Otto Wittner
Poul E. Heegaard
Bjarne E. Helvik

Swarm Based Distributed Search in the AMIGOS Environment

AVANTEL Technical Report

DEPARTMENT OF TELEMATICS, NTNU
ISSN 1503-4097

NTNU
Norwegian University of Science and Technology

Department of Telematics

AVANTEL TECHNICAL REPORT
1/2002

ISSN-no:
1503-4097

TITLE

Swarm Based Distributed Search in the AMIGOS Environment

DATE

2002-12-09

NO. OF PAGES/APPENDICES

23 (excl. title and preface)

AUTHOR(S)

Otto Wittner, Poul E. Heegaard and Bjarne E Helvik

ABSTRACT

Future user controlled development of telecommunication services combined with powerful terminal equipment results in many heterogeneous services running in a peer-to-peer execution environment. In such a system locating a desired service is challenging. In this paper a swarm based optimization algorithm is presented 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 enabling agents to 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 without performance degradation.

KEYWORDS
Swarm Intelligence
Distributed Search
Multicriteria Combinatorial Optimization
Peer-to-Peer
Amigos

Preface

This technical report describes results from research activities within the AVANTEL project. The first author of this report, Otto Wittner, was engaged as researcher for the project for a period of 8 months, ref. contract titled "*Sverm-intelligens for optimal konfigurasjon av distribuerte teletjenester*" signed by Item, NTNU 2001-01-07 and Telenor 2001-01-09. A description of the major results from the engagement follows this preface. The description is presented in the format of a paper suitable for publishing in a conference or a journal. Other contributions in the AVANTEL project made by the first author are as follows:

- Development of example user scenarios for AMIGOS (see AMIGOS document repository)
- Development of example profile descriptions in XML. Profile suggestions for users, services and terminals were developed (see AMIGOS document repository).
- Participation in misc. meeting and discussions where basic concepts and principle of the AMIGOS service were treated.

The major results, i.e. the paper following this preface, is intended to be submitted to a suitable conference before end of January 2003.

Swarm Based Distributed Search in the AMIGOS Environment

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

December 9, 2002

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 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 enabling agents to 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 without performance degradation.

Keywords Telecommunications, distributed optimization, resource paths, ant-like agents, peer-to-peer.

1 Introduction

As the number of user categories and terminal equipment types are increasing steadily, telecommunication operators have recognized a potential for a range of new telecommunication services.[1] One service category, which has quickly gain popularity, is peer-to-peer systems where users develop and provide services to other users with little or no centralized management. User controlled development of services quickly produce a large range of services where both the number of services and service categories are highly dynamic. Users with powerful terminals take the role as service providers, while other users may chose to initiate their services on dedicated servers in the network. The combination of a dynamic set of services and a peer-to-peer environment for service execution makes locating a relevant service challenging.

Many directory systems for peer-to-peer environments have been developed during the last decade [3]. Common for these systems are limited functionality for specifying quality of service (QoS) parameters in service lookup requests. In most cases only some identity of the service requested can be specified and very few (or no) parameters indicating minimum requirements to resources needed to access the service. The result is often an over all uninteresting service offer. For instance when requesting a multimedia stream, access to a high quality version of the stream may be offered but due to lack of network bandwidth the stream becomes uninteresting.

Several bio-inspired algorithm have shown promising results when applied to telecommunication related problems. A class of bio-inspired algorithms known as *swarm intelligence* systems [4] are potentially robust and may scaled well due the their use of distributed autonomous components. In this paper we present a *swarm intelligence* based algorithm which enables implementation of improved QoS controlled service lookup and access. Related work on bio-inspired methods for service component management can be found in [5, 6, 7, 8, 9]. Our algorithm expects a service lookup request to contain a profile which in addition to indicate constraints and preferences for the service type being requested, indicate constraints and preferences for other resources necessary to access the service. 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 the constraints and preferences of the request profile specified by the user.

[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 involving users and user innovation in the service development process [1]. Peer-to-peer is adopted as a potential environment for distributed service execution. The overall initiative has resulted in the AVANTEL project [2]. This paper presents results form research sponsored by the AVANTEL project.

Our algorithm is based on the concept of *swarm intelligence* which inherits behavior aspects from social insects and animals. Swarm intelligence has successfully be applied to a range of optimization problems [10], some typical in the domain of telecommunications [11, 12, 13, 14]. Swarm intelligence systems use a high number of simple autonomous agents to search for problem solutions. Groups of (or all) agents cooperate by sharing experiences gained during search. Information is shared by asynchronous indirect communication, i.e. messages (pheromones) left in shared databases in the search environment.

Our algorithm is based on work previously published in [15, 16, 17]. 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 introduces mechanisms to manage large scale use of our algorithm. On the assumption that service request profiles in many cases will be overlapping, i.e. contain similar constraints and preferences, 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. To find a desirable ranking of solutions cost functions with implicit constraints are developed which output a quality measure of how well a path of resources conform with a profile request.

The reminder of this paper has five section. Section 2 presents background information and introduces the search domain with related terminology and formalisms. Section 3 introduces the behavior foundations for the agents, describes the cost functions which evaluates criteria during the search process, presents reformulations and additions required to realize extended pheromone sharing between agents, and describes the new agent algorithm. Section 4 describes our experimental setup consisting of five simulation scenarios, and reports and discuss simulation results. Finally Section 6 summarize and indicate 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 view as resources with a related profile. An ordered sequence of resources are denoted a *resource path*.

2.1 Motivation

The motivation for adopting a resource view is the heterogeneity of the expected network environment where one of the AVANTEL project's root services, *AMIGOS* [1], is running.

2.1.1 AMIGOS

Advanced Multimedia In Group Organized Services (AMIGOS) provides the basic functionality required for users to manage and visit a *Meeting Place* (MP). A meeting place 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. Technically a meeting place is a software process running on some server in the AMIGOS network environment.

2.1.2 The AMIGOS Environment

Figure 1 illustrates the expected heterogeneity of the environment where the AMIGOS service will be running. Software processes provide MP service resources which again providing access to multimedia objects. A mix 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.

Both terminals and servers shown in Figure 1 are typically connected (by different transmission technologies) to several other terminals and/or servers, i.e. network nodes in the AMIGOS environment often have a connection degree greater than one. A degree greater than one enables a node to act as a router or a relay in addition to its other capabilities.

All terminals, users and services in the AMIGOS environment are expected to have individual *profiles* which describe their capabilities and limitations in the form of QoS parameters.

Figure 1 does not illustrate the expected scale of the AMIGOS environment. The following rough estimations indicate a more realistic scale considering potential users in Norway only:

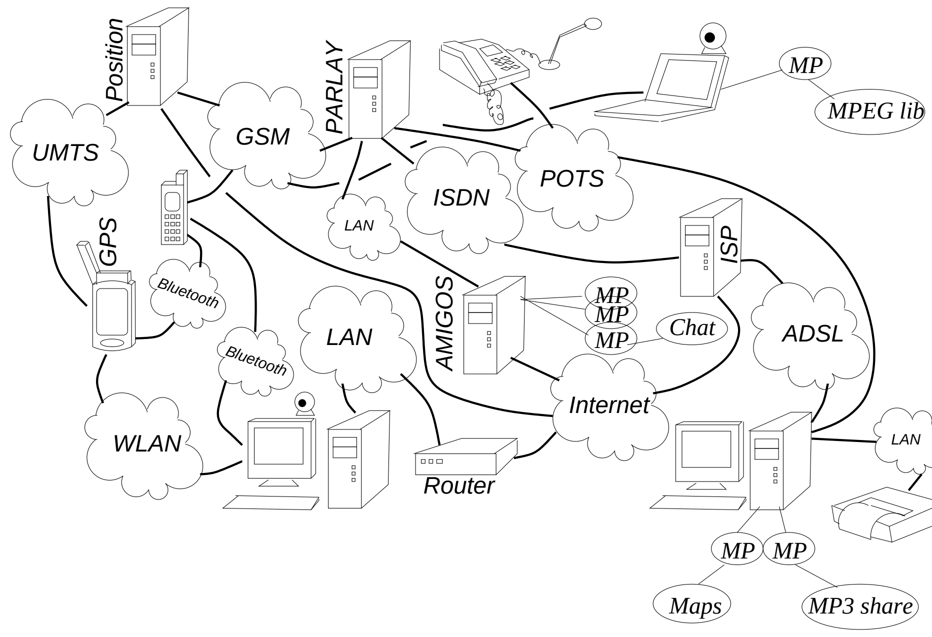


Figure 1: A example network environment where *AMIGOS* is expected to run.

