Collective Intelligence and Priority Routing in Networks

Tony White, Bernard Pagurek, Dwight Deugo

Carleton University, School of Computer Science, Ottawa, Canada K1S 5B6
(arpwhite@scs.carleton.ca, bernie@sce.carleton.ca,
deugo@scs.carleton.ca)

Abstract: This paper describes how biologically-inspired agents can be used to solve complex routing problems incorporating prioritized information flow. These agents, inspired by the foraging behavior of ants, exhibit the desirable characteristics of simplicity of action and interaction. The collection of agents, or swarm system, deals only with local knowledge and exhibits a form of distributed control with agent communication effected through the environment. While ant-like agents have been applied to the routing problem, previous work has ignored the problems of agent adaptation, multi-path and priority-based routing. These are discussed here.

1 Introduction

Networks today have a wide range of applications running on them. Many would assume that making best use of network capacity implies load balancing; however, this is a simplistic assumption and ultimately it is user perception of the quality of service offered by the network that is important. Anyone who has used streaming audio or IP telephony services on the Internet will certainly appreciate this.

Circuit planning in a large network is a hard problem. An off-line or planning solution is possible. In this approach, the set of connections to be created is known in advance and routes for them computed to optimize a fitness function. Typically, the fitness function seeks to balance load across nodes and links in the network and may take account of constraints of the devices themselves and user routing preferences.

On-line approaches are also possible. In traditional networks, routing protocols are often used that attempt to maintain a global view of the network. Several agent-oriented approaches have recently been proposed that appeal to principles drawn from Swarm Intelligence [5], [2], [7] and others. In an on-line approach, agents compute routes for connections in order to optimize their connection routing cost, where cost may represent an aggregate statistic of delay, utilization, reliability and other factors. In these approaches, a global view of the network is not maintained, and we deal only with information that can be measured locally. Swarm approaches are robust with respect to the loss of individual routing agents. Beyond the routing domain, the appeal of swarms of biologically-inspired agents for industrial problem solving has recently been appreciated

[4]. Research into the problems and potential of multiple, interacting swarms of agents is just beginning [8]. This paper builds on prior work by proposing routing solutions for creation of multi-cast routes, in an environment that supports traffic prioritization. It appeals to the SynthECA agent architecture recently proposed [8].

This paper consists of 3 further sections. The next section introduces elements of the SynthECA architecture pertinent to this paper. The following section describes the algorithms used to solve routing problems, and the results of applying them. The paper then summarizes its key messages.

2 SynthECA Agents

Agents in the SynthECA system can be described by the tuple, A = (E,R,C,MDF,m). The important components pertinent to this paper are the ideas of a chemical (C) and a Migration Decision Function (MDF). A detailed description of the architecture can be found in [8].

The chemical concept is used in order to provide communication between agents and to create dissipative fields within the environment. The chemical concept is used to provide communication with, and sensing of, the environment and provides the driving force for agent mobility. A chemical consists of two components, an encoding and a concentration. When the encoding uses the alphabet {1, 0, #} in a string of length *m*, we say that we are using a Binary Array Chemistry of order *m*.

The MDF is a function or rule set that is used to determine where an agent should visit next. The MDF typically uses chemical and link cost information in order to determine the next hop in its journey through the network or may simply follow a hard-coded route through the network. An important consideration in designing an MDF is that it should take advantage of gradients in chemicals that are present in the network. In doing so, agents may take advantage of the actions of other agents. Particular agents may want to move up a gradient (attraction) or down a gradient (repulsion). *The MDF may take advantage of the pattern matching properties of the language used for the chemistry of the system.* Consider a chemical encoding consisting of 2 bits. We might include a term in the MDF consisting of the chemical 1#, where the # symbol matches either a 0 or a 1.

Consider a scenario where an agent has the choice of two links. Link 1 has concentrations Ch(10, 0.1) and Ch(11, 0.7). Link 2 has concentrations Ch(10, 0.5) and Ch(11, 0.6). Thus, an agent moving up a gradient indicated by the 1# pattern would follow link 2 because Ch(1#, 1.1) is sensed for that link. Similarly, an agent moving up the gradient indicated by the 11 pattern would follow link 1. This example is crucially important for the priority discussion later in the paper.

3 Swarm Routing

The swarm algorithm solution to this routing problem relies on the movements of artificial agents on the associated graph designed to make the global shortest path emerge. The communications network is represented in this paper as a weighted graph where the vertices correspond to switching nodes and the edges represent the physical links. When a connection request is made, a colony of agents is created and a Connection Creation Monitoring Agent (CCMA) is created on the source node. The functions of the CCMA are to decide when a path has emerged and when the current path is no longer the shortest path and that path re-planning should occur.

