools in general ([Gol96]). Fine grained dynamic distribution can enable efficient introduction of management tools based on emergent behavior. The simple behaviors, from which emergent behavior emerge, are typical distributed autonomous components. A high and varying number of simple behaviors are in general required to realize emergent behavior, hence support for fine grained dynamic distribution in a management system is desirable.

1. Standard Network Management

During the 1980s two of the more significant standards for network and system management were developed, the Telecommunication Management Network (TMN) based on OSI management [ITU00, CCI92] and the Simple Network Management Protocol (SNMP) [CFSD90]. Recently web- and component based technologies have entered the arena challenging the well established standards. See [Slo94a][HAN99] [Udu99][Sta99] for details and tutorial descriptions of TMN, SNMP and their new rivals.

Standard network management systems have default configurations where management functionality (i.e. components form the FCAPS functional areas) is distributed with varying granularity among a set of management units. Fine grained distribution is possible but seldom implemented. Even more seldom are implementations where components are redistributed while the management system is online, i.e. dynamic distribution. The following sections look at each of the management technologies mentioned above and the extent of support they provide for dynamic distribution.

1.1 SNMPv3

The Simple Network Management Protocol (SNMP) standard, defined by the Internet community, was designed with simple networked systems in mind (especially networked computer systems). A traditional hierarchical information model (Management Information Base, MIB) and a simple communication protocol (SNMP) were specified, the MIB with a limited set of data types, and the protocol with a limited set of packet types and possible message interactions. The intended simplicity of the standard has been the sources of SNMPs success and wide acceptance but it is also a weakness when it comes to managing larger complex networked systems.

SNMP version 1 have no support for adding or removing functionality while a system is online. Management centres and network element agents are loaded to specific locations during system initialization and never later redistributed. Two additions to the SNMP standard suite have later been made which can enable dynamic distribution:

- The *distributed protocol interface* (DPI) [WCC+94] was introduced to improve SNMP based systems ability to handle large MIBs and customised MIBs (*agentX* is a further development of DPI [DWEF00]). Using DPI a hierarchy of agents can share the responsibility of managing a MIB. Sub agents can dynamically be added and removed while the system is running, i.e. some degree of dynamic distribution can be said to exist. Still no new functionality is normally initiated in a network element when a sub agent is added. A new part of the MIB may become accessible or the performance of the management software in the element may improved due to better concurrency in request handling.

- The DISMAN (distributed management) charter of IETF has specified MIB subtrees enabling scheduling of management operations [LS99a] and delegation of management scripts [LS99b]. The latter MIB extension provides what is required for online redistribution of management functionality.

1.2 TMN and OSI Management

The Telecommunication Management Network (TMN) standard on the contrary to SNMP is design for large scale management. Based on the object oriented OSI Management standard a large number of different network elements can be managed using advanced hierarchies of management units.

OSI management incorporates two roles for a system: a *manager role* and an *agent role*. A system component can assume any of the roles whenever appropriate. In other word roles can be considered to be dynamically distributable. Theoretically, by installing all functionality, e.i. complete class libraries for manager objects and managed objects and their interfaces, in all network elements (NE) a fine grained dynamic distribution of functionality can be achieved. Whenever a new task is to be handled by an certain NE a manager object in some other NE can delegate the responsibility to the target NE by creating relevant objects. The target NE will then assume whatever role is appropriate to handle the new task.

On the other hand predicting what kind of functionality that may be required to solve problems of the future and installing the appropriate class libraries is difficult. Since a *Common Management Information Service Element* (CMISE), the service element specified by the OSI management standard [ITU97], does not provide any service for uploading new management classes to a network element, efficient dynamic distribution is difficult to realized.

TMN, which by default employ OSI Management, has a static hierarchical constellation of functional blocks and functional components. All the functional components are not specified to be available in all NEs. A functional block, which map one-to-one to a physical NE, is intended to hold a limited set of functional components. Extensions to CMISE enabling explicit reconfiguration and reallocation of functional components during operation has not been specified, thus efficient dynamic distribution is difficult.

Within ITU's overall *Open Distributed Management Architecture,* CORBA has been recognised as an alternative to CMISE [ITU98]. Implicit dynamic distribution may be achieved through CORBAs life-cycle services (see section 1.5).

