Towards Multi-Swarm Problem Solving in Networks

Tony White, Bernard Pagurek
Systems and Computer Engineering, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6
email: {tony, bernie}@sce.carleton.ca

Abstract

This paper describes how multiple interacting swarms of adaptive mobile agents can be used to solve problems in networks. The paper introduces a new architectural description for an agent that is chemically inspired and proposes chemical interaction as the principal mechanism for inter-swarm communication. Agents within a given swarm have behavior that is inspired by the foraging activities of ants, with each agent capable of simple actions and knowledge of a global goal is not assumed. The creation of chemical trails is proposed as the primary mechanism used in distributed problem solving arising from self-organization of swarms of agents. The paper proposes that swarm chemistries can be engineered in order to apply the principal ideas of the Subsumption Architecture in the domain of mobile agents. The paper presents applications of the new architecture in the domain of communications networks and describes the essential elements of a mobile agent framework that is being considered for its implementation.

1. Introduction

The notion of complex collective behavior emerging from the behavior of many simple agents and their interactions is central to the ideas of Artificial Life [15]. Nature provides us with several examples of social systems comprising individuals exhibiting simple behaviors while the society exhibits complex problem solving capabilities.

Naturally occurring social systems provide considerable inspiration for artificial systems that display emergent behavior. Such systems promise to provide guiding principles for, and engineering solutions to, distributed systems management problems found, for example, in communications networks.

In this paper, we describe the essential principles of Swarm Intelligence (SI) and, in particular, how an understanding of the foraging behaviors of ants [1] has led to novel, distributed control and management in communications networks.

This paper consists of six subsequent sections. In the next section, a brief overview of Swarm Intelligence and Ant Colony search is presented. The paper continues with a presentation of our multi-swarm architecture. The following section describes a scenario drawn from the communications domain where a multi-swarm architecture has been used. Implementation issues, both simulation and using a mobile agent toolkit, are then briefly described. A summary of our findings is then provided and the paper ends with conclusions and future work.

2. Swarm Intelligence

Swarm Intelligence [2] is a property of systems of unintelligent agents of limited individual capabilities exhibiting collectively intelligent behavior. An agent in this definition represents an entity capable of sensing its environment and undertaking simple processing of environmental observations in order to perform an action chosen from those available to it. These actions include modification of the environment in which the agent operates. Intelligent behavior frequently arises through indirect communication between the agents, this being the principle of stigmergy [12]. It should be stressed, however, that the individual agents have no explicit problem solving knowledge and intelligent behavior arises (or emerges) because of the actions of societies of such agents.

Individual ants are behaviorally simple insects with limited memory and exhibiting activity that has a stochastic component. However, collectively ants manage to perform several complicated tasks consistently well.

In the ant problem solving behavior documented; e.g. [14], two forms of stigmergy can be inferred. *Sematectonic* stigmergy involves a change in the physical characteristics of the environment. Ant nest building is an example of this form of communication in that an ant

observes a structure developing and adds to it. The second form of stigmergy is *sign-based*. Here, something is deposited in the environment that makes no direct contribution to the task being undertaken but influences subsequent task related behavior.

Sign-based stigmergy is very well developed in ants. Ants foraging for food lay down quantities of pheromone (a highly volatile hormone) marking the path that it follows with a trail of the substance. An isolated ant moves essentially at random but an ant encountering a previously laid trail will detect it and decide to follow it with a high probability and thereby reinforce it with a further quantity of pheromone. The collective behavior which emerges is a form of autocatalytic behavior where the more the ants follow the trail the more likely they are to do so.

The use of ant foraging behavior as a metaphor for a problem-solving technique is generally attributed to Dorigo [9]. It is considered central to our work. However, since Dorigo's early work on the Travelling Salesman Problem (TSP) and Asymmetric TSP, the technique has been applied to several other classes of problem. These include the Quadratic Assignment Problem (QAP) [16, 22], graph coloring [7], vehicle routing [6] and, as we shall see in the later sections, communications network routing.

