Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Technical
Memo
TM-98-01

Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies

Christian Gerber

January 1998

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies

Christian Gerber

DFKI-TM-98-01

This work has been supported by Siemens AG Germany.

ISSN 0946-0071

Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies

Christian Gerber
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
gerber@dfki.de

January 1998

Abstract

In this paper, we describe an approach to achieve dynamic adaption of an artificial agent society to environment changes. Such changes cause sub-optimalities in the agent society design, leading to overloaded or underloaded agents or agent groups in the society. Our approach provides a mechanism to reveal these sub-optimalities by integrating bottleneck analysis to structural self-adaption in a generic multi-agent framework which allows to incorporate agents of any architecture.

1 Introduction

During recent years, due to advances in the theory of multi-agent systems, but also due to the explosion of hardware power, more and more complex multi-agent systems can be realized. This movement induces the need to introduce structures into such large systems which do not violate the paradigm of agent autonomy since this paradigm has been proven, tried and tested.
Introducing a structure to a group of agents that pursue a common goal, leads to the definition of quantities such as the size of the group, distribution of specialized agents, command structures, types of communication protocols, etc. (see [Ger97] for an overview.) However, for the sake of agent autonomy, the agents' reasoning processes should be left untouched.
As multi-agent systems are usably intended to operate for a rather long period of time (sometimes even ad infinitum), a system which was originally adjusted to work at a high performance level may lose its performance as the environment

changes. Therefore, an *automated self-adaption mechanism* which reorganizes the system is of great value. An approach uniting macro-level (i.e., society) aspects and micro-level (i.e., individual) issues can be found in [GJ97]. To achieve higher performance, heuristics can be integrated to such a mechanism. In this paper, we demonstrate how to incorporate *bottleneck analysis* as a heuristic.
Traditional bottleneck analysis approaches try to derive a rather abstract mathematical model of a complex technical system and the work flow within that system. Usually, a directed graph is used to represent such a system: atomic system components are expressed by nodes; work flow is represented by arcs. Operations Research methods can then be applied on that model in order to detect local overloads.
Such an approach can easily be used to analyze a multi-agent society by interpreting agents as nodes, agent cooperation for achieving a common goal as arcs in the graph and agent actions as work load transitions. However, this kind of approach has some drawbacks: Only agent overloads can be detected; underloads leading to waste of agent power remain uncovered. Furthermore, this approach bases on estimation of the system's behavior: faulty estimations may lead to inaccurate system modifications. Therefore, we propose a new approach to incorporate bottleneck analysis to a agent society self-adaption mechanism: for multi-agent systems we do not need to built up a mathematical model of the system, we can collect bottleneck data directly from the agents representing components or component groups.
This paper is structured as follows: In Section 2 we briefly present basics about bottleneck analysis. Section 3 gives an overview over the underlying principles of the multi-agent framework in use, *SIF* (*Social Interaction Framework*, [FGLS98]). In the subsequent section we describe relevant features of a mechanism to be used for self-adaption on the macro level (i.e., the level of agent societies). Section 5 is the main part of this paper: here we integrate the concepts presented in the three previous sections: We show how bottleneck analysis can be incorporated to a self-adaption mechanism based on *SIF*. In Section 6 we instantiate these generic ideas in a concrete application. Finally, Section 7 gives a conclusion and points to future work.

2 Bottleneck Analysis

Bottleneck analysis is becoming an increasingly important means for optimization of very complex technical systems. Bottlenecks, i.e., unbalanced structures occur if the system was configured using false behavior estimations. Both, local over- or underloaded entities in the system reduce overall performance as overloaded units cause delays, while underloaded units could have been used elsewhere.