1.3 JMX

After the introduction of web technology to Internet, web based solution for network and system management have emerged. Some approaches aim for open architectures, where one is the Java Management Extension (JMX, former JMAPI) [McM00]. JMX apply the Java language for specification of management information and can use the hyper text transfer protocol (HTTP) for information transferral. Easy access to information through standard web browser client software is enabled.

JMX adopts a management architecture with similarities to OSI Management, that is a three level hierarchical architecture. At the lowest level, the instrumentation level , basic management objects (MBeans) are installed to give network elements standardized interfaces. At the middle level, the agent level, MBean servers manage MBean life cycles and provide/forward services to the upper level, the distributed services level, where management applications aggregate, interpret and present management information to an operator.

Since JMX's foundation is the Java programming language and virtual machine it inherits the portability and distribution properties of Java. Classes may be created and uploaded dynamically and objects may be serialized and transmitted between hosts. Hence JMX does indeed provide functionality for fine grained dynamic distribution. As will become clear from Section 2 Java forms the foundations for many systems designed to provide fine grained dynamic distribution.

1.4 WBEM

An other web based open architecture is Web-Based Enterprise Management (WBEM) [DMTF03]. WBEM uses the extensible mark-up language (XML) for specification of management information, and (similar to JMX) the hyper text transfer protocol

(HTTP) for information transferral. As for JMX, easy access to management information through standard web browser client software is enabled.

In WBEM a network elements may in a management context play the role of a *client*, *server* or a *listener* (a listener is a server managing a specific type of messages). As for OSI Management, an network element may assume any of the roles whenever appropriate, hence roles may be claim to be dynamically distributable. However, as for OSI Management, just having network elements assuming different roles does not provide true fine grained dynamic distribution. New functionality must still be uploadable. On the contrary to CMISE in OSI Management, the WBEM standard specifies operations for creating new information model classes. Theoretically this enables the creation of a management information class which provides the necessary services for uploading, installation and execution of new management functionality, which again may enable fine grained dynamic distribution.

1.5 CORBA

Component based platforms have been recognised as good candidates for implementing the core of a network management systems. CORBA (Common Object Request Broker Architecture) is one such platform [OMG02a, ITU98].

CORBA provides mechanisms which enable interactions between software objects in a distributed system through well defined interfaces. Among a range of services for object management specified for CORBA, the *Life Cycle Services* [OMG02b] provide what is required to create, remove, move and copy an object. Any (already existent) object can access the life cycle services and create/ remove/ move/ copy another object using an appropriate available factory object.

Hence, assuming necessary factory objects have been created in a network environment to be managed, the life cycle services of CORBA should provide what is necessary to implement fine grained dynamic distribution of management software.

2. Mobile Agents

Mobile agents, mentioned already in the preface of this thesis, are autonomous software objects which by definition must by able to move themselves (migrate) physically and/or logically from node to node in a network environment, and be able to execute a set of operations while visiting a node [PK98]. Hence mobile agent technology have the fundamental properties required to provide fine grained dynamic distribution in a management system.

Two aspects are important to make mobile agents "come alive". Firstly, a suitable implementation language is required which ensures necessary encapsulation of the code and state information that implements an agents behavior. See e.g. [Tho97] for a survey on mobile agent languages. Secondly, a platform must be installed in all nodes to be visited by mobile agents. Such mobile agent platforms must provide a basic set of services, i.e. creation, execution, migration and termination of mobile agents. A range of mobile agent platforms have been developed during the last decade, and new platforms ares still being introduced. A list, known as the the *Mobile Agent List,* of currently available platforms is available at [Dis01].

The general popularity of the Java language as well as its support for portability and encapsulation has made many mobile agent platform developers choose Java as their development language. At the time of writing of this thesis 18 out of 33 mobile agent platforms in the Mobile Agent List are base on Java.

The popularity of Java as a foundation for mobile agent based systems together with such systems ability to provide fine grained dynamic distribution indicate the potential of applying JMX (Section 1.3) as a basis for a network management system. That is, merging JMX and a Java based mobile agent platform may produce a management system foundation able to provide both traditional management services as well as support for management tools of the future.

3. Active Networks

Active Networks is a DARPA funded program, and may be described by the following quotation form the program's home page [DAR97]

> Active networks allow individual user, or groups of users, to inject customized programs into the nodes of the network. "Active" architectures enable a massive increase in the complexity and customization of the computation that is performed within the network, e.g., that is interposed between the communicating end points.

