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

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

*Centre for Quantifiable Quality of Service in
Communication Systems (Q2S),
Centre of Excellence, NTNU

†Telenor Research and Development
Telenor Trondheim

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 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 [3]. 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 [4] 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 [5], some in the domain of telecommunications [6, 7, 8, 9]. 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 [10, 11, 12]. 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 Multimedia In Group Organized Services (AMIGOS) [1], 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
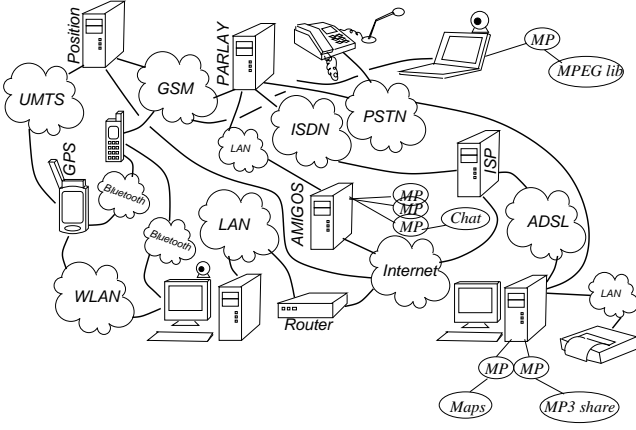

Figure 1: A example network environment where *AMIGOS* is expected to run.

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*

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

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 $i = 0 \ldots N_\pi - 2$. 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 $\pi \in \Omega$

$$\Omega = \{\{\omega_0, \omega_1, \cdots, \omega_{N_\pi-2}, \omega_{N_\pi-1}\} : \quad \omega_{0\ldots N_\pi-2} \in R_t, \\ \omega_{N_\pi-1} \in R_p\}$$

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

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

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. 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

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

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.

3 Agent Behavior

Our search algorithm is based on *swarm intelligence* [4] 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 [13, 14, 15] exist. Most of these build on Dorigo & al.'s *Ant Colony Optimization* system [5] 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 [6] and further developed in [7, 16, 17]. Schoonderwoerd & al.'s relates to Dorigo & al.'s work on Ant Colony Optimization (ACO) [18]. 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 [19] Rubinstein develops a centralized search algorithm with similarities to Ant Colony Optimization [5, 20]. 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 [21]) 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.

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

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

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

where $P_{t,ij}$ is the transition probability from node i to j at iteration t. The solution of (3) is shown in [19] to be

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

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 [10] 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

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

and

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

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 [7]. However, none of the earlier systems implements a search focus stage (the adjustment of γ_t) as in the CE algorithms [21]. 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 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 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

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

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

define

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

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 [19]) 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.

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

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

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

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

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

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

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

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 [10, 11, 12] 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 [12]) 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) :

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

$$\bar{H}(\pi_k, \bar{\gamma}_t) = \begin{cases} H_1(\pi_k, \gamma_{t,1}),\, H_2(\pi_k, \gamma_{t,2}), \\ \ldots,\, H_{|\Xi|}(\pi_k, \gamma_{t,|\Xi|}) \end{cases} \quad (13)$$

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

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

for reasons described below.

Path backtracking and pheromone placing activities performed by agents, i.e. step 3 of the algorithm from Section

---

2By 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.

3.1.1, now generate an updated vector of probability distributions, i.e. (5) becomes

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

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

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

$$(16)$$

By carrying $\bar{\gamma}_t$, γ_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* [22]. 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 [10] 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 $\bar{\gamma}_t$, γ_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 $\bar{\gamma}_t$, γ_t, and $\bar{\zeta}_r(k)$. Build a next-hop probability table and use the tabu list 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 (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 [23] 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)[24]. NS2 is an open source simulator package capable of simulating realistic IP-based scenarios [25].

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.
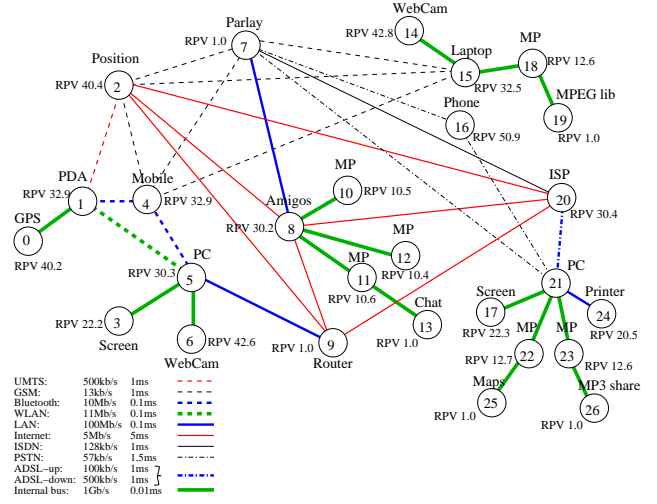

Figure 2: The test environment which is a graph representation of the environment illustrated in Figure 1. An RPV (resource profile value) holds both a node's RPI (resource profile index) and RPQ (resource profile quality). See [23] for details.