- Total number of users: $N_u \approx 10^6$ (i.e. in the same order as the number of GSM subscribers)
- Number of significantly different terminal / server / router types: $N_t \approx 10^5$
- Number of network transport services: $N_n \approx 10^2$

In the following sub-sections we establish formal terms for describing components in the AMIGOS environment.

2.2 Resource Paths

We view the AMIGOS environment as a composition of telecommunication resources. A resource is denoted ω. In a real instance of the environment, resources we consider will be only those which can be accessed, configured and controlled (see Section 5 for details on implementation issues).

When users are active in the network environment they will access a set of resources. We denote such an ordered set of resources a *resource path*

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

where N_π is the number of resources in the path.

We have chosen to divide the individual resources found in a path into three categories: *Client*, *Transport* and *Peripheral* resources.

A client resources, the first resource $\omega_0 \in R_c$ in a path, enables a user to access other resources in the environment by providing a suitable interface.Typical client resources are terminals running client applications (e.g. a browser). R_c is the set of all client resources.

A transport resource, an intermediate resource $\omega_i \in R_t$ where $i = 1 \ldots N_\pi - 2$, provides a transport service of some quality. Typical transport resources are internal and external transmission links (clouds in Figure 1) and network nodes acting as routers or switches. R_t is the set of all transport resources.

A peripheral resource the last resource $\omega_{N_\pi - 1} \in R_p$ in a path, provides some value added service. Users will normally desire to access a specific type of peripheral resource. Typical peripheral resources are AMIGOS meeting places, multimedia libraries, processing power, multimedia sensors, positioning sensors etc. R_p is the set of all peripheral resources.

In our view access to minimum two resource will be required for any user activity, at least one client resource and one peripheral resource. Thus

$$\min N_\pi = 2 \;\Rightarrow\; \pi = \{\omega_0, \omega_1\} \quad \text{where } \omega_0 \in R_c \text{ and } \omega_1 \in R_p$$

In general a resource path will only contain a single client and a single peripheral resource, but may contain a sequence of transport resources. Thus $\pi \in \Omega$ where

$$\Omega = \{\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}, \omega_{N_\pi-1}\} : \omega_0 \in R_c,\, \omega_{1\ldots N_\pi-2} \in R_t,\, \omega_{N_\pi-1} \in R_p\}$$

is the set of all resource paths. $N_\pi \in \{2, 3, 4, \ldots\}$ is the total number of resources in a path.

A common limitation of today's lookup services for peer-to-peer systems is that the complete resource path required to access a peripheral resource is not taken into account during search. In formal terms this means ignoring the sub-path $\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}\}$ or parts of it. Our algorithm in this paper takes into account the complete path of resources, i.e. $\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}, \omega_{N_\pi-1}\}$.

2.3 Profiles and QoS Objectives

In the AMIGOS environment, users, terminals and services are expected to have individual *profiles* containing QoS parameters. When a user k initiates a requests r for a service, a user request profile $\bar{\zeta}_r(k)$ is generated. For a resource ω_x in the environment we denote $\bar{\zeta}(\omega_x)$ the profile of the resource. Thus there are to classes of profiles:

- Profiles of the user request class contain QoS parameters specifying constraints and preferences in the request, i.e. *QoS objectives*.
- Profiles of the resource class contain QoS parameters specifying limitations and capabilities of a resource ω_x.

2.3.1 User Request Profile

In general, a user request profile may include a large set of QoS parameters. To maintain scalability, we have defined a finite and ordered set Ξ of QoS objectives from which a specific request profile may be constructed. Each element ξ_i in Ξ, where $i = 1, ..., |\Xi|$, is a specific QoS requirement, e.g.

- $\xi_k, \ldots, \xi_l$ may be a range of maximum end to end delays of $\{x_k, \ldots, x_l\}$ seconds,
- $\xi_r, \ldots, \xi_s$ may be a range of minimum bandwidth of $\{y_r, \ldots, y_s\}$ bits/s,
- $\xi_m, \ldots, \xi_n$ may be specific types of resources $\{z_m, \ldots, z_n\}$, e.g. types of peripheral equipment.

A user request profile may now be expressed as $\bar{\zeta}_r(k) = \{\alpha_1, \ldots, \alpha_{|\Xi|}\}$. In this investigation, a binary user request profile is used, i.e.,

$$\alpha_i = \begin{cases} 1 & : \text{ when requirement } i \text{ should be met} \\ 0 & : \text{ otherwise} \end{cases}$$

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. the delays and bandwidths above, only one 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 parameter i in the user request profile. Hence, the resource profile $\bar{\zeta}(\omega_x)$ associated with resource x is represented as a loss vector

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

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, however we consider two extreme values already here. In cases where a QoS requirement ξ_i is irrelevant with respect to resource ω_x, we simply have $\ell_i(\omega_x) \equiv 0$. In cases where a resource ω_x refuses to reveal (e.g. for security reasons) its loss value related to a QoS requirement ξ_i, we have $\ell_i(\omega_x) \equiv \infty$.

Assuming that infinite loss is in general undesirable, we may now define the *search space* $\Omega_{k,r} \subseteq \Omega$ for a user request profile $\bar{\zeta}_r(k)$ by

$$\Omega_{k,r} = \{\pi : \alpha_i \ell_i(\omega_x) < \infty, \, \omega_x \in \pi, \, \alpha_i \in \bar{\zeta}_r(k), \, i \in \{1, \ldots, |\Xi|\}\}$$

i.e. the set of all resource paths consisting of resources which reveal all loss values relevant for the user request. The subscripts k and r of Ω are left out for readability in the remaining of the paper unless this causes ambiguity.

2.4 Environment Simplification

Recall that $\pi \in \Omega$ and

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

In general, resources ω_x where $x = 1, \cdots, N_\pi - 2$ can be alternating transmission links and router nodes in any combination. Although, in a future dynamic network, a probable scenario will be to have several alternative link resources between two different router nodes, we simplify by assuming that only one unique transmission link between a pair of router nodes exists. As a results of this simplification for the rest of this paper we consider ω_x to represent the ith router node together with its corresponding outgoing transmission link, i.e. ω_x represents a pair of resources, one node resource and one link resource.

3 Agent Behavior

Our search algorithm is based on *swarm intelligence* which can be defined as: "*... algorithms or distributed problem-solving devices inspired by collective behavior of social insect colonies and other animal societies*" [4]. Our algorithm mimics the foraging behavior of ants, thus:

- uses a high number of agents with simple behaviors.
- generates one *species* of agents (i.e. one type of agent) for every user request. A species of agents have the only mission of searching for resource paths conforming with the criteria given by the profile of a specific user request.
- lets multiple species of agents search in parallel.

Our algorithm provides a general method for generating solutions to *combinatorial multi-criteria optimization problems* (CMCO problems). A range of nature inspired systems for CMCO exist where many are in the category of evolutionary programs [18, 19].

A few CMCO systems are base on swam intelligence [20, 21, 22]. Most of these build on Dorigo & al.'s *Ant Colony Optimization* system [10] which requires centralization and batch oriented operations to generate solutions efficiently. Our algorithm however, is fully distributed with no central control component. All agents are created, executed and terminated asynchronously of each other, and have weak dependencies to each other. Section 3.1 describes the behavior foundations of our agents in more detail.

Recall that to ensure scalability our algorithm is based on the existence of only a limited number of unique QoS parameters for describing resources and specifying user request. By taking advantage of this limitation our agents are able to cooperate in building a shared distributed "road map" which indicates where to find resources with specific QoS parameters and what quality the resource has. Section 3.3 describes in detail the behavior components realizing this inter-agent cooperation.

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 [11] and further developed in [12, 23, 24]. Schoonderwoerd & al.'s work builds on Dorigo & al.'s work on Ant Colony Optimization (ACO) [25]. 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 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 toward 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.