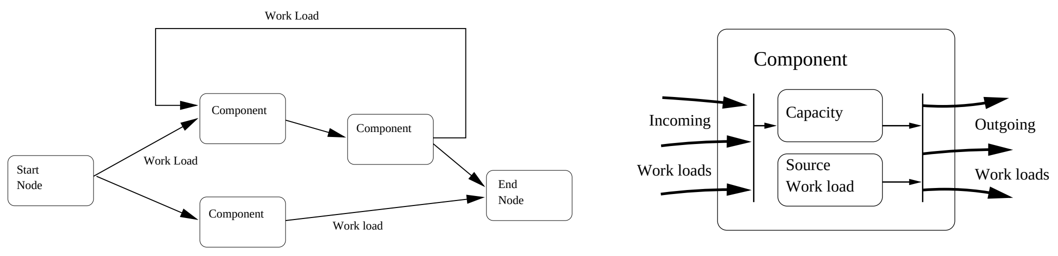


Figure 1: System Model and Component Model

Bottleneck analysis provides an assessment method for units in such technical systems: a unit is characterized on a rather high level of abstraction by only regarding its input/output behavior. The system is represented as a directed graph; its nodes coincide with system components, arcs represent the work flow. In general, two additional nodes are introduced: a start and end node, representing the beginning and the end of the work flow. Figure 1a shows a simple example.
Arcs are labeled with *work load* values. Such a value represents the pass-through of load per time unit. Each component is labeled with a *capacity* value, a *source work load* and a *transition function* (see Figure 1b). The capacity is a measure of pass-through, limiting the processing of incoming work loads. It has to be regarded as an upper threshold of work load to be processed without leading to delays. A unit's source work load constitutes work load that originates from that particular unit, i.e., that did not emerge from other factors. The effect of work load (incoming as well as new source load) on a component results in a reduction of capacity and of new work load leaving the unit, determined by the transition function.
The transition function has to compute

- the contribution of the various incoming loads to capacity consumption,
- the contribution of the new produced work load to capacity consumption,
- the total capacity consumption and the throttling of incoming work load (i.e., the overload), should the situation arise,
- and the outgoing work load depending on the above quantities.

In a traditional approach, a bottleneck is regarded as a location of overload within a system structure. *Real bottlenecks* are usually distinguished from *hidden ones*: Real bottlenecks occur if the capacity of a unit is smaller than the work load it has to cope with. A hidden bottleneck is an imbalance that is currently not affecting the system's overall performance, but which will do so if other, real bottlenecks are removed. Goal of bottleneck analysis is the detection and removal of bottlenecks.

By characterizing the work flow in a fashion described above, a mathematical model of the system is put up. Network flow algorithms developed in Operations Research can then be applied to that model in order to detect bottlenecks.
Such an approach can be easily adapted to detect sub-optimalities within a society of artificial agents. However, it has some severe drawbacks:

- Only overloaded units can be found. Underloaded components cannot be detected; their unused work power is lost.
- Work load and capacities have to be estimated for the mathematical model of the system. If these estimations are incorrect, no mechanism can find real bottlenecks.
- In traditional bottleneck analysis, work load and capacity are not considered variable over time. In such cases of variations, average values are taken or the procedure has to be rerun for each variation.
- Similarly, the analysis has to be repeated after each system modification due to detection of real bottlenecks in order to find hidden ones.