As described in [TSS$^+$97] there are several approaches for realizing active networks, two extremes are the *programmable switch* approach and the *capsule* approach. The latter, the *capsule approach*, is essentially to enable mobile agent technology (Section 2) at the network and transport layers in a network. A packet may contain an autonomous executable object (code and state) which is executed by active network enabled nodes the packet visits on its way through the network. In the *programmable switch* approach an active packet only contains a reference to the code to be executed. Hence the relevant executable code must be injected into active network enabled nodes before an active packet may be executed. Such preloading of code, known as class loading, is common in distributed systems base on Java.

Since active network technology enables every packet sent to carry executable code, support for fine grained dynamic distribution is indeed provided. The fact that functionality can be dynamically uploaded into the lower layers of the protocol stack of a network element, opens for efficient implementation of many network management operations [RS00].

Bibliography

[ALB02] Hasina Abdu, Hanan Lutfiyya, and Michael A. Bauer. Optimizing management functions in distributed systems. *Journal of Network and Systems Management*, 10(4):505–530, 2002.

[AM02] Kaizar A. Amin and Armin R. Mikler. Dynamic agent population in agent-based distance vector routing. In *Proceedings of the Second International Workshop on Intelligent Systems Design and Applications, ISDA 2002*, Atlanta, USA, August 2002.

[AP94] S. Aidarous and T. Plevyak, editors. *Telecommunications Network Management into the 21st Century*. IEEE Press, 1994.

[Bal95a] Michael O. Ball. *Handbooks in Operation Research and Management Science, Network Models*, volume 7. North Holland, 1995.

[Bal95b] Michael O. Ball. *Handbooks in Operation Research and Management Science, Network Routing*, volume 8. North Holland, 1995.

[BCD$^+$00] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. RFC2748: The COPS (Common Open Policy Service) Protocol. IEFT, January 2000.

[BCD02] Mauro Birattari, Gianni Di Caro, and Marco Dorigo. Towards the formal foundation of ant programming. In Marco Dorigo, Gianni Di Caro, and Michael Samples, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume LNCS 2463 of *Lecture notes in computer science*, pages 189–200, Brussels, Belgium, September 2002. Springer.

[BCG$^+$03] Ozalp Babaoglu, Geoff Canright, Luca Maria Gambardella, Andreas Deutsch, and Jim Crutchfield. Biology-inspired techniques for self-organization in dynamic networks. http://www.cs.unibo.it/bison, January 2003. Visited August 2003.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[BJ77] Gouri K. Bhattacharyya and Richard A. Johnson. *Statistical Concepts and Methods*. Wiley, 1977.

[ByH93] Rolv Bræk and Øystein Haugen. *Engineering Real Time systems*. Prentic Hall, 1993.

[Can02] Geoffrey Canright. Ants and loops. In Marco Dorigo, Gianni Di Caro, and Michael Samples, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume LNCS 2463 of *Lecture notes in computer science*, pages 235–242, Brussels, Belgium, September 2002. Springer.

[CCI92] CCITT. Management Framework for Open Systems Interconnection (OSI). CCITT Rec. X.700, September 1992.

[CD97] G. Di Caro and M. Dorigo. Antnet: a mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997.

[CD98] Gianni Di Caro and Marco Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, Dec 1998.

[CDM91] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of ECAL91 - European Conference on Artificial Life*, Paris, France, 1991. MIT Press/Bradford Book.

[CDZ97] K. I. Calvert, M. B. Doar, and E. W. Zegura. Modeling Internet Topology . *IEEE Communications Magazine*, 35(6):160 –163, June 1997.

[CFSD90] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin. RFC 1157: Simple Network Management Protocol (SNMP). IETF, April 1990.

[CL00] D. Camara and A.A.F. Loureiro. A gps/ant-like routing algorithm for ad hoc networks. In *Proceedings of Wireless Communications and Networking Conference, WCNC 2000*, volume 3, pages 1232–1236, Chicago, IL, USA, September 2000.

[CLN00] Jan Chomicki, Jorge Lobo, and Shamin Naqvi. A logic programming approach to conflict resolution in policy management. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 121–132, San Francisco, 2000. Morgan Kaufmann.

[CMPS90] J. D. Case, R. Mundy, D. Partain, and B. Stewart. RFC 3410: Introduction and Applicability Statements for Internet Standard Management Framework. IETF, April 1990.

[Con70] John. H. Conway. The game of life. *Scientific American*, October 1970.