3.1.1 The Cross Entropy Method

In [26] Rubinstein develops a search algorithm with similarities to Ant Colony Optimization [10, 27]. 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 intensity of pheromones pointing from node r towards node s (one of r's neighbor nodes). An agent's stochastic search for a sample path resemble 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, 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, i.e. $P_t \to P_{t+1}$, 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 [28]) 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 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" 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 of (2) is shown in [26] 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.

3.1.2 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 [15] a distributed and asynchronous version of Rubinstein's CE algorithm is developed. By a few approximations (3) and (1) may be replaced by the autoregressive counterparts

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

and

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

respectively where

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

and $0 < \beta < 1$. 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 (5)) 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 (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 [12] 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.

3.2 Cost Functions

Cost functions applied in this paper output a measure for the level of QoS loss introduced by none-conformance between a specific QoS parameter in a user request and service capabilities of a sequence of resources in a resource path.

Recall that $\Omega_{k,r} \subseteq \Omega$ is denoted the *search space* of a request k for user r. 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 to subsets: *Constraints* 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).

In the next section we compose an overall cost function which measure impairments to constraint as well as best-value QoS parameters. To achieve this we require two classes of terms in the function.

Implicit constraint checks These are terms that implicitly checks constraints given by the constraint QoS parameters, i.e. terms which perform a close to binary conformity check between the requested level of QoS and the service capabilities offered by resources. By returning a high penalty (a high loss value) whenever a resource violates a constraint given by a constraint QoS parameter, the implicit constraint terms will provide a rough sorting of the search solutions in feasible and infeasible solutions. High penalty will contribute to that infeasible solutions rapidly become uninteresting and disappear from the search focus area of solution space. Due to the fact that infeasible solutions are not totally excluded, there is, in a transient phase, a none-zero probability for ending up with search results (resource paths) where some relevant constraint QoS requirements are violated. However, as shown in [16] for swarm based system and in [18] for evolutionary algorithms, the strategy of allowing infeasible solutions may provide important intermediate "stepping stones" in the search space and help the search process converge faster and towards better solutions.

Solution ordering These terms provide a fine grained and continuous measure for the quality of conformance between requested level of QoS specified by best-value QoS parameters and the provided level of QoS. Thus the terms enable a detailed ordering of the candidate solutions found.

To realize *implicit constraint checks* and *solution ordering*, we define two support functions. Both functions perform a sense of rough normalization by mapping values onto the domain $\langle 0.5,\ 1]$. By applying these function, we avoid the need for individual rescaling of QoS loss values, thus values of very different original scale may be compared and/or summarized.

To enable ordering of solutions on a common scale we define

$$h(y) = \frac{1}{1+e^{-\eta \cdot y}}, \quad y > 0 \tag{6}$$

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

$$u(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0.5 & \text{otherwise} \end{cases} \tag{7}$$

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 [26]) 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 with one cost value for each QoS parameter specified:

$$\bar{L}(\pi) = \{L_1(\pi), L_0(\pi), \ldots, L_{|\Xi|}(\pi)\} \tag{8}$$

where additivity is preserved by having

$$\bar{L}(\pi) = \sum_{\omega_x \in \pi} \bar{\ell}(\omega_x) \tag{9}$$

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 (6) and (7) appropriately to normalize and give correct focus to the different elements, thus let

$$L_i^*(\pi) = \begin{cases} u(L_i(\pi)), & \xi_i \in \Xi_C \\ h(L_i(\pi)), & \xi_i \in \Xi_B \end{cases} \tag{10}$$

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

$$\bar{L}^*(\pi) = \{L_1^*(\pi), L_0^*(\pi), \ldots, L_{|\Xi|}^*(\pi)\}$$

The overall scalar cost of a resource path becomes

$$L(\pi) = \bar{L}^*(\pi) \cdot \bar{\zeta}_r(k) = \sum_{i=1}^{|\Xi|} \alpha_i L_i^*(\pi) = \sum_{i \in \Xi_C} \alpha_i u(L_i(\pi)) + \sum_{j \in \Xi_B} \alpha_j h(L_j(\pi)) \tag{11}$$

where, as presented in Section 2.3, the various α_i are specifying the QoS parameters of the user request profile $\bar{\zeta}_r(k)$ which initiated the search for a resource path. $L(\pi)$ may be viewed as an overall measure

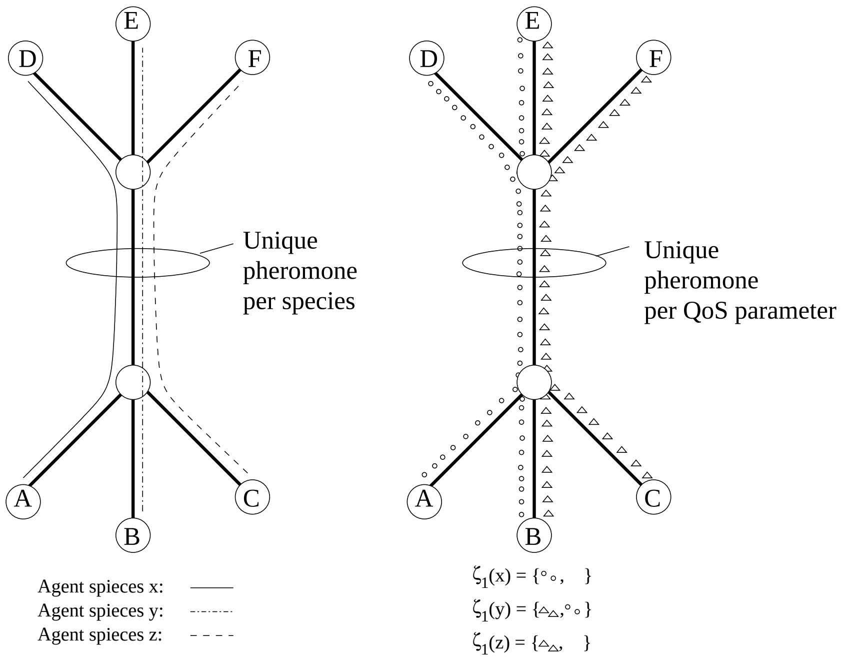


Figure 2: Moving from unique id based (scalar) pheromones to vectors of pheromones where each elements relate to a unique QoS parameter that may be shared by user requests.

for the level of conformance between the request profile $\bar{\zeta}_r(k)$ and all the resource profiles $\bar{\zeta}(\omega_x)$ for all $\omega_x \in \pi$.

The rationale for accumulating all loss values in the vector $\bar{L}(\pi)$ during a search and not use (11) directly, is that this enables a more efficient collection and dissemination of QoS information through pheromone values in the nodes as further outlined in the next section.

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 [15, 16, 17] the distribute CE algorithm allocated one unique pheromone type to every agent species in operation. Reapplying this allocation strategy would mean on unique pheromone type for every user request. In the AMIGOS environment the number of unique user requests to be managed may be very large (c.f. scale indication in Section 2.1.2). Allocating unique pheromones will result in:

- a large amount of pheromone data to be managed by each resource in the network.
- a need for a large number of agents per species to ensure convergence towards relevant solutions in reasonable time.
- a large amount of network traffic generated by agents (c.f. previous pin).

To manage scalability we make different agent species cooperate (on the contrary to work in [17]) in updating a shared set of pheromone values. Instead of a unique id identifying a user request (and a corresponding agent species) we construct a vector containing an element for each QoS parameter in the user request profile. Figure 2 shows how three ant-like agent species (generated by three user requests) may place pheromones in a network. The left figure illustrate use of traditional unique pheromones per species, while the right figure illustrate how less unique pheromones are required when pheromones relate to QoS parameters. Two relevant QoS parameters exist in the scenario of Figure 2. Both parameters apply to species *y* while only a single parameter apply to *x* and z.

By controlling the total number of unique QoS parameters $|\Xi|$ available, we can keep the number of unique pheromones requiring storage space in resources to a manageable level. Note that the total number of possible unique profiles $N_{\bar{\zeta}}$ will be 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. "maximum delay = 70 ms" excludes "maximum delay = 80 ms".