3. Agent System Architecture

In our system, ant-like agents solve problems by moving over the nodes and links in a network and interacting with chemical messages deposited in that network. Chemical messages have two attributes, a label and a concentration. These messages are the only medium of communication used both between swarms and individual swarm agents. Data and chemicals are considered synonymous in our system.

Agents in our multi-swarm system are of limited intelligence; i.e. they belong to the 'lightweight' category of agents, and are capable of only simple behaviors. Such agents are reactive in nature and have the ability to sense and modify their environment locally. Our agents stand in stark contrast to agents supporting the Belief-Desire-Intention (BDI) model [19]. However, we freely acknowledge the desirable nature of hybrid reactive-reflective architectures such as the Touring Machine architecture [10] and, in fact, our lightweight agents interact with stationary agents on platforms used for management and planning in our networks. Having the capability for mobility, ant-like agents are potentially able to modify local environments on network elements (or components) in the entire network that they inhabit. Agents communicate locally, when co-resident on a node and only through their local chemical environment.

Agents in our system can be described by the tuple, $\mathcal{A}=(\mathcal{E},\mathcal{R},\mathcal{C},\mathcal{MDF},\mathcal{m})$. They have a uniform architecture consisting of five components:

- emitters ($\mathcal{E}$),
- receptors ($\mathcal{R}$),
- chemistry ($\mathcal{C}$),
- a migration decision function ($\mathcal{MDF}$),
- memory ($\mathcal{m}$)

3.1. Emitters

The emitters associated with an agent are used to generate chemicals that are deposited where the agent is currently located. Using ants and their foraging behavior as an example, pheromones are laid down as the ant returns from searching for a source of food. Emitters have an associated Emitter Decision Function (EDF) which is used to decide the rate of production of an emitted chemical. The emitted chemical is digitally encoded, having an associated pattern that uses the alphabet {1, 0, #}. This encoding has been inspired by those used in Genetic Algorithms [11] and Classifier Systems [13]. The hash symbol in the alphabet allows for matching of both one and zero and is, therefore, the "don't care" symbol. A chemical encoding including one or more "don't care" symbols can be thought of as a generalized chemical or an instance of several classes of chemical.

The function of an emitter is to alter the local environment inhabited by the agent. Using the above alphabet it is possible, for example, for an agent to generate a digital chemical with the encoding 1#01 which will be sensed by an agent with a 1101 receptor and by an agent with a 1001 receptor.

An emitter can be either on or off depending upon its internal state; i.e. the concentrations of chemicals stored within agent memory.

3.2. Receptors

The receptors associated with an agent are used to sense chemicals that are present in the agent's local environment and chemical changes that occur in it. Using, once again, ants and their foraging behavior as an example, pheromones are sensed by the ant as it searches for a source of food. Receptors have an associated Receptor Decision Function (RDF) that is used to determine the sensitivity to the chemical in question and it is possible to associate actions with a receptor. The sensed chemical is digitally encoded, once again having an associated pattern that uses the alphabet {1, 0, #}. It is possible, therefore, to

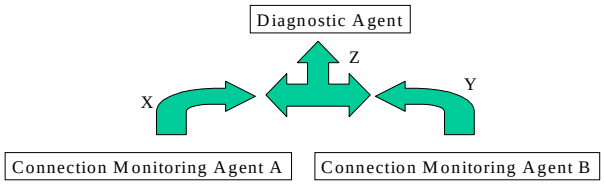
engineer wide spectrum sensors that detect many chemicals. For example, a receptor engineered to sense the encoding 10## will be able to detect the chemicals having the 1000, 1001, 1010 or 1011 encoding. Generally, a receptor having n positions in the encoding with the hash symbol will be able to detect 2^n chemicals. Like an emitter, a receptor can be either on or off depending upon its internal state.