[Cru94a] James P. Crutchfield. The calculi of emergence: Computation, dynamics and induction. *Physica D*, 75:11–54, 1994.

[Cru94b] James P. Crutchfield. *Complexity: Metaphors, Models, and Reality*, volume XIX, chapter Is anything ever new? Considering emergence, pages 479 – 497. Addison-Wesley, Santa Fe Institute Studies in the Sciences of Complexity, Reading, MA, USA, 1994.

[Dam00] R. I. Damper. Editorial for the special issue on 'emergent properties of complex systems': Emergence and levels of abstraction. *International Journal of Systems Science*, 31(7):811–818, 2000.

[Dam02] Nicodemos C. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, Departement of Computing, February 2002.

[DAR] DARPA: VINT project. UCB/LBNL/VINT Network Simulator - ns (version 2). http://www.isi.edu/nsnam/ns/.

[Dar95] Charles Darwin. *The Origin of Species: by means of natural selection, or the preservation of favoured races in the struggle for life*. John Murray, London, UK, 1895.

[DAR97] DARPA. Active networks home page. http://www.sds.lcs.mit.edu/darpa-activenet/, April 1997. Visited August 2003.

[DBC$^+$00] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. RFC2748: The COPS (Common Open Policy Service) Protocol. IEFT, January 2000.

[DC99] Marco Dorigo and Gianni Di Caro. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(3):137–172, 1999.

[DDLS01] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Specification Language. In *Workshop on Policies for Distributed Systems and Networks (Policy2001)*, HP Labs Bristol, 29-31 Jan 2001.

[De95] M. Decina and T. Plevyak (editors). Special Issue: Self-Healing Networks for SDH and ATM. *IEEE Communications Magazine*, 33(9), September 1995.

[DG97] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computing*, 1(1), April 1997.

[DIES$^+$02] A. Das, R.J. Marks II, M.A. El-Sharkawi, P. Arabshahi, , and A. Gray. The minimum power broadcast problem in wireless networks: an ant colony system approach. In *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, September 2002.

[Dis01] Distributed Systems group, Institute of Parallel and Distributed High-Performance Systems, University of Stuttgart. The Mobile Agent List. http://mole.informatik.uni-stuttgart.de/mal/preview/preview.html, Jan 2001.

[DMTF03] Inc. Distribute Management Task Force. Web-based enterprise management (wbem) initiative. http://www.dmtf.org/standards/wbem, Visited August 2003.

[DWEF00] M. Daniele, B. Wijnen, M. Ellison, and D. Francisco. RFC 2741: Agent Extensibility (AgentX) Protocol Version 1. IETF, January 2000.

[EBGS$^+$98] F. Henaux E. Bonabeau, S. Gurin, D. Snyers, P. Kuntz, and G. Thraulaz. Routing in telecommunication networks with smart ant-like agents. In *Proceedings of the Second International Workshop on Agents in Telecommunications Applications (IATA '98)*, volume LNCS 1437 of *Lectures Notes in AI*. Springer Verlag, 1998.

[FHW96] V. J. Friesen, J. J. Harms, and J. W. Wong. Resource Management with VPs in ATM Networks. *IEEE Network*, 10(5), Sep/Oct 1996.

[GB02] Ryan M. Garlick and Richard S. Barr. Dynamic wavelength routing in wdm networks via ants colony optimization. In Marco Dorigo, Gianni Di Caro, and Michael Samples, editors, *Ant Algorithms: Third International Workshop, ANTS 2002*, volume LNCS 2463 of *Lecture notes in computer science*, pages 252–261, Brussels, Belgium, September 2002. Springer.

[GGP02] C. Gagné, M. Gravel, and W. Price. Scheduling a single machine where setup times are sequence dependent using an ant-colony heuristic. In *Abstract Proceedings of ANTS'2000*, pages 157–160, Brussels, Belgium, 7.-9. September 2002.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Glo96] F. Glover. *Tabu Search*. Kluwer, 1996.

[Gol96] German S. Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Colombia University, 1996.

[Gol98] D. Goldberg. *Genetic Algorithms in Search, Optimization and MachineLearn ing*. Addison Wesley, 1998.

[GS98] W.D. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 537 –543, 7-11 June 1998.

[HAN99] H. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems*. Morgan Kaufmann, 1999.

[Hei95] Philip Heidelberger. Fast simulation of rare events in queueing and reliabili ty models. *ACM Transactions on modelling and Computer Simulation*, 5(1):43–85, January 1995.