The basis for generating pheromones are cost values output from the cost functions described in the previous section. Now recall the algorithmic step calculating the temperature, i.e. (5). The existence of a unique pheromones 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 (5) for each cost value $L_i^*(\pi_t)$ found in (10) :

$$\bar{\gamma}_t = \left\{\gamma_{t,1},\, \gamma_{t,2}, \ldots,\, \gamma_{t,|\Xi|}\right\}$$

$$\bar{H}(\pi_t, \bar{\gamma}_t) = \left\{H_0(\pi_k, \gamma_{t,0}),\, H_1(\pi_k, \gamma_{t,1}), \ldots,\, H_{|\Xi|}(\pi_k, \gamma_{t,|\Xi|})\right\} \tag{12}$$

where

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

We also calculate γ_t by (5) where

$$H(\pi_k, \gamma_t) = e^{-\frac{L(\pi_k)}{\gamma_t}} \tag{13}$$

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 generates an updated vector of probability distributions, i.e. (4) becomes

$$\bar{P}_{t+1,rs} = \frac{\sum_{k=1}^{t} I(\{r,s\} \in \pi_k)\beta^{t-k}\bar{H}(\pi_k, \bar{\gamma}_k)}{\sum_{l=1}^{t} I(\{r\} \in \pi_l)\beta^{t-k}\bar{H}(\pi_l, \bar{\gamma}_l)} \tag{14}$$

During forward search however the agents require P_{t+1} to select which node to visit next (step 2 from Section 3.1.1), i.e. the probability matrix built from the over all scalar cost measure $L(\pi_k)$ applied in (13). By (13) and (11) we have

$$H(\pi_k, \gamma_t) = \prod_{i=1}^{|\Xi|} e^{-\frac{\alpha_i L_i^*(\pi_k)}{\gamma_t}} = \prod_{i=0}^{|\Xi|} H_i(\pi_k, \gamma_{t,i})^{\alpha_i \frac{\gamma_{t,i}}{\gamma_t}} \tag{15}$$

thus just by carrying $\bar{\gamma}_t$, γ_t and the request profile $\bar{\zeta}_r(k) = \{\alpha_1, \ldots, \alpha_{|\Xi|}\}$ agents can reproduce P_{t+1} during forward search.

3.4 Discovery of Destination Node

Recall that an agent builds a path in a Markov Chain fashion. After adding one resource r to the path, the next resource to include is selected (using the probability vector $P_{t,r}$) from the set of neighbor nodes of the last resource added. Since a user request profile $\bar{\zeta}_r(k)$ do not specify a specific destination resource, we have connected a loopback link resource to every potential destination node, i.e. every peripheral resource is connected to themselves by a link resource ω_{LB} where $\bar{\zeta}(\omega_{LB}) \equiv \bar{0}$. If an agent traverses a loopback link and visits the same peripheral resource twice in a row, the resource is selected as the agents destination resource, and the path found is considered complete.

This "trick" of 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}). Hence the formal foundations hold.

[2]By introducing none-deterministic requirements, i.e. weighting of alternatives by having $\alpha_i \in \langle 0, 1\rangle$, the profile space becomes even richer/larger.

3.5 Agent Algorithm

The most intuitive implementation of our algorithm requires to types of components: *mobile agents* and *network nodes running mobile agents platforms*. See section 5 for a technical discussion. 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 [15] for details on autoregression), and basic arrival, departure and execution functionality for the mobile agents. Rest of the algorithm is implemented in agents. An agent roughly performs the following steps after being created at its client resource (home node) ω_0 with user request profile $\bar{\zeta}_r(k)$ describing its search mission:

A: Forward search

1. Clear tabulist containing resources already visited. Clear $\bar{L}(\pi_t)$. Fetch $\bar{\gamma}_t$, γ_t, and $\{\alpha_i : \forall i\}$ from client resource's database.
2. Record visit to current resource ω_x in tabulist. If current resource has been visited twice in a row, proceed with step B-1, *path evaluation and backtracking* (see below).
3. Reconstruct relevant row $P_{t+1,x}$ of P_{t+1} by (15) using $\bar{\gamma}_t$, γ_t, and $\{\alpha_i : \forall i\}$. Build a next-hop probability table based on $P_{t+1,x}$. Use the tabulist to avoid revisiting none-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 (9).
6. Move to resource ω_{x+1} and proceed with step A-2.

B: Path evaluation and backtracking

1. Calculate finale path cost values $\bar{L}^*(\pi_t)$ by (10).
2. Increase search focus, i.e decrease ρ, if no new best path was found among the last D paths.
3. Calculate new temperatures $\bar{\gamma}_{t+1}$ and γ_{t+1} using (5), cost values in $\bar{L}^*(\pi_{t+1})$ and (11).
4. Backtrack every hop towards the client resource. At each hop update the transition matrix (leave pheromones) by use of (12) and (14)
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 client resource's database while this agent was searching forward). Store the new values of $\bar{\gamma}_{t+1}$ and γ_{t+1}.
6. Goto step A-1 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 for paths satisfying the same (or different) user request profile(s) as long as atomic update operations for resource database variables are provided by all resources.

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 an Active Network (AN) [29] enabled version of "Network Simulator 2" (NS2). NS2 is an open source simulator package capable of simulating realistic IP-based scenarios [30]. The AN extension makes it convenient to implement simulation scenarios where multiple mobile ant-like agents solve problems by exploring a network topology [31]. AN is in addition a potential technology for implementing the algorithm in a real world system (section 5).

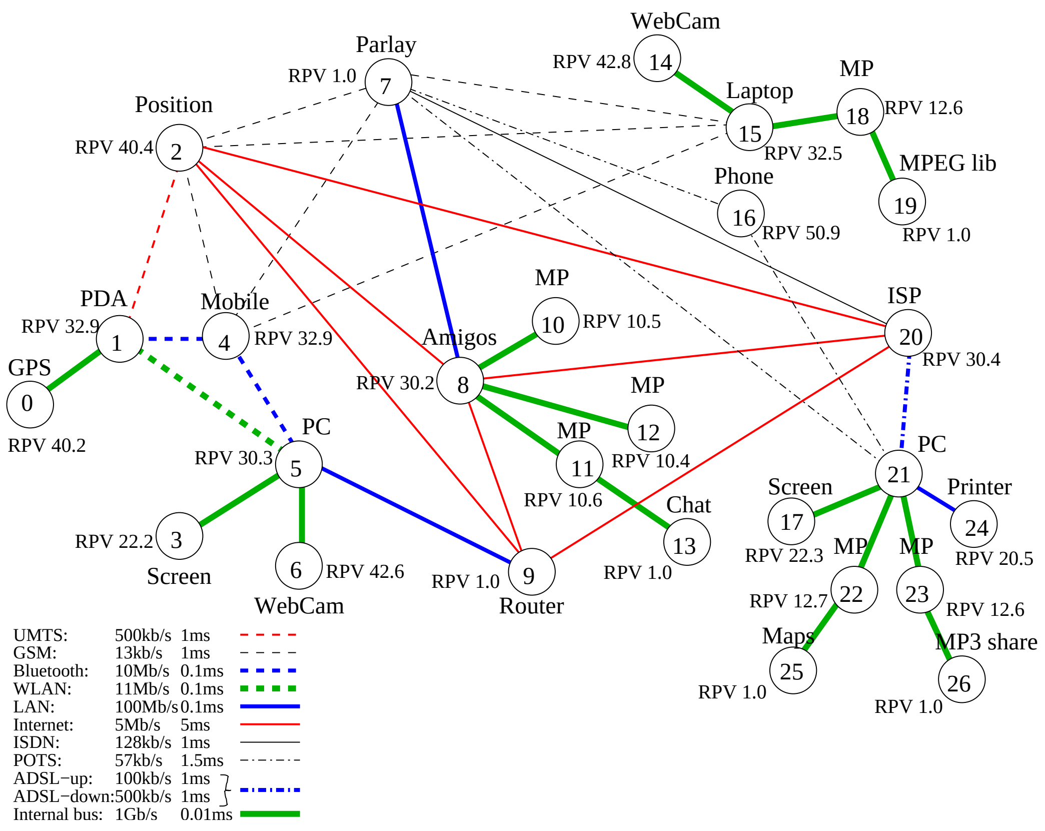


Figure 3: The test environment which is a graph representation of the environment illustrated in Figure 1.