3.3. Chemistry

The chemistry associated with an agent is the set of chemical reactions that can occur within the agent. While the reaction set is limited to, at most, two reactants or products, larger reactions can be synthesized by building chains of these five types of reaction. There are five types of reaction that can occur within an agent. These are shown below.

$$X \rightarrow \text{'nothing'}$$
$$X + Y \rightarrow Y$$
$$X + Y \rightarrow Z$$
$$X + Y \rightarrow X + Z$$
$$X + Y \rightarrow W + Z$$

The first reaction can be thought of as evaporation of a chemical. An example of such a reaction would be the evaporation of pheromone from an ant trail. The second type of reaction is the catalytic breakdown of a chemical, with Y representing the catalyst. An example of such a reaction might be a parasitic interaction between two types of agent such as could be observed when one ant is trying to throw another 'off the scent' when competing for finding a path to a given food source. Another example, this time from the telecommunications domain, is the scenario where an agent representing higher priority traffic reduces the concentrations of lower priority traffic' pheromones (that are used to mark a given route) in order to have the lower priority traffic find an alternate route. The third reaction type represents the fusion of two chemicals and it is this type of reaction that we envisage being used to communicate information from one layer of a multi-swarm hierarchy to another. This type of reaction provides a mechanism which multi-swarm systems could use to implement Subsumption Architectures [4, 5]. This is shown diagrammatically in Figure 1. Figure 1 shows two connection monitoring agents that have quality of service monitoring for a specific connection as their primary responsibility. They detect decreasing quality of service for a shared network resource on, say, a link. As a result, they lay down quantities of X and Y that indicate an increasing level of dissatisfaction with the quality of the connection. A diagnostic agent encoding a type three reaction has one or more receptors that allow for the detection of X and Y, allowing for the generation of Z.



Monitoring agents A and B lay down increasing quantities of pheromone with decreasing quality of service on a given link. Diagnostic agent senses increasing levels and initiates diagnostic activity when threshold exceeded.

Figure 1: Type 3 Reaction Example

The fourth reaction type represents a catalytic reaction where one chemical is converted to another but *only* in the presence of a mediating chemical, the catalyst. This type of reaction can be thought of, in computational terms, as providing a conditional construct where, only if we have a certain confidence in a given state can we perform a specific transformation of one chemical to another. The fifth reaction type is the most general in that the two reactants are converted to two products that are distinct from the reactants. Depending on whether a given chemical is part of one swarm layer or the swarm layer above it, the five types of chemical reaction can be considered as providing both inhibitory and excitatory stimuli to the upper swarm layer. For example, Figure 1 can be viewed as providing an excitatory stimulus to the fault detection swarm layer considered being above the connection-monitoring layer within our multi-swarm architecture.

All of the reactions use digitally encoded chemicals, i.e. all chemicals use the {1, 0, #} alphabet. Hence, reactions of the form below are supported[1].

$$011 + 1\#0 \rightarrow 1\#0$$

The above reaction, an example of a type two reaction, allows for the catalytic breakdown of the 011 chemical to occur via two catalysts, namely 110 or 100. Unification occurs between chemicals on a bit-by-bit basis and is carried through from reactants to products, i.e. the "don't care" symbol in a given position within two reactants or between reactants and products can be either 1 or 0 in a single reaction. Using the above reaction as an example, four possibilities exist. These are shown on the next page.

[1] Note that the concentrations of the two chemicals have been assumed to be one in this example.

$$011+110 \rightarrow 110$$
$$011+100 \rightarrow 100$$
$$011+110 \rightarrow 100$$
$$011+100 \rightarrow 110$$

The first two reactions are implied, with the "don't care" symbol being unified to 1 and 0 respectively. However, the latter two reactions are not implied, as in both reactions the "don't care" symbol in the second position has to unify to both 1 and 0.