[HSGK98] Martin Heusse, Dominique Snyers, Sylvain Guerin, and Pascale Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1(2-3):237–254, 1998.

[HW01] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[IMM01] Steffen Iredi, Daniel Merkle, and Martin Middendorf. Bi-criterion optimization with multi colony ant algorithms. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, number 1993 in LNCS, Zurich, Switzerland, March 2001. Springer.

[ITU93] ITU. Introduction to CCITT Signalling System No. 7. ITU Recommendation Q.700 (03/93), March 1993.

[ITU97] ITU. Open Systems Interconnection - Common Management Information Service. ITU Recommendation X.710 (10/97), October 1997.

[ITU98] ITU. Open Distributed Management Architecture - Amendment 1: Support using Common Object Request Broker Architecture (CORBA). ITU Amendment 1 (06/98) to Recommendation X.703, June 1998.

[ITU00] ITU-T. Overview of TMN Recommendations. ITU-T Rec. M.3000, February 2000.

[ITU02] ITU-T. Specification and description language (sdl). ITU-T Rec. Z.100 (08/02), August 2002.

[Joh01] Steven Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities and Software*. Allen Lane The Penguin Press, October 2001.

[KESM$^+$02] I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks, P. II Arabshahi, and A.A Gray. Adaptive-sdr: adaptive swarm-based distributed routing. In *Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN '02*, pages 351–354, Honolulu, HI, USA, Mai 2002. IEEE.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science 220*, pages 671–680, 1983.

[KZS$^+$01] Dale Kutnick, William Zachmann, Val Sribar, Jack Gold, and David Cearley. Commentary: Ibm advances toward autonomic computing. http://news.com.com/2009-1001-253817.html?legacy=cnet, March 2001. Visited August 2003.

[LK99] S. Lipperts and B. Kreller. Mobile agents in telecommunications networks - a simulative approach to load balancing. In *Proceedings on the 5th International Conference on Information Systems, Analysis and Synthesis, ISAS'99*, 1999.

[LS99a] D. Levi and J. Schoenwaelder. RFC 2591: Definitions of Managed Objects for Scheduling Management. IETF, May 1999.

[LS99b] D. Levi and J. Schoenwaelder. RFC 2592: Definitions of Managed Objects for the Delegation of Management. IETF, May 1999.

[LS99c] Yun-Chia Liang and A. E. Smith. An Ant System Approach to Redundancy Allocation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99)*, volume 2, 6-9 July 1999.

[LS99d] E. Lupu and M. Sloman. Conflicts in Policy-based Distributed Systems Management. *IEEE Transactions on Software Engineering - Special Issue on Inconsistency Management*, 25(6):852–869, Nov. 1999.

[LW03] M. Lawlor and T. White. A self organizing social insect model for dynamic frequency assignment in cellular networks. In *Proceedings of 2nd International Joint Conference on Autonomous Agents and Multi Agent Systems, AAMAS 2003*, Melbourne, Australia, July 2003.

[LZHH02] Suiliong Liang, A.N. Zincir-Heywood, and M.I Heywood. The effect of routing under local information using a social insect metaphor. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, volume 2, pages 1438–1443, Honolulu, HI, USA, Mai 2002. IEEE.

[MC93] M. J. Maullo and S. B. Calo. Policy Management: An Architecture and Approach. In *Proceedings of the IEEE First International Workshop on Systems Management, 1993*, pages 13 –26, UCLA, California, April 1993.

[McM00] Eamonn McManus. Jsr-000003 java management extensions (jmx) v1.0 specification. Java Community Process (http://jcp.org), July 2000.

[MESW01] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. RFC3060: Policy Core Information Model - Version 1 Specification. IETF, February 2001.

[MFZ00] P. Martin-Flatin and S. Znaty. Two Taxonomies of Distributed Network and Systems Management Paradigms. Technical Report DSC/2000/032, DSC, EPFL, Lausanne, Switzerland, July 2000.

[MGR97] M. H. MacGregor, W. D. Gover, and K. Ryhorchuk. Optimal spare capacity preconfiguration for faster restoration of mesh networks. *Journal of Network and System Management*, 5(2):159–170, Jun 1997.

[Mic96] Zbigniew Michalewicz. *Genetic algorithms + Data Stuctures = Evolution Programs*. Springer Verlag, second edition, 1996.

[MKL$^+$02] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, March 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf.