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 3 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 where enabled as routing nodes, i.e. they forward traffic destined for other nodes. Some classes of node resources, e.g. battery powered terminals, would in a real situation not wish to use energy on forwarding traffic. We have chosen to ignore this fact because most simulation scenarios would become uninteresting if we disable routing in only a few nodes in our relatively simple environment.

4.2 QoS Parameters in the Test Environment

For our simulation test scenarios we have chosen to apply user request profiles containing six QoS parameters, i.e. for all requests $\sum_{\forall i} \alpha_i = 6$. In all cases the six parameters include a precision criteria for the peripheral resource (have we found what we are looking for), a quality index for the peripheral resource and four criteria describing the required quality of the client and transport resources. More specifically a parameter ξ_i from one of each of the following parameter groups exist in a request profile.

$\xi_i \in \Xi_{RPI}$, **resource correctness:** - is a requested resource profile index (RPI) which acts as a summary index for some set of resource capabilities. In our test environment, RPIs are irrelevant for all resources except for peripheral resources. An RPI classifies a peripheral resource, e.g. all web cameras have similar RPI. $\Xi_{RPI} \subseteq \Xi$ is the set of all valid RPI parameters. RPI parameters are integer values. Two similar RPIs which may substitute (in part) for each other, have similar values, i.e. the absolute difference is small.

$\xi_i \in \Xi_\delta$, **delay constraints:** - is the maximum accepted delay induced by a resource path. In our test environment only link resources induce delay. $\Xi_\delta \subseteq \Xi$ is the set of all valid maximum induced delay parameters.

$\xi_i \in \Xi_\beta$, **bandwidth constraints:** - is the minimum accepted bandwidth provided by a resource. In our test environment only link resources have limited bandwidth. $\Xi_\beta \subseteq \Xi$ is the set of all valid minimum accepted bandwidth parameters.

$\xi_i \in \Xi_{RPQ}$, **resource quality:** - is the preferred resource profile quality (RPQ) which is a value indicating the quality of an RPI, e.g. several web cameras my have the relevant RPI, however cameras with a quality (e.g. resolution) close to RPQ are preferred. $\Xi_{RPQ} \subseteq \Xi$ is the set of all valid RPQ parameters. An RPQ parameter has an integer part equal to the relevant RPI and a decimal part indicating a level of quality, i.e. $\xi_i \in \Xi_{RPQ}$ is a real value. Low decimal values indicate good quality.

$\xi_i \in \Xi_{\delta^*}$, **relative delay:** - is the delay expected to be experienced when relative delay (expected v.s. measured) equals one. $\Xi_{\delta^*} \subseteq \Xi$ is the set of all valid expected relative delay parameters.

$\xi_i \in \Xi_{\beta^*}$, **bandwidth utilization:** - is the bandwidth expected to be provided by a resource when 100% bandwidth utilization is experienced. $\Xi_{\beta^*} \subseteq \Xi$ is the set of all valid expected bandwidth utilization parameters.

Table 1 shows the set of all QoS parameters (i.e. Ξ) used in our simulation scenarios. The first column lists valid parameter indices, the second indicate which group a parameter belongs to, the third shows the value of a parameter, the fourth indicate which category a parameter belongs to and the final column gives a short description of the parameter.

Consider an example of a user request profile for request r_k with six QoS parameters. Let RPI be 40, δ and δ^* be $90\,ms$, β and β^*be $10\,kb/s$, and RPQ be 40.1, which result in the profile

$$\bar{\zeta}_r(k) = \{0,1,0,1,0,0,1,0,0,0,1,0,1,0,0,1,0,0\}$$

Since only some of the QoS parameters are relevant for router-link resources (recall simplifications from Section 2.4), and others for peripheral resources, two variants of resource profiles exist:

$$\bar{\ell}(\omega_x) = \{0,\, 0,\, 0,\, \ell_4(\omega_x), ...,\, \ell_9(\omega_x),\, 0,\, 0,\, 0,\, \ell_{13}(\omega_x), ...,\, \ell_{18}(\omega_x)\} \tag{16}$$

and

$$\bar{\ell}(\omega_y) = \{\ell_1(\omega_y), \ell_2(\omega_y), \ell_3(\omega_y), 0, ..., 0,\, \ell_{10}(\omega_y), \ell_{11}(\omega_y), \ell_{12}(\omega_y),\, 0, ...\} \tag{17}$$

Table 1: The set Ξ of all valid QoS parameters used in the simulation scenarios.

Index i	Group	Value, ξ_i	Categori	Description
1	Ξ_{RPI}	12	Constraint, Ξ_C	MP multimedia resource
2	Ξ_{RPI}	40	Consraint, Ξ_C	Sensor resource
3	Ξ_{RPI}	20	Consraint, Ξ_C	Animation output
4	Ξ_δ	90 ms	Consraint, Ξ_C	Long delay
5	Ξ_δ	100 ms	Consraint, Ξ_C	Longer delay
6	Ξ_δ	10 ms	Consraint, Ξ_C	Medium delay
7	Ξ_β	10 kb/s	Consraint, Ξ_C	Low bandwidth
8	Ξ_β	100 kb/s	Consraint, Ξ_C	Low+ bandwidth
9	Ξ_β	1 Mb/s	Consraint, Ξ_C	Medium bandwidth
10	Ξ_{RPQ}	12.1	Best value, Ξ_B	Top quality MP multimedia resource
11	Ξ_{RPQ}	40.1	Best value, Ξ_B	Top quality sensor resource
12	Ξ_{RPQ}	20.1	Best value, Ξ_B	Top quality animation output
13	Ξ_{δ^*}	90 ms	Best value, Ξ_B	Long delay
14	Ξ_{δ^*}	100 ms	Best value, Ξ_B	Longer delay
15	Ξ_{δ^*}	10 ms	Best value, Ξ_B	Medium delay
16	Ξ_{β^*}	10 kb/s	Best value, Ξ_B	Low bandwidth
17	Ξ_{β^*}	100 kb/s	Best value, Ξ_B	Low+ bandwidth
18	Ξ_{β^*}	1 Mb/s	Best value, Ξ_B	Medium bandwidth

are the loss vectors for router/link resource ω_x (where $x = 0, ..., N_\pi - 2$) and a peripheral resource ω_y (where $y = N_\pi - 1$) respectively. Note however that in our test scenarios only loss values corresponding to the QoS parameters having $\alpha_i = 1$ in the request profile are considered. This minimum processing of loss elements (i.e. processing only elements relevant for an agent's search mission) is implemented as part of the agent behavior. Thus continuing the example above we may reduce (16) and (17) to

$$\bar{\ell}(\omega_x) = \{...\,0,\ \ell_4(\omega_x),\ 0,\ 0,\ \ell_7(\omega_x),\ 0, ...,\ 0,\ \ell_{13}(\omega_x),\ 0,\ 0,\ \ell_{16}(\omega_x),\ 0,\ 0\}$$

and

$$\bar{\ell}(\omega_y) = \{0,\ \ell_2(\omega_y),\ 0,\ ...,\ 0,\ \ell_{11}(\omega_y),\ 0, ...\}$$

respectively (where $x = 0, ..., N_\pi - 2$ and $y = N_\pi - 1$).

In general, as captured by (9) and (11), the whole range of loss elements may be process by an agent, i.e. an agent may calculate and carry loss information related to QoS parameters not relevant for its search mission and its request profile. This would implement a type of *"super cooperative"* behavior where agents not only cooperate with agents having similar request profiles, but even help other agents with potentially disjoint request profiles. Creating test scenarios for *"super cooperative"* behavior is future work.