All of the reaction types have an associated reaction rate, i.e. a measure that determines the speed with which the reaction can occur. Reaction rates are temperature dependent, with the dependence characterized by Arrhenius' equation. Temperature and energy are considered to be essentially the same in our system. Consequently, a unique chemical encoding that can be generated by chemical reactions (as any other) has been chosen to represent temperature. By using the same representation for energy and chemicals, endothermic and exothermic reactions can be used to cool and heat the system respectively. Endothermic reactions are characterized by a decrease in temperature and, as such, reaction types 2, 3, 4 and 5 can represent this type of reaction. This is shown in the example reactions below, where T is meant to represent the energy consumed by the reaction, i.e. it appears on the left-hand side of the reaction.

$$T + Y \rightarrow Y \qquad X + Y \rightarrow T$$
$$T + Y \rightarrow Z \qquad X + Y \rightarrow X + T$$
$$X + T \rightarrow X + Z \qquad X + Y \rightarrow W + T$$
$$X + T \rightarrow W + Z$$

Similarly, reaction types 3, 4 and 5 as is shown in the example reactions above may represent exothermic reactions. In these reactions T once again represents the energy generated by the reaction, i.e. it appears on the right-hand side of the equation.

Changing the *local* temperature of the system changes the degree to which swarms interact. Low temperatures see little interaction between swarms whereas high temperatures see high levels of interaction. It should be stressed that temperature is a local characteristic of the environment and no attempt is made to make this information globally available. The temperature chemical can be thought of as a local control parameter limiting or promoting agent interaction, i.e. providing inhibitory or excitatory stimuli within the multi-agent system.

In our system, all agents are provided with a temperature receptor by default, thereby being able to sense the local temperature. However, this need not be the case, one could imagine a design where internal and external agent temperatures were maintained.

3.4. Memory

The memory associated with an agent stores the chemicals and their concentrations that are held internally to the agent. It is the holder of the state of the agent. Symbolic information can also be stored in memory; however, the agent alone may use this type of information. These types of agent cannot communicate such information to the environment. Only chemicals for which emitters or receptors are not provided are stored within agent memory.

3.5. Migration Decision Function

The Migration Decision Function (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. This latter migration strategy is often referred to as an itinerary in the mobile agent literature. Alternatively, when migrating, the agent may use the default migration node available to it.

3.6. Agent Operation

While the chemistry of an agent appears similar to a classifier system at first glance, it is only superficially so. Firstly, the agent chemistry is fixed and no Bucket Brigade algorithm [11, for example] or similar apportionment of credit scheme is intended to operate in order to modify the chemistry. An agent's chemistry is *fixed*, having been engineered in order to achieve a given function within the mobile agent subsumption architecture. Secondly, an agent operates continuously and all reactions operate in parallel in order to modify the local environment. This is quite different from the way in which message processing occurs within a classifier system. Upon arrival at the node, an agent registers interests in particular chemicals. Chemical concentration changes caused by agent chemical reactions are communicated to the local environment for which the agent has emitters. These concentration changes are then automatically communicated to other agents resident at the node as a result of their registration for notification of chemical concentration changes. Once the agent has performed its task on a particular node; e.g. measurement of quality of service of a connection or simply sensing the concentration of a specific chemical, the MDF is invoked in order to determine the node to migrate to in the network. No fixed residency time is assumed; some agents will remain at a node for long periods of time, others will not.

4. Scenario

As an example of a multi-swarm interacting system moving on a network we have chosen to investigate route finding, maintenance and fault detection in a communications network. In our environment we have a completely distributed view of the network. Such a view is *highly* desirable as it makes management of these networks easier and scalable.

In our system, drawn from the domain of transmission networks, we are attempting to create connections between nodes in the network, monitor them for quality of service degradation and diagnose faults when they occur. It is assumed, and this can be the case, that a network manager does not exist and so no global view of the network is maintained. Consequently, a distributed route finding solution as represented by the Ant Search class of algorithms is a good candidate for route finding.

To date, three applications of the ant metaphor in the domain of routing have been documented [22] (used in this paper), [18] and [8]. The work reported in [18] embraces routing in the circuit switched networks while [8] deals with packet switched networks. Both [18] and [8] propose the control plane as the domain in which their systems would most likely operate. Di Caro and Dorigo [8], in particular, provide compelling experimental evidence, based upon simulation, as to the utility of AntNet in the network routing problem domain by comparing ant-based routing with the current and proposed routing schemes used in NSFNET. The scenario described here is somewhat different and applies to a management context such as is found in a Synchronous Optical Network (SONET) transmission network.