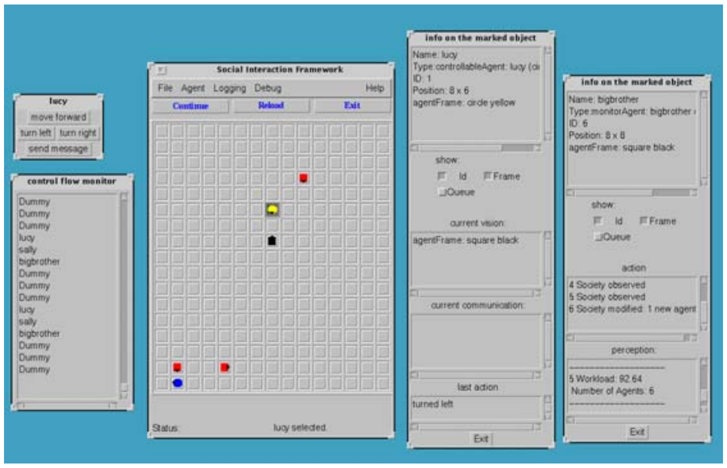
For these reasons, we propose a different approach to bottleneck analysis in societies of artificial agents: We do not estimate work load and capacities for building up a mathematical model, but make direct use of the multi-agent system architecture which provides us with information about agents' work load. The bottleneck detection and system adaption mechanism is directly integrated into the system architecture allowing dynamic adaption to situation changes. We will present that mechanism in detail in Section 5, after having proposed the basic mechanism of *SIF* and a self-adaption mechanism for agent societies in the next two sections.

3 *SIF*—an Effector-Medium-Sensor Framework

The MAS simulation environment *SIF* [FGLS98]), (Figure 2) has been developed in JAVA™ for the study of social interaction between artificial agents of arbitrary architectures.[1] *SIF*'s underlying basic mechanism is a so-called *Effector-Medium-Sensor* (EMS) architecture based on Russell and Norwig's definition of an agent in [RN95].

> An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

[1]If perception, communication and action of some agent architecture is not directly compatible to *SIF*, a simple wrapper object has to be written as an interface.



```
Dummy
Dummy
Dummy
lucy
sally
bigbrother
Dummy
Dummy
Dummy
lucy
sally
bigbrother
Dummy
Dummy
Dummy
```

Status: lucy selected

```
Name: lucy
Type:controllableAgent: lucy (ci
ID: 1
Position: 8 x 6
agentFrame: circle yellow
```

```
agentFrame: square black
```

```
turned left
```

```
Name: bigbrother
Type:monitorAgent: bigbrother
ID: 6
Position: 8 x 8
agentFrame: square black
```

```
4 Society observed
5 Society observed
6 Society modified: 1 new agent
```

```
--------------------------
5 Workload: 92.64
 Number of Agents: 6
--------------------------
```

Figure 2: The Social Interaction Framework (*SIF*)

The central component of the architecture is the *medium*. *Effectors* emit *actions* to the medium, which in turn sends the effect of that actions as *percepts* to *sensors*. Figure 3a shows the correlations.
Agents are entities equipped with effectors and sensors in order to emit actions to and receive percepts from the *world server*, an instantiation of the medium, representing the environment. Hence, agents perceive other agents' actions not directly; the world server "filters" the effects of actions by computing the local perspectives of perceiving agents. The world server sends these perspectives as percepts to the sensors of perceiving agents. *Perception ranges* can be installed: a percept is only received by a sensor if the emitter happens to be in the perception range of that sensor. Figure 3b visualizes the information flow. Similarly, agents communicate only via the world server, not directly with each other. They receive communication acts as percepts which might be blurred.
In order to file all incoming percepts, agents and media must have an event queue at their disposal: incoming percepts are stored there and dispatched sequentially. However, for performance reasons, communication percepts are treated with higher priority. Therefore, an additional communication queue is introduced. Due to the generic layout, any type of agent can be connected to the world server, ranging from to simple JAVA™ objects to sophisticated INTERRAP agents [Mül96].