4.3 Cost functions

As indicated in Table 1 QoS parameters in the first three parameter groups described in the previous section, i.e. $\xi_i \in \{\Xi_{RPI} \cap \Xi_\delta \cap \Xi_\beta\}$ are classified as constraint parameters, hence

$$\{\Xi_{RPI} \cap \Xi_\delta \cap \Xi_\beta\} \subseteq \Xi_C$$

Parameters in the three last groups of QoS parameters, i.e. $\xi_i \in \{\Xi_{RPQ} \cap \Xi_{\delta^*} \cap \Xi_{\beta^*}\}$ are classified as best-value parameters, hence

$$\{\Xi_{RPQ} \cap \Xi_{\delta^*} \cap \Xi_{\beta^*}\} \subseteq \Xi_B$$

By (9) and (10) 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 (6). The 18 elements of the cost vector $\bar{L}^*(\pi)$ are the following,

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

The calculations required to derive a loss value $\ell_i(\omega_x)$ for the resource profile of a resource ω_x implements 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:** A peripheral resource ω_x in our scenarios have a resource profile value RPV_{ω_x} registered. The RPI part of RPV_{ω_x} is checked against the requested RPI given by ξ_i. In the current implementation, the absolute distance between ξ_i and the integer part of RPV_{ω_x} is returned as the loss value,

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

$\xi_i \in \Xi_{RPQ}$**, resource quality:** The ratio of the distance between requested RPQ value ξ_i and RPV_{ω_x} of resource ω_x, and the decimal part of ξ_i is calculated,

$$\ell_i(\omega_x) = \frac{|\xi_i - RPV_{\omega_x}|}{\xi_i - \lfloor \xi_i \rfloor} \tag{19}$$

$\xi_i \in \Xi_\beta$**, bandwidth constraints:** The capacity B_x of a router-link resource ω_x is checked against the minimum bandwidth requirement given by ξ_i. If the capacity is below ξ_i a non zero contribution equal to the exceeded capacity is returned as loss. Otherwise, zero loss is returned. This loss element adds significant value to the final cost function when the absolute bandwidth requirements are not met, i.e. the path π is not feasible

$$\ell_i(\omega_x) = [\xi_i - B_x]^+ \tag{20}$$

$\xi_i \in \Xi_{\beta^*}$**, bandwidth utilization:** The utilization of a route-link resource ω_x is estimated by the ratio between the minimum bandwidth requirement ξ_i and the router-link capacity B_x. Observe that due to the summation in (9) we give a preference to short paths with high capacity

$$\ell_i(\omega_x) = \xi_i / B_x \tag{21}$$

$\xi_i \in \Xi_\delta$**, delay constraints:** The delay induced by router-link resource ω_x is estimated by adding the propagation delay T_x of the link and the time it takes to transmit N_{ant} bits on to a B_x capacity link. N_{ant} is the number of bits in an active packet containing one of our agents. Total delay Δ_x is estimated by

$$\Delta_x = T_x + \frac{N_{ant}}{B_x} \tag{22}$$

We want the cost element $L_i(\pi)$ to represent a comparison of the sum of all induced delays, $\sum_{\omega_x \in \pi} \Delta_x$, and the maximum delay requirement given by ξ_i, thus

$$\ell_i(\omega_x) = \left[\Delta_x - \frac{\xi_i}{N_{\pi,\delta}}\right]^+ \tag{23}$$

where $N_{\pi,\beta}$ is the number relevant delay constraints for path π. Thus if $\sum_{\omega_x \in \pi} \Delta_x$ exceeds the maximum value given by ξ_i, path π is not feasible and a significant value is added to the cost function. Note that knowledge of the complete path π is required to apply (23), hence in practice $\ell_i(\omega_x)$ will return only Δ_x and further calculations are postponed until the agent reaches the peripheral resource of path π.

$\xi_i \in \Xi_\delta$**, relative delay:** We want the cost element $L_i(\pi)$ to represent the ratio between the sum of induced delays by all resources in the path, i.e $\sum_{\omega_x \in \pi} \Delta_x$, and the maximum delay requirement given by ξ_i, thus

$$\ell_i(\omega_x) = \frac{\Delta_x}{\xi_i} \tag{24}$$

where Δ_x is estimated as described above.

Table 2: Simulation scenarios 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	3	4	4	———-“———
		4	1	———-“———
		4	5	———-“———
E	6	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}
		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}

4.4 Scenarios

To verify correctness of the implementation of the algorithm, we first created a simple scenario where a set of agents of the same species search for one type of peripheral resource. Since discovery of peripheral resources do not introduce new moments (section 3.4) this scenario should produce results comparable to scenarios for previous implementations of the algorithm. A set of simulations gave such comparable results. The results are omitted from this paper.

Further, to get an impression of the algorithms stability, i.e. to verify that pheromone sharing results in cooperative behavior (and not interference or disturbance) between agent species, we created five simulation scenario. The following sections describe two test cases using the five scenarios.

4.4.1 Full Overlap in Profiles

The first four scenarios, A,B,C and D, test cooperation between three agent species. Table 2 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, i.e. in scenarios A, B and C there are 12 agents per species while in scenario D there are 4 agents per species.

If full cooperation between the agents species exist, results form scenario A,B, and C should be comparable with results from D, i.e. there should be little difference in convergence speed and quality of the paths found.

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. Table 2 shows the parameters used in the scenario. Six agent species search in parallel applying a mix of user request profiles. The first species, which we denote E_4, has the same client resource (node 4) and request profile as the species in scenario A. The other five species 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 2), 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 and species E_4 is as correct as possible, 6 agents per species is 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 2) where shuffled for every simulation initiated while the related request profiles (column 5 in Table 2) are kept in the same order. The results were averaged over 20 simulations based on 20 different orders of the client resources.

Table 3: Simulation results

Scenario	Client resource (node #)	Average convergence time	Average path cost	Best path cost	Convergence to best in simulation
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	4	309.9 (221.1)	3.5252 (0.0040)	3.5229 (12)	13 (0.0021)
	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	389.4 (386.5)	3.6196 (0.1962)	3.5229 (9)	11 (0.0732)

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 the first species (with client resource at node 4) in scenario D.

4.4.3 Agent Parameterization

Our algorithm, as earlier versions, requires a small set of general parameters to be configure. All scenarios were configured with the following parameter values:

Initialization phase $D = 500$. To initialize pheromone levels (build and initial probability matrix) all agents start off with a random search behavior, i.e. after excluding previous visited resources they choose their next resource using a uniform random distribution. After an agent species collectively have found D paths, all agents of that species change behavior and continue searching now guided by CE (pheromones).

Search focus, $\rho = 10^{-2}$. The search is concentrated towards the approximate 1% best of the paths found and "remembered" (see β parameter), i.e. a tight focus.

Search focus adjustment, $\rho^* = 0.95$. Search is tightened by 5% by applying $\rho = \rho \cdot \rho^*$ if no new best solution exists among the last D solutions found.

History, $\beta = 0.995$. The system "remembers" a collection of paths found. However as the "age" of a path increases the quality decreases following a geometric distribution controlled by β, i.e. about 240 paths are "clear" in memory (weighted by > 0.3) at any time.

Convergence probability, $cp = 0.9$. Convergence for an agent is defined to be complete when the probability of re-traversing the last path found by the agent is greater than 0.9.

The values are heuristically chosen to avoid premature or slow convergence.

4.5 Results

Table 3 summarizes results from the simulation scenarios described above. Results are based on 20 simulation runs for each scenario. The two first columns identify the scenario and the relevant client resource. Column 3 and 4 presents average convergence time in seconds (simulated real time) and average path cost of final path found respectively, both with standard deviations in brackets. Column 5 presents the cost of the overall best final path found during all simulations, with the number of simulations that converged to this path in brackets. The last column presents the number of simulations that converged to the best path found during the simulation, i.e. how often agents report the best path they can find as the final path. The average difference in cost between final paths reported and the best path found is given in brackets.

4.5.1 Full Overlap in Profiles

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

Average values for convergence times differs more than the cost value, and large standard deviations indicate significant spread. Figure 4 compares the convergence progress of scenarios A, B and C with scenario D. 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% (12-13 out of 20) of the simulations have converged before the average convergence times. The 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. As shown in Figure 5, when two very similar near optimal solutions exists in the search space, agents can oscillate between finding the one or the other solution for quiet a few iteration before one solution is chosen as the better one. Thus our (too) simple convergence criterion forces the agents to choose only one out of a set of solutions which in practice should all be presented to the users as potential resource paths.

Average convergence times increase by around 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 as in 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" (no. of agents in operation multiplied by convergence time). Thus reduction in the total number of unique pheromones required is possible without loss of performance when full overlap in request profiles exists.