[MKM99] Nelson Minar, Kwindla Hultman Kramer, and Pattie Maes. Cooperating mobile agents for mapping networks. In *Proceedings of the First Hungarian National Conference on Agent Based Computation*, 1999.

[MM99] C. E. Mariano and E. Morales. Moaq an ant-q algorithm for multiple objective optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 894–901, Orlando, Florida, USA, 13-17 July 1999.

[Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics magazine*, April 1965. McGraw-Hill publications 35th anniversary.

[OMG02a] OMG. Common object request broker architecture (corba): Core specification v3.0. Object Management Group, http://www.omg.org, December 2002. Visited August 2003.

[OMG02b] OMG. Corba: Life cycle service specification v1.2. Object Management Group, http://www.omg.org, September 2002. Visited August 2003.

[OS99] K. Oida and M. Sekido. An agent-based routing system for qos guarantees. In *Proceedings of 1999 IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC '99*, volume 3, pages 833–838, Tokyo, Japan, October 1999. IEEE.

[PD87] Jacques M. Pasteels and Jean-Louis Deneubourg, editors. *From Individual to Collective Behavior in Social Insects*. Experientia Supplementum. Birkhäuser, Basel, 1987.

[PD00] Larry L. Peterson and Bruce S. Davie. *Computer Networks - A Systems Approach*. Morgan Kaufmann, 2000.

[PK98] Vu Anh Pham and A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine*, 36(7):26–37, July 1998.

[Pos81] Jon Postel. RFC 791: Internet Protocol. IETF, September 1981.

[Pso99] Konstantinos Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, First Quarter, 1999.

[Rei01] G. Reinelt. TSPLIB. Institut für Angewandte Mathematik, Universität Heidelberg, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/, 2001.

[Res03] IBM Research. Autonomic computing. http://www.research.ibm.com/autonomic/index_nf.html, 2003. Visited August 2003.

[RS00] D. Raz and Y. Shavitt. Active networks for efficient distributed network management. *IEEE Communications Magazine*, 38(3):138–143, 2000.

[Rub99] Reuven Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[Rub01] Reuven Y. Rubinstein. *Stochastic Optimization: Algorithms and Applications*, chapter Combinatorial Optimization, Cross-Entropy, Ants and Rare Events - Section 7: Noisy Networks. Kluwer Academic Publishers, 2001.

[Rubar] Reuven Y. Rubinstein. The Cross-Entropy and Rare Events for Maximum Cut and Bipartition Problems - Section 4.4. *Transactions on Modeling and Computer Simulation*, To appear.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. RFC3031: Multiprotocol Label Switching Architecture. IEFT, January 2001.

[RW88] Kenneth A. Ross and Charles R. B. Wright. *Discrete Mathematics*. Prentice Hall, 2nd edition, 1988.

[SA94] Simon Steward and Steve Appleby. Mobile Software Agents for Control of Distributed Systems Based on Principles of Social Insect Behavior. *BT Technology Journal*, 12(2):104–113, April 1994.

[San02] Richard Torbjørn Sanders. Service-Centered Approach to Telecom Service Development. In *Proceedings of IFIP WG6.7 Workshop IFIP WG6.7 Workshop and EUNICE Summer School on Adaptable Networks and Teleservices, (EUNICE 2002)*, NTNU, Trondheim, Norway, september 2002.

[San03] Richard Sanders. Avantel - advanced telecom services. http://www.item.ntnu.no/avantel/, Visited August 2003.

[Sch00] J. Schuringa. Packet Routing with Genetically Programmed Mobile Agents. In *Proceedings of SmartNet 2000*, Wienna, September 2000.

[SDC97] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *IJCAI (2)*, pages 832–839, 1997.

[SG99] D. Stamatelakis and W.D. Grover. Rapid Span or Node Restoration in IP Networks using Virtual Protection Cycles. In *Proceedings of 3rd Canadian Conferance on Broadband Research (CCBR'99)*, Ottawa, 7 November 1999.

[SG00] D. Stamatelakis and W.D. Grover. Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles ("p-cycles"). *IEEE Transactions on Communications*, 48(8):1262–1265, August 2000.

[SHBR96] R. Schoonderwoerd, O.E. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. Technical Report HPL-96-76, HP Labs, May 1996.

[SHBR97] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, 5(2):169–207, 1997.