For the context of this paper, we are interested in forming a connection between a source and one or more destinations for the purpose of creating a link in a logical network. It should be noted that this path may be protected, i.e. two node and link diverse paths may exist between a given source and destination. This (possibly protected) path, in turn, can be used as a resource, a link, in the next logical layer.

4.1. Agent Classes

In our system we have three agent classes related to route finding, one class concerned with connection monitoring and one class which has the function of detecting network poor quality of service conditions. These will now be described in terms of the ($\mathcal{E}, \mathcal{R}, \mathcal{C}, \mathcal{MDF}, m$) formalism introduced earlier.

The three agent classes related to route finding are explorers, allocators and deallocators. The function of an explorer agent is to find a path from a given source to a specific destination. The metaphor used to describe the behavior of explorer agents is that of ants foraging for food. Explorer ants possess a single emitter (e) and three receptors (r_1, r_2, r_3). The emitter and receptor r_1 are both tuned to a single chemical or pheromone (T). The receptor r_2 is used to measure the costs of links in the network (C). The receptor r_3 is used to detect the perceived quality or reliability associated with links in the network (Q).

The explorer agent has two distinct modes of operation. When moving towards the requested destination, the emitter is turned off and the receptors are used to detect the connection-specific chemical and link costs respectively. The agent's memory is used to store the links traversed by the agent. When moving back towards the source node having reached the required destination, the receptors are turned off and the emitter is turned on. A single chemical reaction (e) is defined for the explorer agent. This reaction allows for the generation of the pheromone used to reinforce an emerging path. The mdf used by the agent is defined by a series of equations that specify the probability with which a given link will be used for agent migration. The probability with which an explorer agent (k) chooses a node j to move to when currently at the i^{th} node at time t is given by:

$$p_{ijk}(t) = [T_{ijk}(t)]^{\alpha}[C(i,j)]^{-\beta}[Q_{ijk}(t)]^{-\gamma} / N_{ik}(t)$$

$$N_{ik}(t)=\Sigma_{j\,\varepsilon(S(i)-Tabu_k)}[T_{ijk}(t)]^{\alpha}[C(i,j)]^{-\beta}[Q_{ijk}(t)]^{-\gamma}$$

where α, β and γ are control constants and determine the sensitivity of the search to pheromone concentration, link cost and component quality respectively. N_k is simply a normalization factor that makes p_{ijk} (t) a true probability. S(i) is the set of integers, {n} such that there exists a link between the i^{th} and n^{th} nodes. $T_{ijk}(t)$ is the quantity of pheromone present on the link between the i^{th} and j^{th} nodes for the k^{th} agent at time t. C(i,j) is the cost associated with the link between the i^{th} and j^{th} nodes. Q_{ijk} (t) is the quality or reliability measure associated with the link and the j^{th} node for the k^{th} agent at time t. $Tabu_k$ is the list of edges traversed by the k^{th} agent. The C function is meant to represent the cost to the user for consuming bandwidth on a given link while the Q function represents the confidence that we have in the various components involved in the connection being able to transport data effectively.

When explorer agents return to the source node, a decision is taken as to whether a path has emerged. Essentially, if a given percentage of the last n agents have followed a single path then path emergence is considered to have occurred. Once emerged, an allocator agent is sent into the network in order to create the connection.

The allocator agent has no emitters or receptors. It has a simple memory that stores the route that has emerged. An allocator operates in two modes: forward and backward. The allocator agent has a simple *mdf* that simply pops the first entry from the list of links used in the route that is stored in memory. The agent allocates resources for the connection at each node in forward mode. In backward mode the allocator performs no action at each node.

A deallocator agent has an identical ($\mathcal{E}$, $\mathcal{R}$, $\mathcal{C}$, $\mathcal{MDF}$, m) description to that of an allocator agent. The only difference between the two agent types is the action performed at each node when in forward mode. A deallocator agent releases resources in forward mode in contrast to the action performed by the allocator agent and is sent when confidence in the existing route falls below a given threshold, the connection is no longer required or the route is no longer viable. This might be due to component failure, for example.