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 3. (Results for the other 5 species in scenario E have little value due to the shuffling of client resources.) Again results are comparable. Average path cost is some what higher for E_4 than for scenario A and D_4 (the species in scenario D using node 4 as client resource), however 45% of the the simulations for E_4 converges to the same best solution as found in A and $D_{4.}$

Figure 6 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 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, i.e 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.

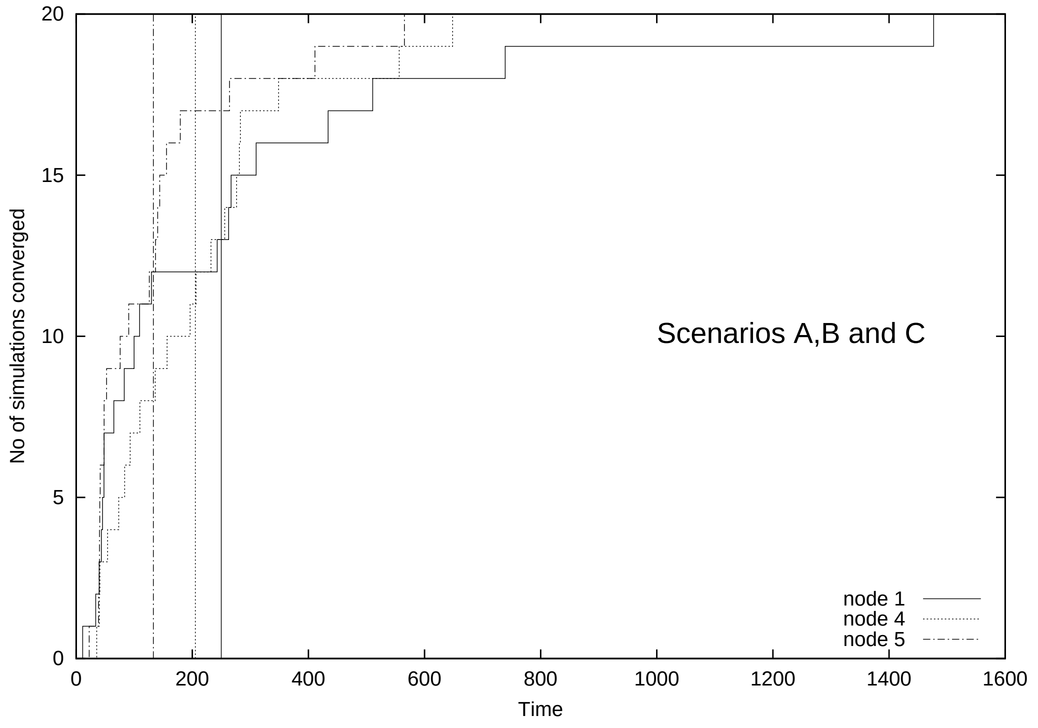
Hence we can with reasonable confidence conclude that cooperation between species take place both when there is full overlap and partial overlap in user request profiles. For firm conclusions more tests are required. However our simulation scenarios indicate that pheromone sharing may contribute in realizing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

5 Implementation Issues

Even if the formal foundations for our algorithm may seem complex, a limited effort is required for implementing an ant-like agent. The agent behavior presented in Section 3 is simple, and the set of services required for an agent to be able to visit a resources is very limited.

The following are examples of technologies which may be suitable for implementing an ant-like agent:

- A message containing only agent state. Agent code must be preinstalled or loaded on demand by resources to be visited, e.g. a Java class.
- Mobile agent technology, i.e. both code and state is encapsulated in an object which transmits itself from host to host. This technology requires dedicated *mobile agent platforms* to be running in all resources to be visited.



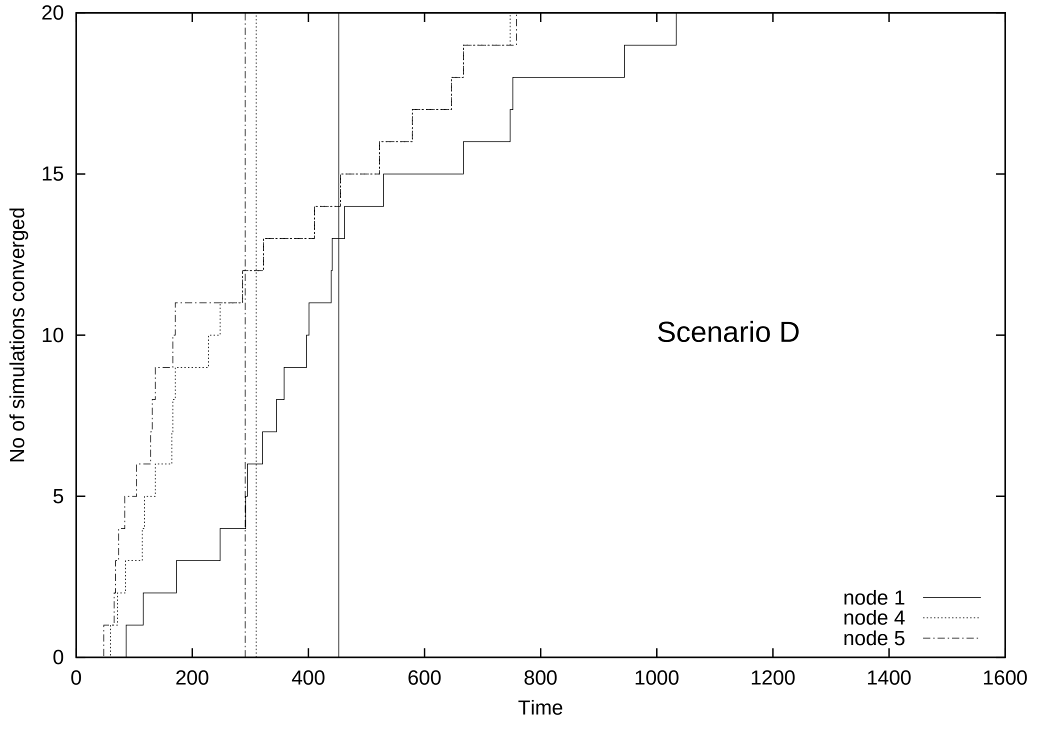


Figure 4: Convergence progress for scenario A (node 4), B (node 1) and C (node 5) in upper diagram, and scenario D in lower diagram.

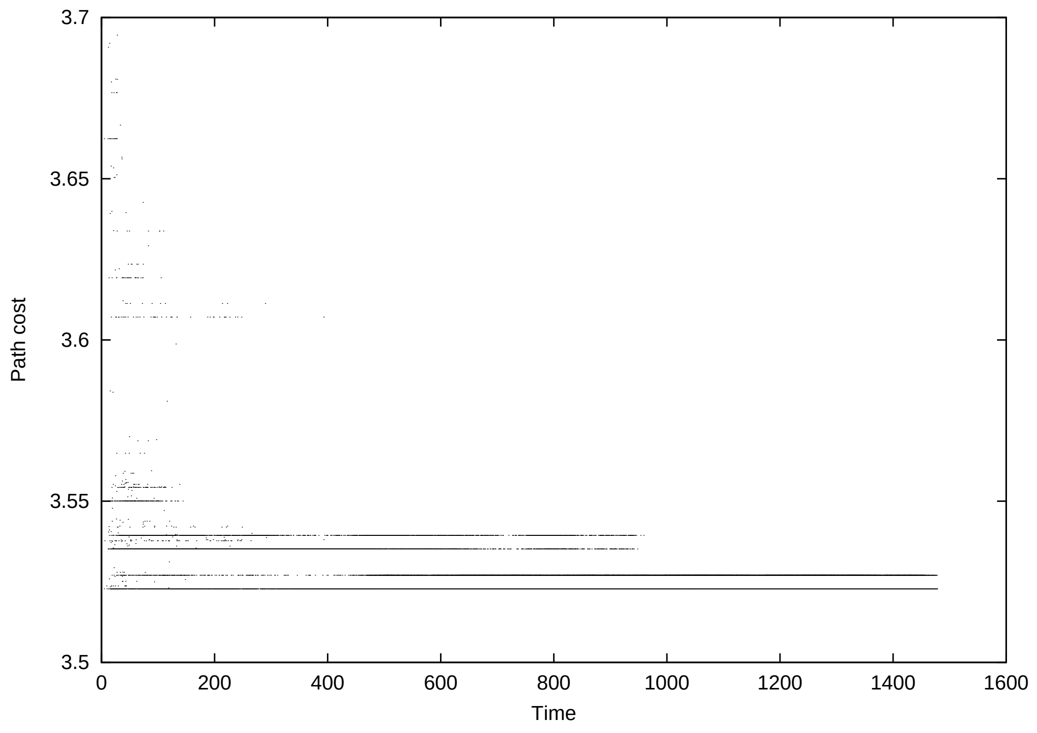


Figure 5: Progress of convergence for one of the "slow" simulations of scenario B. (Note: To better differentiate between the lower cost values, the range of the y-axis has been reduced, i.e. path cost above 3.7 are not shown.)