[Sin99] M. C. Sinclair. Evolutionary telecommunications: A summary. In *Proceedings of GECCO'99 Workshop on Evolutionary Telecommunications: Past, Present and Future*, pages 209–212, Orlando, Florida, USA, July 1999.

[Slo94a] Morris Sloman. *Network and Distributed Systems Management*. Addison-Wesley, 1994.

[Slo94b] Morris S. Sloman. Policy Driven Management for Distributed Systems. *Journal of Network and Systems Management*, 2(4):333–360, 1994.

[Sta99] William Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley, 1999.

[Ste90] Luc Steels. Towards a theory of emergent functionality. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, SAB'90*, Complex Adaptive Systems, pages 451–461, Cambridge, MA, USA, 1990. The MIT Press.

[Tho97] Tommy Thorn. Programming Languages for Mobile Code. *ACM Computing Surveys*, 29(3):213–239, Sep 1997.

[TSS$^+$97] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[TW99] K. Tumer and D. Wolpert. Avoiding braess' paradox through collective intelligence. Technical report, NASA Ames Research Center, 1999.

[Udu99] Divakara K. Udupa. *TMN, Telecommunications Management Network*. McGraw-Hill Telecommunications, 1999.

[VS99a] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[VS99b] Griselda Navarro Varela and Mark C. Sinclair. Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, Washington DC, USA, July 1999.

[WBP98] T. White, A. Bieszczad, and B. Pagurek. Distributed Fault Location in Networks Using Mobile Agents. In *Proceedings of the 3rd International Workshop on Agents in Telecommunication Applications IATA'98*, Paris, France, July 1998.

[WCC$^+$94] B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, and G. Waters. RFC 1592: Simple Network Management Protocol Distributed Program Interface Version 2.0. IETF, March 1994.

[Weg98] Peter Wegner. Interactive foundations of computing. *Theoretical Computer Science*, 192(2):315–351, February 1998.

[WH00] Otto Wittner and Bjarne E. Helvik. Simulating mobile agent based network management using network simulator. Poster in Forth International Symposium on Mobile Agent System (ASA/MA 2000), September 2000.

[WH02a] Otto Wittner and Bjarne E. Helvik. Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks. In *The Third International Workshop on Ant Algorithms, ANTS'2002*, Brussels, Belgium, Sept 2002.

[WH02b] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.

[WH02c] Otto Wittner and Bjarne E. Helvik. Robust implementation of policies using ant-like agents. In *Proceedings of The Network Control and Engineering for QoS, Security and Mobility with focus on Policy-based Networking IFIP and IEEE Conference, NetCon'2002,*, Paris, France, October 2002. Kluwer.

[WHH03a] Otto Wittner, Poul E. Heegaard, and Bjarne E. Helvik. Scalable distributed discovery of resource paths in telecommunication networks using cooperative ant-like agents. In *Proceedings of Congress on Evolutionary Computation, CEC2003*, page Submitted, Canberra, Australia, December 2003. IEEE.

[WHH03b] Otto Wittner, Poul E. Heegaard, and Bjarne E. Helvik. Swarm based distributed search in the amigos environment. AVANTEL Technical Report ISSN 1503-4097, Department of Telematics, Norwegian University of Science and Technology, December 2003.

[Wil93] P.H. Williams. *Model building in mathematical programming*. Wiley, 1993.

[WP98] T. White and B. Pagurek. Towards Multi-swarm Problem Solving in Networks. In *Proceedings of the 3rd International Conference on Multi-agent Systems (ICMAS'98)*, July 1998.

[WP99] T. White and B. Pagurek. Application oriented routing with biologically-inspired agents. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-99*, Orlando, Florida, USA, July 1999. Morgan Kaufmann.

[WPO98] T. White, B. Pagurek, and Franz Oppacher. Connection Management using Adaptive Mobile Agents. In *Proceedings of 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDAPTA'98)*, 1998.

[WTF99] David H. Wolpert, Kagan Tumer, , and Jeremy Frank. Using collective intelligence to route internet traffic. *elligence to route internet trac. In*, 11, 952-958 1999.

[WX00] Ying Wang and Jianying Xie. Ant colony optimization for multicast routing. In *Proceedings of The 2000 IEEE Asia-Pacific Conference on Circuits and Systems*, pages 54–57. IEEE, 2000.

[ZBMD00] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based Search for Combinatorial Optimization. IRIDIA IRIDIA/2001-15, Universite Libre de Bruxelles, Belgium, 2000.