Further details regarding the connection allocation algorithm, explorer, allocator and deallocator agent behaviors can be found in [22].

Evaporator agents also circulate within the network. The function of these agents is to evaporate chemical concentrations relating to connection finding. They are equipped with a single receptor capable of sensing all connection-related chemicals. They implement a type one reaction in order to effect chemical evaporation. Evaporator agents are required in order to ensure that we do not "greedily" choose the first path found but allow a balance of "exploration and exploitation" to occur in route finding. Evaporation agents are examples of agents where the "don't care" symbol is used in the emitter/receptor description. Evaporator agents have an *mdf* that allows them to cycle through all nodes in the network in a periodic fashion.

Explorer agents continue to search for better routes through the network even after a connection has been set up. Also, once set up, the end-to-end quality of service for the connection is monitored from the source node. When significant changes in quality of service are observed, a monitoring agent is sent out into the network in order to modify the Q values for the components used in the connection. Monitoring agents have either one receptor (r_i) or one emitter (e_i). If the quality of service of the connection has increased, the monitoring agent with a receptor is sent. If the quality of service of the connection has decreased, an agent with an emitter is sent. The monitoring agent's emitter generates the "quality of service" chemical, or q-chemical, using a single chemical reaction in situations where quality of service has decreased and evaporates existing concentrations of q-chemical when quality of service has significantly improved. The receptor senses the q-chemical. The monitoring agent has a simple *mdf* that simply pops the first entry from the list of links used in the route that is stored in memory.

The final agent type in the current system design is the fault detection agent. Fault detection agents circulate through the network and monitor the q-chemical concentrations on nodes and links. They have a single receptor (r_i) and no emitters. Fault detection agents do not have an associated chemistry; i.e. they are merely observers of network state. The *mdf* associated with fault detection agents is probabilistic in nature and is given by the equation:

$p_{ij}(t) = Q(i,j) / \Sigma_k Q(i,k)$ for f% of the time and random otherwise.

Random migrations are made for (100-f)% of the time in order to ensure that the entire network is reached in reasonable time. A probabilistic choice, based upon Q values, is made for f% of the time in order to revisit parts of the network that are experiencing poor quality of service. It should also be noted that oscillation between two high Q components is explicitly prevented, i.e. a fault detection agent cannot return to a previously visited network element for t migrations. This list of tabu elements is stored in agent memory.

The function of a fault detection agent is to observe components with high Q values. When the observed Q value exceeds a threshold value, the agent initiates diagnostic activity by executing rules associated with r_i.

5. Implementation

A Smalltalk simulation has been built for the scenario described in section 4. This simulation is being used to investigate the interaction of the many parameters that characterize the system; e.g. reaction rates, number of agents, agent generation frequency and several others.

A Java-based mobile code infrastructure [20, 17] is being extended to support our architecture, see [24]. The view presented in [3] is that network elements of the future will be Java-enabled and that Java is an important enabling technology for intelligent and active networks of the future. The Perpetuum Mobile Procura (PMP) toolkit has components for migrating, authenticating, instantiating and running mobile agents and provides a number of communication mechanisms for local and remote agent communication. The PMP toolkit defines a Virtual Managed Component (VMC) that forces mobile agents to talk to managed resources indirectly. In our system, the VMC is used to access network component resources and to store chemicals and their concentrations. A simple dictionary is being used for this purpose and a 'notification of concentration change' service is provided by an observer/observable mechanism.

6. Results and Discussion

The authors' research [22, 24] provides experimental evidence that the basic system described in section 4 can effectively compute routes and plan connections in a network. While only simulated results are available, the system has demonstrated that routing solutions to the point-to-point, point-to-multi-point and protected path problems (a problem related to shortest Hamiltonian cycle) for a variety of graph topologies can be effectively computed. The results in [22] indicate that 15% fewer blocked connections are typically observed when comparing standard shortest path routing to the ant-like agent routing described.