There are three classes of routing-related agent. *Explorer* agents search for a path from a source to a destination. *Allocator* agents allocate resources on the links used in a path. *Deallocator* agents deallocate resources on the links used in a path.

In establishing multiple point-to-point connections, the problem becomes more constrained since the connections consume bandwidth and that after a while, some links might run out of available bandwidth. The extension is quite straightforward, the graph edges have an associated available bandwidth and a condition is added for the agents to use a given edge: it should have enough bandwidth. Every time a path has emerged and a connection has been established the amount of available bandwidth is decreased on every edge of the path thereby adding additional bandwidth constraints to the graph. Three routing problems have been solved. These are described in the next 3 sections.

3.1 Point to Point Routing

For this case, the algorithm is quite straightforward. Explorer agents are created by the CCMA and leave the node in order to explore the network following their local rules.

The explorer agent has two modes of behaviour. If travelling towards its destination, it finds links at each node which the agent has not yet traversed, and which have enough bandwidth available for this connection. It selects a link from this set based on the probability function $p_{ijk}(t)$. Having selected a link, the selected link is added to the tabu list. The cost of the journey so far is updated and then the link to the next node is traversed. An agent whose path cost exceeds a given threshold dies.

At the destination, the explorer agent switches to trail-laying mode. When travelling back to the source node the agent pops the tabu list and moves over the link just popped dropping pheromone at a constant rate proportional to the cost of the route found.

The CCMA at the source node maintains a set of statistics relating to the set of routes that have been found so far in both point-to-point and point to multi-point connections. This is achieved by querying the returning agents (and agents which are sent from the destination node to this node) about the path that they took. This information is maintained by their tabu list. The node records the frequency of agents following a particular route over a given time period (a moving window). It also records details such as the total cost of the

route. A good route is one for which a proportion of agents in the current time window exceeds a specified limit; e.g., 95%. When the limit is exceeded, the node sends out an allocator agent that creates the connection by allocating resources in the network.

Once an allocator agent is dispatched, if it does not succeed in establishing the connection because, for example, another connection used all the available bandwidth, it simply backtracks. In the meantime, explorer agents continue to explore the problem space. The CCMA at the source node may send out an allocator agent again when the path emergence criteria are satisfied, or may choose to delay sending the allocator out again until the problem space settles to a steady state. Chemicals laid down on a link evaporate over time. This is controlled by a constant evaporation rate, r.

3.2 Point to Multi-point Routing

Point to multi-point connections can be regarded as multiple point to point connections starting from the same source node. The only modification to the previous point to point algorithm concerns the allocator. Rather than sending a different allocator for each destination, identical allocators are sent from the source toward the destinations. Only the first allocator passing on the link will allocate the bandwidth, and fan out points are created on bifurcation nodes.

3.3 Cycle (or Multi-path) Routing

Cycle or multi-path routing can be regarded as two node and link disjoint paths (excluding source and destination nodes) that connect a source node to a destination node. The only modification to the original algorithm for the explorer agent is that upon reaching the destination it turns around and finds a path back to the source node that does not use any of the nodes or links used in the outward journey. Paths of this type are frequently constructed for the purpose of fault tolerance. SONET networks are constructed using cyclical paths.

3.4 The Route Allocation Algorithm

There are two phases to the movement of an explorer agent: an outward exploring mode and a backward trail-laying mode. The algorithm used by an explorer agent for a *single* connection is shown below.

```
do:                                          {**}
  Set t:= 0                                  While i > 1
  For every edge (i,j), set S_ij(t) := 0, cr_k := 0     Move to node Tabu_k[i].
  Place m agents on source node.             S_ij(t) := S_ij(t) + ph(cr_k)
```

```
  Explorer agents created at frequency e_f
end
Set i := 1 {tabu list index}
For k := 1 to m do
  Place starting node, s, of the k^th agent in
Tabu_k[i].
Repeat until destination reached:
   Set i := i + 1
   For k :=1 to m do
     Choose next node, p_ijk(t) {eqtn 1}
     Move the k^th agent to node j.
     cr_k = cr_k + C_ij(u)
     If  cr_k > cr_max then
        kill the k^th explorer agent.
     Insert node j in Tabu_k[i].
     At destination go to {**}.
end
     i := i - 1
end
 {At the source node}
 do:
    If the path in Tabu_k is the same as b% of
    paths in PathBuffer then create and
     send  an allocator agent
     if t > T_max then
        create and send an allocator agent
        for shortest path found
end
```

In the algorithm above, the following symbols are used:

$S_{ij}(t)$ is the quantity of pheromone present on the link between the i^{th} and j^{th} nodes,

$C_{ij}(u)$ is the cost associated with the link between the i^{th} and j^{th} nodes at utilization, u.

cr_k is the cost of the route for the k^{th} explorer agent.