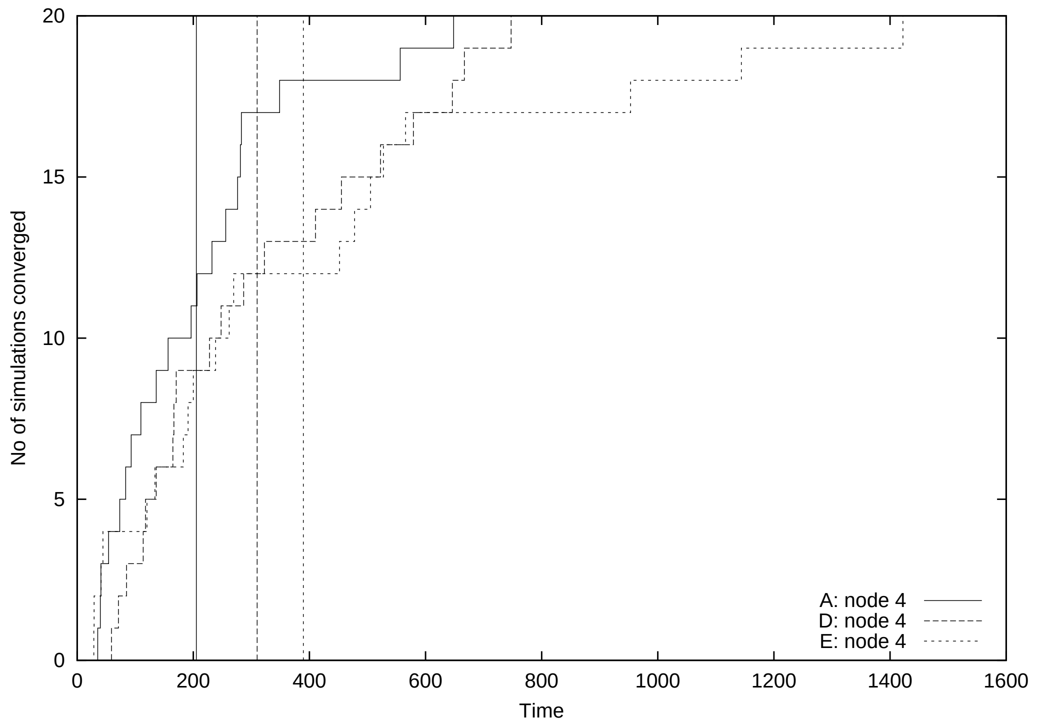


Figure 6: Convergence progress for scenario A, D_4, and E_4 .

- Active packets, i.e. state and a limited amount of code is wrapped in an tagged IP packet. This technology requires resources to be *active network enabled*.
- Ericsson' s Java Frame, which extends Java and provides mechanisms for efficient encapsulation and migration. Java Frame can easily be extended to implement an *mobile agent platform*.

An ant-like agent system may be operative even if only a limited number of resource are capable of handling agent visits. The number of resources seen by the agent, i.e. the search space Ω, would simple be the capable resources. Independent upgrading of individual resources in a network should be possible, and may be performed whenever a resource is considered relevant for the search space.

However a less straight forward part of putting the system into operation within a realistic setting are the definition of loss functions (c.f. Section 2.3.2). Each class of resource require a customized loss function. To provide a smooth and efficient search space as possible, loss functions should be continuous and preferably return values with a physical relation to the resource in question. Establishing a design process for composing such function is future work.

6 Summary

As the overall heterogeneity of user equipment and provided services in telecommunication environments increase, the need for new management techniques are required. As more users take the role as service providers, a peer-to-peer type of execution environment will continue to emerged. Locating specific resources or sets of resources in a manner which captures requested levels of QoS has so far been addressed only to a limited extent.

In this paper we propose a swarm based distributed multi-criteria optimization algorithm which is capable of searching, in an efficient manner, for near optimal paths of resources in a large and complex network environment. 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 function providing an efficient search space.

Results from a set of test scenarios indicate that pheromone sharing lead to cooperation between agents. Compared to a none-pheromone-sharing system, a lower total number of unique pheromones can be used without loss of performance. 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 on our list of future research activities.

References

[1] R. T. 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.

[2] R. Sanders, "Avantel - advanced telecom services." http://www.item.ntnu.no/avantel/, visited 2002-10-18 2002.

[3] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-peer computing," Tech. Rep. HPL-2002-57, HP Laboratories, Palo Alto, March 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artifical Systems*. Oxford University Press, 1999.

[5] S.-J. Sun, D.-W. Lee, and K.-B. Sim, "Artificial immune-based swarm behaviors of distributed autonomous robotic systems," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, (Seoul, Korea), IEEE, May 2001.

[6] K. Mori, "Expandable and fault tolerant computers and communications systems -autonomous decentralized systems," in *Proceedings of The Fourth IEEE Symposium on Computers and Communications*, (Red Sea, Egypt), p. 228, July 06 - 08 1999.

[7] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Adaptive autonomous management of ad hoc networks," in *Proceedings of 8th Network and Operations Management Symposium*, (Florence, Italy), 15-19 April 2002.

[8] M. Wang and T. Suda, "The bio-networking architecture: A biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications," in *Proceedings of the 1st IEEE Symposium on Applications and the Internet (SAINT)*, (San Diego, CA), IEEE, 8-12 January 2001.

[9] T. Itao, T. Nakamura, M. Matsuo, T. Suda, and T. Aoyama, "Service emergence based on relationship among self-organizing entities," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT'02)*, (Nara City, Nara, Japan), pp. 194–203, January 28 - February 01 2001.

[10] M. Dorigo and G. D. Caro, "Ant Algorithms for Discrete Optimization," *Artificial Life*, vol. 5, no. 3, pp. 137–172, 1999.

[11] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based Load Balancing in Telecommunications Networks," *Adaptive Behavior*, vol. 5, no. 2, pp. 169–207, 1997.

[12] G. D. Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research*, vol. 9, pp. 317–365, Dec 1998.

[13] 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.

[14] G. N. Varela and M. 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.

[15] B. E. Helvik and O. 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.

[16] O. Wittner and B. 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.

[17] O. Wittner and B. 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), IEEE, May 12-17th 2002.

[18] Z. Michalewicz, *Genetic algorithms + Data Stuctures = Evolution Programs*. Springer Verlag, second ed., 1996.

[19] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[20] 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*, (Orlando, Florida, USA), pp. 894–901, 13-17 July 1999.

[21] 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*, (Brussels, Belgium), pp. 157–160, 7.-9. September 2002.

[22] S. Iredi, D. Merkle, and M. Middendorf, "Bi-criterion optimization with multi colony ant algorithms," in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, no. 1993 in LNCS, (Zurich, Switzerland), Springer, March 2001.

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

[24] J. Schuringa, "Packet Routing with Genetically Programmed Mobile Agents," in *Proceedings of SmartNet 2000*, (Wienna), September 2000.

[25] M. Dorigo, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computing*, vol. 1, April 1997.

[26] R. Y. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology and Computing in Applied Probability*, pp. 127–190, 1999.

[27] 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.

[28] R. 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.

[29] K. Psounis, "Active Networks: Applications, Security, Safety, and Architectures," *IEEE Communications Surveys*, vol. First Quarter, 1999.

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

[31] O. Wittner and B. 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.