Some care has to be taken in the choice of system parameters. Our research is ongoing in the area of self-adaptation of system parameters; e.g. chemical and link cost sensitivities and early results for this activity are reported in [24]. We are investigating the sensitivity of our swarm systems to the number of agents engaged in problem solving as well as considering the effects of noise; i.e. unreliable agent knowledge.

As connection quality of service changes, connections are dropped and new routes quickly found with traffic rapidly moving away from regions of the network that have proven unreliable. Further, the fault detection agent, detecting q-chemicals from multiple connection monitoring agents, quickly identified the faulty components within the network. A diagnostic example is shown in Figure 2.

In Figure 2, two connections are defined, one from A to B and another from C to D. Both connections experience poor quality of service and use monitoring agents to drop q-chemical in the network. The numeric labels on the nodes and edges represent the concentrations of q-chemical for that component. As can be seen in Figure 2, node E sees twice the q-chemical as it is the common element for the two paths. The fault detection agent therefore initiates diagnostic activity on this component. Further details on the use of this architecture and its implementation for diagnosis can be found in [23].

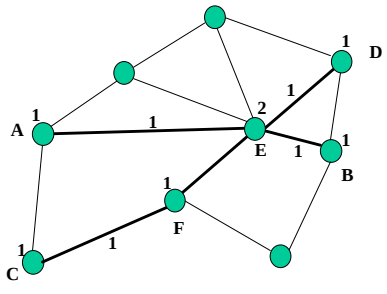


Figure 2: Example Diagnosis

Hill climbing in the space of q-chemical as the primary strategy for fault localization leads the fault detection agent to 'zero in' on faulty components far more quickly than random search. Consider the situation shown in Figure 2. Imagine that there is a single fault detection agent present at A. If we allow migration based only on q-chemical concentration, the agent is forced to move to E; i.e. the detection agent reaches the faulty node in a single move. Consider random migration as an alternative policy. Using this policy, the agent has only a $1/3^{rd}$ probability of reaching E directly and has a far higher expected number of moves before it reaches the faulty component. We have experimented with a number of small graphs (10-20 nodes) with a variety of connection patterns and have found the hill climbing strategy to take less than $1/4^{th}$ of the number of moves that a random strategy would require.

While the simple example described above demonstrates the utility of chemical interference – in this case constructive – and a hill climbing migration strategy, it should be noted that a simple detection mechanism based upon a concentration threshold leads to false diagnoses. We are, therefore, investigating the use of Reinforcement Learning as a means of learning the correct diagnosis for a network state. In these learning systems, a vector of q-chemical concentrations on the node and its links represents state. Referring to Figure 2 once again, the state vector for node E would be (2,1,0,0,1,1,1), where we start with the concentration of q-chemical on the node followed by the links beginning with the link from A to E and moving in a clockwise direction. Feedback to the learning system is the result of whether corrective action resulting from diagnosis improves the state of the system; i.e. q-chemical concentrations are reduced. No change in q-chemical concentration implies a misdiagnosis. Results of this work will be communicated in a future publication.

7. Conclusions and Future Work

This paper has provided a formal description of a multi-agent system that relies on Swarm Intelligence and, in particular, trail laying behavior in order to solve problems in a communications network. We have demonstrated how fault detection can arise as a result of trail laying behavior of simple agents and we have proposed the use of ideas from Subsumption as guiding the design of multi-swarm systems.

While the ideas presented in this paper are conceptually appealing, considerable work remains to analyze the utility of the approach. A great deal of experimentation is currently underway. A theoretical investigation has also begun. It is hoped that results based on Holland's basic

theorems on the calculus of symbol systems[2] may well provide guiding principles and insights for the design of systems such as ours.

Acknowledgements

We would like to acknowledge the support of the Telecommunications Research Institute of Ontario (TRIO) and the National Science and Engineering Research Council (NSERC) for their financial support of this work.