$Tabu_k$ is the list of edges traversed.

T_{max} is the maximum time that is allowed for a path to emerge.

PathBuffer is the array of paths obtained by the (up to m) explorer agents.

cr_{max} is the maximum allowed cost of a route.

ph(crk) is the quantity of pheromone laid by the k^{th} explorer agent.

$p_{ijk}(t)$ is the probability that the k^{th} agent will choose the edge from the i^{th} to the j^{th} node as its next hop given that it is currently located on the i^{th} node.

More generally, for multiple simultaneous connection finding, with one chemical used for each connection to be computed, the probability, $p_{ijk}(t)$, with which the k^{th} agent chooses to migrate from its current location, the i^{th} node, to the j^{th} node at some time, t, is given by:

$$p_{ijk}(t) = F_{ijk}(t) / N_{ik}(t), \quad R < R^* \qquad (1.)$$

$$= H_{ij}(t) \text{ otherwise}$$

$$N_{ik}(t) = \Sigma_{l \text{ in } A(i)} F_{ilk}(t) \quad (2)$$

$$F_{ijk}(t) = \Pi_r [S_{ijr}(t)]^{\alpha_{kr}} [C_{ij}(u)]^{-\beta} \quad (3)$$

$$F_{ijk}(t) = \Pi_r [S_{ijr}(t)]^{\alpha_{kr}} [C_{ij}(u)]^{-\beta}, \; j = j^{max} \quad (4)$$

$$= 0 \text{ otherwise}$$

where:
α_{kr}, β are control parameters for the k^{th} agent and r^{th} chemicals for which the k^{th} agent has receptors, $\alpha_{kr} = 0$ if the agent does not have a receptor for the r^{th} chemical,
$N_{ik}(t)$ is a normalization term,
A(i) is the set of available outgoing links for node i,
$C_{ij}(u)$ is the cost of the link between nodes i and j at a link utilization of u,
$S_{ijr}(t)$ is the concentration at time t of the r^{th} chemical on the link between nodes i and j,
R is a random number drawn from a uniform distribution (0,1],
R* is a number in the range (0,1],
$H_{ij}(t)$ is a function that returns 1 for a single value of j, j*, and 0 for all others at some time t, where j* is sampled randomly from a uniform distribution drawn from A(i),
$F_{ijk}(t)$ is the migration function for the k^{th} agent at time t at node i for migration to node j,
j^{max} is the link with the highest value of the product: $\Pi_r [S_{ijr}(t)]^{\alpha_{kr}} [C_{ij}(u)]^{-\beta}$.

3.5 The Agent Creation Algorithm

Due to the pheromone sensitivity of the agents, it is possible for premature convergence of the route-finding algorithm to occur. Sometimes, the agents can be attracted by that trail in such a way that they will not explore other links any more. To avoid being locked into a local optimum, the pheromone sensitivity has to be modified. With lower pheromone sensitivity, the agents are more likely to explore other links. The problem is when to lower the sensitivity and what new pheromone sensitivity to give to the agents.

Each explorer agent encodes its α and β sensitivity values that are used in the calculation of $p_{ijk}(t)$. Initially, these values are selected randomly from a given range. Hence, the population of m agents initially sent out into the network has a range of sensitivity values. When these agents return to the source node, having found a route to a given destination, the route cost, cr_k, is used to update the fitness value, $f(\alpha,\beta,k)$, associated with the (α,β) pair. The equation used to update $f(\alpha,\beta,k)$ is given by:

$f_{new}(\alpha,\beta,k) := f_{old}(\alpha,\beta,k) + \gamma(crk - f_{old}(\alpha,\beta,k))$, $0 < \gamma < 1$.