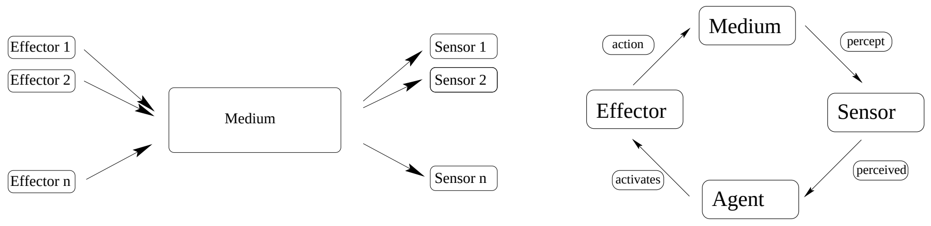


Figure 3: Sensor-Medium-Effector Model and Information Flow

4 A Self-Adaption Scheme for Agent Societies

In this section, we briefly point out the architecture of our self-adaption mechanism for a group of agents. A detailed description can be found in [GJ97]. The key idea is to control a society of still autonomous agents on a meta-level through structurization and organization. None of the agents' object-level tasks will be addressed by the adaption mechanism.
In [RW91], Russell and Wefald argue that an approximation of optimal behavior cannot be reached in a domain-independent manner without solving the higher-order problem of additionally approximating the optimal agent program. This justifies the use of *meta-reasoning* in the single-agent case.
Unfortunately, in the multi-agent case, complexity of the object-level (i.e., combining all perception histories and producing reasonable, simultaneous decisions) is intractable for a one-stage meta-level reasoning. We therefore propose a multi-staged resource adaption/meta-reasoning mechanism: The agent society, agent groups and subgroups all are represented explicitly by *monitor agents* whose tasks is to optimize the structure of the represented society, groups or subgroups. This multi-staged meta-level reasoning enables us to overcome complexity explosion mentioned above.
We regard the task to adapt the structure of a group or society of artificial agents to the current environment as an optimization problem by characterizing a *search space* and an *objective function* to be optimized. The objective function has to denote the current system's performance while a multi-dimensional search space must describe the system's set of possible configurations. With regard to the application, the *objective function* has to be defined depending on several factors the application designer has to combine, such as operating time, quality of the result, etc. Each modifiable property of the system reflects one dimension of the *search space*. The search space dimensions can be derived from scalable principles of a multi-agent application: *Structural principles* are for instance

- *number of agents* in the group,
- *number of specialists* for certain tasks,

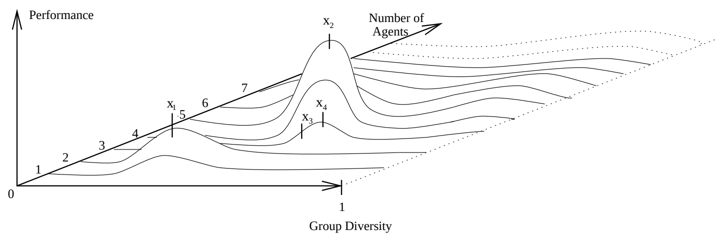


Figure 4: A Simple Example of a Performance Relation

- *organizational form* of the group,
- *migration*, i.e., distribution of agents over the net, etc.

Communication principles (for details see [FMP$^+$97]) can be expressed through

- *introduction of communication channels* between subunits or even between agents belonging to a common subunit,
- *usage of different communication protocols* leading to different communication complexities,
- *assignment of communication roles to agents* leading to rankings inside subunits, etc.

Agent architecture principles are explicit resource distribution among the various agent modules (see [GJ97] for a unified approach based on resource distribution management). In this work we focus on the first two principles: communication principles and, mainly, on agent group structure principles, as they are more independent from the chosen agent architecture.
Figure 4 shows a simple, two-dimensional example: here performance of a group depends only on its diversity and the number of agent members. *Number of agents* is represented by a discrete dimension, whose domain ranges from 0 to possibly infinity. *Group Diversity* is modeled as a continuous dimension, its domain ranging from 0 (i.e., all agents have identical characteristics) to 1 (i.e., their traits vary greatly).
As global optimality can hardly be achieved in a reasonable amount of time, but may be lost very easily, we incorporate a mechanism for achieving and maintaining relative high performance during the complete run of an application. This mechanism bases on the local search method for finding *local* optima.