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.

- $\Xi_\delta \subseteq \Xi_C$ 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 [23] 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.

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

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

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

- $\xi_i \in \Xi_\beta$, *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.

$$\ell_i(\omega_x) = [\xi_i - B_x]^+ \qquad (19)$$

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

$$\ell_i(\omega_x) = \xi_i / B_x \qquad (20)$$

- $\xi_i \in \Xi_\delta$, *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

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

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

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

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.

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

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}
B		1	12	1	———"———
C		1	12	5	———"———
D	D_4	3	4	4	———"———
			4	1	———"———
			4	5	———"———
E	E_4	6	6	4	{**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}
			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 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.

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*.

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 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
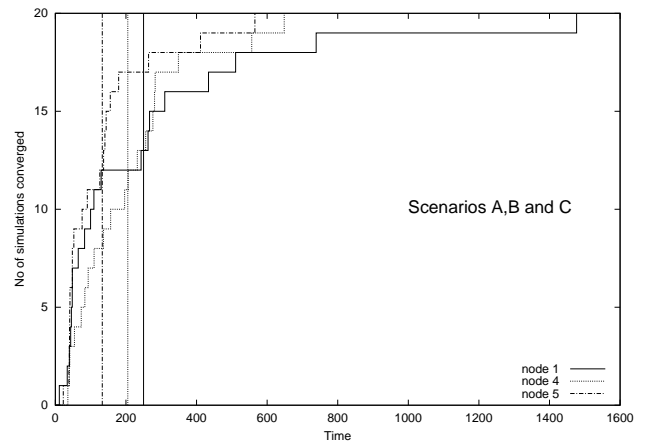

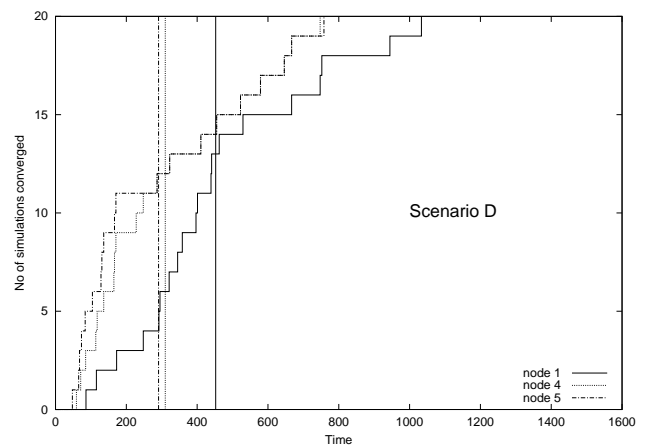Figure 3: Convergence progress for scenario A (node 4), B (node 1) and C (node 5) .

Figure 4: Convergence progress for scenario D.

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 slow-

Table 2: Simulation results

| Scenario	Client resource (node #)	Average convergence time		Average final path cost		Best final path cost		Final equals preferable path	
		Mean	Stdev	Mean	Stdev	Cost	# sim	# sim	$|final - preferable|$
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	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
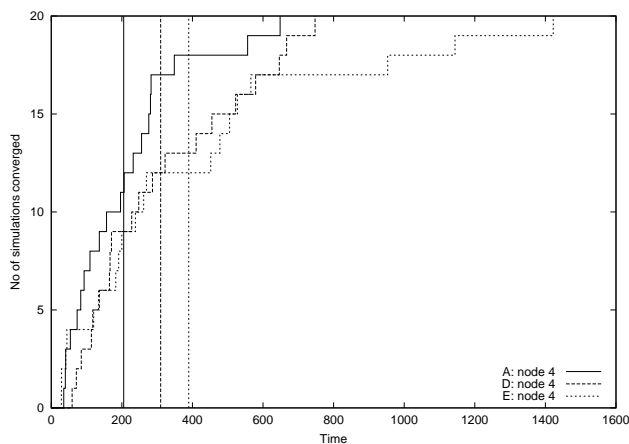
Figure 5: Convergence progress for scenario A, D_4, and E_4
.

est 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. 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 none-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

[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 August 2003.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview," *IEEE Communications Magazine*, vol. 36, pp. 26–37, July 1998.

[23] O. Wittner, P. E. Heegaard, and B. 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.

[24] 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.

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