An agent returning with a lower route cost than the current $f(\alpha,\beta,k)$ will cause $f(\alpha,\beta,k)$ to decrease. However, several agents must return with the same crk value before $f(\alpha,\beta,k)$ approaches cr_k. A discrete space for (α,β) was chosen in order to ensure that updates to $f(\alpha,\beta,k)$ would occur. A discounted feedback mechanism as shown above is required in this system because of the stochastic nature of the search. The same (α,β) encoding may result in several different crk values and $f(\alpha,\beta,k)$ represents the average over all possible

routes found in the network. Clearly, with γ set to zero it is possible to ignore previous searches with a given encoding.

As noted earlier, the source node retains a path buffer that contains m paths. The source node also retains m (α,β) pairs and their associated fitness values. When new agents need to be created and sent out to explore the network, the fitness values are used to create new (α,β) pairs. First, parent (α,β) encodings are selected based upon their $f(\alpha,\beta,k)$ values. The lower the value of $f(\alpha,\beta,k)$, the more likely the (α,β) encoding is to be chosen.

The way we have achieved this is with a Genetic Algorithm-like (GA) process. As stated above, each agent has its own cost and pheromone sensitivity. At the very beginning, all the agents have random sets of parameters that are defined within a given range.

When an agent returns, its set of parameters is stored along with the cost of the route found. Its parameters are linked to the cost of the path found. This cost has the same role as the fitness function of a GA. When creating a new agent, the sets of all of the last returning agents are considered. An intermediate population of parameters is created: each set has a probability of being chosen proportional to its fitness. Some random parameters are automatically added to the population. Given that the encoding is a bit string, the spectrum of parameter values is discrete, a property which is essential for the use of the updating equation for $f(\alpha,\beta,k)$. The negative values allow agents to flee the main trail and therefore to explore new links. If these values are useful they will be stored for future 'breeding', otherwise they will be forgotten. Then the genetic operators such as mutation and crossover are carried out. Finally, agents with the corresponding set of parameters are created and sent out to explore the graph.

This approach differs from a conventional GA in that, in this algorithm, we are trying to *avoid* the convergence of the population because it tends to lead to local optima. This is perhaps closer to work on co-evolving populations because the environment of the agents (the network and its representation, the graph) is modified by their actions. There is considerable inter-play between the pheromone laying activities of one agent with the cost of a path found by another agent and, therefore, the fitness associated with the (α,β) encoding of pheromone and cost sensitivity values.

3.6 Experimental Setup

Two graphs were used during experimental investigation of the adaptive system for the point to point, point to multi-point and multi-path problems. These are shown in Figure 1 and Figure 2. The numbers associated with the edges in these networks represent the costs of the edges at zero edge utilization. Each edge is considered to have a capacity of 63 units. For problem one, the point to point path finding scenario, ten randomly generated traffic profiles were created for all source-destination pairs with bandwidth requirements sampled uniformly from the set {0, 2, 4, 6, 8, 10} bandwidth units. A bandwidth requirement of zero units was taken to mean that no path need be calculated for the

Fig. 1. Experimental Graph

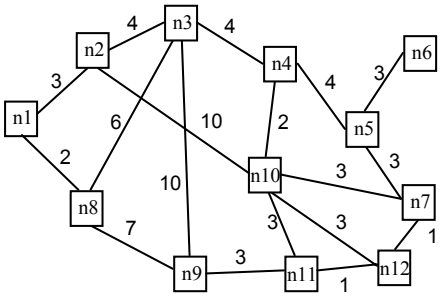
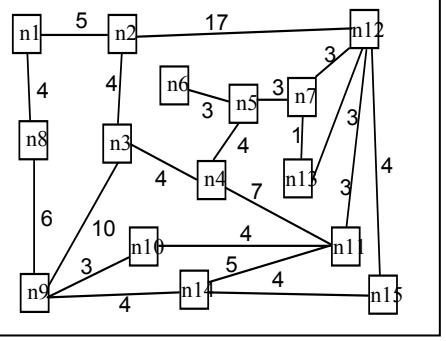


Fig. 2. Experimental Graph



source-destination pair. Paths were calculated such that the utilization of the network increasing by the bandwidth requirements of the traffic as paths emerged. All paths were computed in parallel. Initial network edge utilizations of 0, 30, and 50% were considered in order to test the effects of four different cost functions. For problem three, the same randomly generated traffic profiles were used for experimentation. For problem two, ten randomly generated traffic profiles were created with 2, 3 or 4 destinations. Bandwidth requirements for the point to multi-point requests were identical to problem one.