Method

1. The application designer produces an initial model of the application by instantiating scalable parameters: a start point in the search space is defined.
2. The system is run over a pre-defined time period Δt; the average system performance is measured, based on an evaluation of the objective function described above.
3. The direction to move in the search space and the size of the step are determined (see details in Section 5.2). That step is performed by the monitor agent that reconfigures its group.
4. If system performance has improved after the time period Δt, a further step in that direction is taken and step 4 is repeated.
5. Otherwise the step is undone and step 3 is repeated.

In the above example a starting point may be x_3. A maximal performance gain will be achieved by increasing the group diversity: x_4 may be achieved. By increasing the number of agents to five, the optimal configuration x_2 will be found.
Of course, the nature of the environment may change (e.g., a new type of specialist is introduced) leading to adjustments of the formal search space and the current position (here a new dimension *Number of agents of the new type* is added; the value of the current position on that dimension is 0.) In order to overcome some local minima, random jumps may additionally be performed from time to time - similar to the *Simulated Annealing* technique [KGV83]; if performance increase is not measured, that jump is undone again.
Obviously, in this approach the autonomy of member agents is partially traded for more central control leading to a general behavior which is more oriented towards the overall goal of the society. Depending on the application domain, the competences of the monitor agent may be set differently; the resulting society can vary from a loosely moderated group up to a hierarchical structure. However, in any case, members of such a controlled society still are agents that make independent local decisions based on their architecture (which can be reactive, deliberative, BDI-oriented, etc.) Therefore, this approach does not violate the core of the paradigm of agent autonomy.

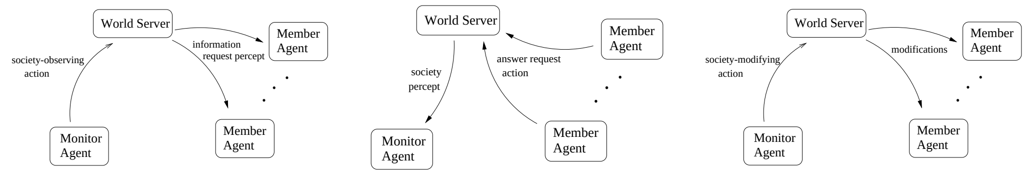


Figure 5: Control and Information Flow

5 Integration of Bottleneck Analysis into a Self-Adaption Mechanism

For the realization of the above algorithm two issues must be addressed: *How to control the group structure* and *How to perform a local step* (step 3 in the algorithm). We will now address these issues.

5.1 Control in Structured Agent Groups

In contrast to other group formation and adaption approaches (e.g.,[DP95]), once created, agent groups and societies are represented *explicitly* by a monitor agent. Such a representative of an agent group or society can be society members or additional agents whose sole functionality is to monitor. In this section, we describe in detail how a monitor agent controls its group or society by detecting bottlenecks.
Just like any other agent in the *SIF* framework, a monitor agent interacts with its environment (including other agents) via sensors and effectors: Its sensors receive percepts from the world server; its effectors emit actions whose effects on the environment are computed by the world server. The monitor agent is equipped with the group structure adaption mechanism described in the previous section.
However, the monitor agent needs special sensors and effectors: From time to time *Deltat*, it has the goal to monitor the structure of the agent group it is responsible for. It therefore emits a *society-observing* action, triggering the world server to determine the current society status (i.e., number of agents, their structure etc.) and the average system performance since the last system reconfiguration. Furthermore, all members of the group receive via their communication sensors *information-request percepts* requesting information on their goal queues (if existing), communication queues and percept queues. (Figure 5a). By defining the perception range of that communication sensors appropriately, it can be realized generically that only agents receive that request which are members of the group the monitor agent represents.