References

1. Beckers R., Deneuborg J.L. and Goss S., Trails and U-turns in the Selection of a Path of the Ant Lasius Niger. In J. theor. Biol. Vol. 159, pp. 397-415, 1992.
2. Beni G., and Wang J., Swarm Intelligence in Cellular Robotic Systems, Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy, 1989.
3. Bieszczad, A. and Pagurek, B., Network Management Application-Oriented Taxonomy of Mobile Code, Proceedings of the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998.
4. Brooks, R.A., Achieving Artificial Intelligence Through Building Robots, A.I. Memo 899, MIT A.I. Lab, 1986.
5. Brooks, R.A., Intelligence Without Representation, Artificial Intelligence, Vol. 47, pp. 139-159, 1991.
6. Bullnheimer B., R.F. Hartl and C. Strauss, Applying the Ant System to the Vehicle Routing Problem. 2nd Metaheuristics International Conference (MIC-97), Sophia-Antipolis, France, 1997.
7. Costa D. and A. Hertz, Ants Can Colour Graphs. Journal of the Operational Research Society, 48, 295-305, 1997.
8. Di Caro G. and Dorigo M., AntNet: A Mobile Agents Approach to Adaptive Routing. Tech. Rep. IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.
9. Dorigo M., V. Maniezzo and A. Colorni, The Ant System: An Autocatalytic Optimizing Process. Technical Report No. 91-016, Politecnico di Milano, Italy, 1991.
10. Ferguson, I.A.,. On the Role of BDI Modelling for Integrated Control and Coordinated Behavior in Autonomous Agents. Journal of Applied Artificial Intelligence, 9(4), 1995.
11. Goldberg, D., Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.
12. Grassé P.P., La reconstruction du nid et les coordinations inter-individuelles chez Bellicoitermes natalenis et Cubitermes sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. In Insect Societies, Vol. 6, pp. 41-83, 1959.
13. Holland, J. H., Escaping Brittleness: the Possibilities of General-Purpose Learning Algorithms applied to Parallel Rule-Based Systems. In Machine Learning, an Artificial Intelligence Approach, Volume II, edited by R.S. Michalski, J.G. Carbonell and T.M. Mitchell, Morgan Kaufmann, 1986.
14. Hölldobler B. and Wilson E.O., Journey to the Ants. Bellknap Press/Harvard University Press, 1994.
15. Langton, C.G., Artificial Life, Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Things, Los Alamos, New Mexico, Addison Wiley, 1987.
16. Maniezzo V., A. Colorni and M. Dorigo, The Ant System Applied to the Quadratic Assignment Problem. Tech. Rep. IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.
17. Schramm, C., Bieszczad, A. and Pagurek, B., Application-Oriented Network Modeling with Mobile Agents, Proceedings of the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998.
18. Schoonderwoerd R., O. Holland and J. Bruten, Ant-like Agents for Load Balancing in Telecommunications Networks. Proceedings of Agents '97, Marina del Rey, CA, ACM Press pp. 209-216, 1997.
19. Shoham, Y., Agent-oriented programming. Artificial Intelligence, 60(1):51-92, 1993.
20. Susilo, G., Bieszczad, A. and Pagurek, B., Infrastructure for Advanced Network Management based on Mobile Code, Proceedings of the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Luisiana, February 1998.
21. Taillard E. and L. M. Gambardella, An Ant Approach for Structured Quadratic Assignment Problems. 2nd Metaheuristics International Conference (MIC-97), Sophia-Antipolis, France, 1997.
22. White T., Pagurek B. and Oppacher F., Connection Management using Adaptive Mobile Agents, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), July, 1998.
23. White T., Bieszczad A., Pagurek B., Distributed Fault Location in Networks Using Mobile Agents. Proceedings of the Workshop on Intelligent Agents for Telecommunications Applications (IATA '98), July, 1998.
24. White T., Pagurek B and Oppacher F., ASGA: Improving the Ant System by Integration with Genetic Algorithms. Proceedings of the Symposium on Genetic Algorithms (SGA '98), July, 1998.

[2] The authors would like to acknowledge the contribution of one of the reviewers here.