A population size of 50 was used with path emergence considered to have occurred when 90% of the population follows a given path. A maximum of 100 cycles of the path finding algorithm was allowed before path calculations were stopped and 20 agents per cycle were sent out into the network for path finding. The value of α was allowed to vary in the range -0.25 to 3 and the value of β was allowed to vary in the range -0.125 to 1.5. A total of 16 bits was allowed for the encoding of α and also for β. When adaptive search was contrasted with its non-adaptive counterpart, with constant α and β, values of 2 and 1 respectively were used. These constant values were found to be a reasonable compromise for path finding. A value of 10 was chosen for the constant of proportionality for the quantity of chemical to be laid. An indirect representation was used with mapping of bit strings into floating point values in the above ranges in such a way as to cover the ranges uniformly. Values of 0.8 and 0.01 were used for the probabilities of crossover and mutation respectively. Single point crossover was used as the crossover operator.

3.7 Results

Comprehensive results are reported elsewhere [8]. However, comparing the results of using static routing to the adaptive, swarm-based routing described above, the swarm system routed 18% more traffic in the network. We have observed that the algorithm with adaptive chemical sensitivity values is able to adapt to a new situation *much* more quickly when using these adaptive parameters. The time needed to discover the new path was

typically 30% lower (and often much, much lower) when compared to an algorithm using static parameters. When individual components were caused to fail, traffic was quickly re-routed to remaining network paths.

3.8 Multi-priority Routing

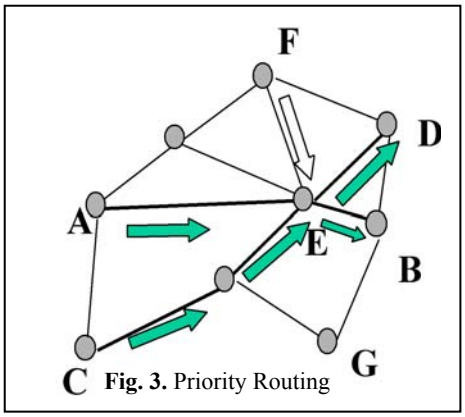


Fig. 3. Priority Routing

In many networks we would like to route application traffic with varying priorities. In the model proposed here, the idea is to move existing traffic to longer routes when a higher priority connection request enters the system. Consider, for example, the scenario as represented in Figure 3. Here, two routes have already been computed: AB and CD. These are indicated with the dark arrows. A new connection request now enters the system (FG) as shown by the white arrow. The idea, then, is to have the AB and CD explorer agents (which continue to explore the network even after route allocation) to recognize that the environment has changed and to deallocate the existing route and find one that avoids the higher priority traffic.

In adding multi-priority routing, the explorer algorithm proceeds as before, with one modification. The MDF of the explorer agent does not sense the lower priority connection pheromones but has two receptors instead of one used in the basic routing system. In fact, we use a two class pheromone system. The first class of pheromones has the same meaning as in the earlier sections of this paper, i.e., it is a member of the Binary Array Chemistry of order N. The second class of pheromones is a different length chemical, i.e., it is a member of the Binary Array Chemistry of order N^p, where $N^p \neq N$. In this way, the utilization (and therefore the cost) of a link appears lower to a higher priority connection when compared to a lower priority connection. This requires that chemicals have a well-defined and standardized encoding. As before, all explorer agents find the shortest path using the algorithms described in the previous sections and, upon path emergence, an allocator is sent out to allocate resources in the network. The allocator differs here from the previous design. In a multi-priority system, the allocator deposits pheromone on its way back from the destination having allocated resources for the connection. The reason for the allocator depositing pheromone on its way back from the destination is that we want to ensure that resources have been allocated before communicating the existence of the new connection to other connections in the network. The allocation pheromone, or a-chemical, *is* sensed by lower priority connection explorer agents. Upon return of the allocation agent, the lower priority explorer agents, or lp explorers as we shall refer to them, begin to experience higher costs for links on their paths that have had resources

allocated for the higher priority connection. With the added costs, lp explorers will deposit smaller quantities of their connection pheromones, indicating a reduced confidence in the path. If confidence in the path is sufficiently reduced, i.e., it is no longer the shortest path, the majority of lp explorers will begin to follow the new shortest path. At this point, the CCMA will send out an allocation agent for the new shortest path and a deallocation agent for the old shortest path. Hence, the higher priority connection has forced the movement of lower priority connections off of its path.