The agents in question emit the requested pieces of information through performing an *answer-request action.* Once all answers have been collected (or timed out), the world server sends that data as a *society percept* to the monitor agent (Figure 5b). The monitor agent in turn, uses the collected data to compute the next local step in the search space (i.e., the modifications in the society configuration). The monitor agent emits an *society-modifying action* (containing instructions how to modify the group) to the world server which commits the modifications, or computes impacts on the society respectively, as it cannot be guaranteed that all actions are committed successfully. (Figure 5c). Additionally, over time the group constellation may change due to external reasons. Therefore, in order to get correct information on the society situation, the monitor agent needs to perform a new *society-observing action* the next time that group has to be controlled. This procedure is repeated until the application is terminated.
At first glance, this approach may occur to be rather centralistic, since the monitor agent is the only unit that makes structural decisions. However, it has to be noted that *only* structural decisions are found by the monitor agents; object-level decisions are still made by the member agents. Furthermore, at a more detailed look, this method can be seen as a variation of the *Contract Net Protocol* (CNP), a well-known decision finding procedure in *Distributed Problem Solving* which could also be used in this setting. In both cases, a central unit (here the monitor agent) has to solve a problem and announces it to a set of agents (here members of the group). However, instead of collecting the members' evaluations of the problem based on their local profiles (i.e., their bids for organizing the society according to their local preferences), in our case the monitor agent collects these profiles in order to get a more global profile enabling it to find a better solution.

5.2 Performing a Local Step

In the model above the monitor agent has to apply a local step in the search space described in Section 4. Obviously, pure uninformed search will lead in step 3 to a random choice of direction in case a new step is *not* taken in the same direction as the previous one. However, in such cases, heuristics can be incorporated to determine the next step. Figure 6 shows the incorporation of heuristics: they can be gained from sciences such as social psychology (see for instance [Nas88]) and management theory (see e.g., [Fre95] about the optimal size and structure of a group to perform a certain task), but also from the design of computer players in real time computer strategy games. Furthermore, a learning component (based on reinforcement learning of the decision making history, see [Mit97] for an overview) may be employed in order to draw conclusions about the usefulness of previous modifications. In this paper, however, we focus on heuristics that can be gained from data collected from world server and member agents.

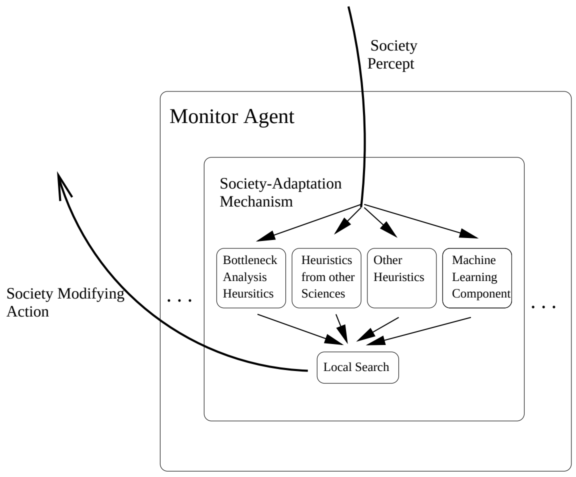

Society Percept
Monitor Agent
Society-Adaptation Mechanism
Bottleneck Analysis Heursitics
Heuristics from other Sciences
Other Heuristics
Machine Learning Component
Society Modifying Action
Local Search
. . .
. . .

Figure 6: Control Mechanism Components

It is the nature of a heuristic to give a rough guideline or indication to find faster a better solution. Depending on the application, such a guideline needs not to be exact or correct for all cases. Additionally, an indication may be interpreted in different ways leading to different suggestions what step to perform next. The following observations are modeled to heuristics in that sense: If there is no other indication to decide between several choices, relying on one of these heuristics prevents from making a random choice. Furthermore, note that these heuristics are kept generic on purpose, since our approach is designed to be integrated into a framework of heterogenous agents.

- The size of member agent goal queues (if existing) indicates their work load: A very large goal queue may be a sign that more agents are needed that are specialized on achieving that type of goals. On the other hand, if the queue is only filled sparsely, the work of this agent may be done more efficiently by another agent of the same type (provided there are some), considering the fact that an idle agent still consumes time and space resources. This may therefore be an indication to delete that particular agent.