The above is probably best illustrated with an example. Consider again the network and connections shown in Figure 3. Let the order of the connection and priority chemistries be 8 and 4 respectively. The latter implies that 4 priority levels exist. Let all edges in the network have unity cost, i.e., $C_{ij}(u) = 1$, except edges FD and GH which have $C_{ij}(u) = 2$. The paths indicated by dark arrows for connections AB and CD are then the shortest paths and will be found by explorer agents. Now, consider the introduction of the high priority connection FG. The shortest path for this connection is FE-EB-BG. Assuming that the allocator for this connection drops 3 units of the a-chemical, lp explorer agents for the AB connection will begin to experience higher costs for the AE-EB path, now 4, which makes the path longer than the shortest path, now AE-ED-DB with cost 3. After some time, the CCMA agent for the AB connection will observe that the majority of its explorer agents follow the new shortest path and will send deallocation and allocation agents out into the network in order to move the connection to the new shortest path.

Obviously, the above example has been constructed to demonstrate the path moving algorithm. However, it is sensitive to the quantity of a-chemical deposited in the network by the allocator agent for the higher priority connection. If the concentration of a-chemical is too low, the lower priority connections sharing links with the high priority connection will not be forced to move. This limitation can be overcome by having the CCMA monitor the quality of service associated with the connection. If the measured quality of service falls below the required service level agreement, an agent is sent out into the network that deposits higher concentrations of the a-chemical, thereby increasing the effective cost of the links as seen by the lp explorer agents.

Returning to the above example, assume that the high priority connection allocation agent deposited only 1 unit of the a-chemical. In this case, the cost of the path for the AB connection does not appear to change, and no re-routing occurs. The CCMA for the high priority question, seeing its quality of service to be below the agreed value, sends an agent out into the network to deposit a further quantity of the a-chemical, say 2 units. The rate of increase of a-chemical concentration achieved by sending out multiple agents that deposit a-chemical we call the *priority momentum factor*. This is described in [7,8]

At this point, the quantity of a-chemical on the high priority connection links is 3, as before, and re-routing consequently takes place. With a lower priority momentum factor the required number of agents depositing a-chemical to achieve re-routing will be higher, but re-routing will inevitably occur when the route is made “expensive enough” as seen by the lp explorer agents.

4 Conclusions

This paper has shown that multi-agent swarm techniques can solve complex routing problems on networks. The main strengths of the algorithm are its robustness, the simple nature of the agents, and that it continues searching for new solutions even if a very good one was found. The integration of Genetic Algorithms with the basic Swarm Algorithm has improved the speed of convergence of the routing algorithm. The generality of the SynthECA architecture was validated when adding multiple priority levels to the algorithm proved straightforward. The algorithms described here work best on networks where connections are long lived [3]. For the algorithm to operate on a real network, several things are required. First, agent mobility must be supported. Second, chemical concentrations storage is needed. Finally, on the source node, the CCMA must be present in order to determine whether a route has emerged. All of the above points can be addressed with the application of Java, a mobile agent framework and the use of many of the Virtual Managed Component (VMC) concepts found in [6]. Besides these requirements, consideration must be given to the amount of bandwidth taken up by routing agents. Research addressing resource usage has been forthcoming [1].

References

1. Boyer, J., Pagurek, B., White, T. 1999, Methodologies for PVC Configuration in Heterogeneous ATM Environments Using Intelligent Mobile Agents. In Proceedings of the 1st Workshop on Mobile Agents and Telecommunications Applications (MATA '99).
2. Di Caro G. and Dorigo M. 1998, AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317-365.
3. Pagurek B., Li Y., Bieszczad A., and Susilo G. 1998, Configuration Management In Heterogeneous ATM Environments using Mobile Agents, Proceedings of the Second International Workshop on Intelligent Agents in Telecommunications Applications (IATA '98).
4. Parunak H. Van Dyke 1998, Go to the Ant: Engineering Principles from Naturally Multi-Agent Systems, In *Annals of Operations Research*. Available as Center for Electronic Commerce report CEC-03.
5. Schoonderwoerd R., Holland O. and Bruten J. 1997, Ant-like Agents for Load Balancing in Telecommunications Networks. Proceedings of Agents '97, Marina del Rey, CA, ACM Press pp. 209-216.
6. Susilo, G., Bieszczad, A. and Pagurek, B. 1998, Infrastructure for Advanced Network Management based on Mobile Code, Proceedings IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana.
7. White T., Pagurek B. and Oppacher F. 1998, Connection Management using Adaptive Mobile Agents, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98).
8. White T. 2000, SynthECA: A Society of Synthetic Chemical Agents, Ph.D. diss., Carleton University.