- The size of agent communication queues can be used to validate the usefulness of the chosen type of communication protocol: a too large queue reveals that currently applied communication protocols are too complex as they are leading to an overload. On the other hand, an almost empty communication queue is a sign for an underload: a more complex communication protocol that might lead to better results could be used without reducing overall performance.

- Furthermore, an analysis of agent communication queues in terms of the distance of communication partners in a network and of communication complexity can be used to decide when to employ agent migration: If complex protocols are pursued over a long distance, it may become more efficient to send the agent over the net and let it perform communication locally.
- Depending on the communication language used between agents, the type of communication acts found in communication queues gives also hints about the optimal structure: hierarchies or other social structures of a group may speed up cooperation as cooperation roles are already pre-defined. If an agent emits many command communication acts such as *request* and often refuses requests from other agents, a more hierarchical structure may be introduced, giving this agent a high position.

All these observations can be used as heuristics to enable a monitor agent to approximate optimal structure and communication patterns in an agent group.

6 Application: The Colonization Scenario

In this section, we show how the concepts presented above can be applied in a concrete multi-agent application.

6.1 General Properties

In the *colonization scenario* a collection of agents are exposed to an unknown environment in order to build up and to live in colonies (which could be placed on the North American continent in a colonization simulation of the recent centuries, as well as on unknown planets in a simulation of future outer space missions.)
The environment offers raw material sources of various kinds. Colonist agents have to process them in order to produce groceries to be consumed and to build up settlements, means of transportation, etc. The environment may also provide threats to colonists, e.g., natural disasters may occur or predators may exist. Environment and predators are also modeled as agents which allows us to interpret the scenario as a *zero-sum game:* agents, including the environment representative, process goods which are transferred from agent to agent as a result of actions. No goods are introduced or taken away from the system. For instance, food production is a good transfer from the environment agent to farming agents. By setting an initial raw material supply of the environment rather low, a shortage of goods can be simulated; on the other hand, to simulate regenerating sources of raw material, the initial value can be set very high.

Colonist and predator individuals have unique skills to perform certain jobs; they have some sort of "life energy", ranging from 0 (the agent is dead) up to 100 (the agent is perfectly healthy). Any activity an agent performs results in energy loss. The agents' main goal is to keep their life energy as high as possible. Increasing energy is achieved by consuming food. Specialization and cooperation between agents is reasonable because the process of producing food can hardly be achieved solitary, not only for colonist agents, but also for predators. Besides their "ordinary" actions such as communication actions, *move* actions, various subtypes of *collect-raw-material*, *process-raw-material* or *consume-food* actions, agents have *found-group*, *join-group*, or *leave-group* actions at their disposals. So if, according to their architecture, agents are enabled to reason over their next actions, processes of group formation and development can be realized canonically.

6.2 The Self-Adaption Mechanism

Agent groups are *explicitly* represented by a monitor agent. Agents groups again can cluster to larger units which again are represented explicitly. Persistent goal of agent groups or societies is to grow as strong as possible, i.e., to consist of as many members as possible that can achieve their main goals as well as possible. Groups and societies are represented explicitly by (monitor) agents. The main goal of a group (i.e., the objective of the corresponding monitor agent) is to gain as much control over food as possible in order to enable the survival of group members. Hence, the monitor agent representing a group must reason on the number of group members, type of agents, patterns of command in the group, etc., (all scalable quantities enumerated in Section 4) to optimize the group composition. The system is initialized with a society configuration the user has inserted prior to the run of the simulation. All member agents perform their tasks, i.e., they collect and process raw material, consume food etc. After a time interval Δt, the monitor agent performs an *information-request action* in order to receive information about the food stock and the number and status of the member agents.

The monitor agent now computes a local step in the configuration space taking into consideration the difference in the food stock, the previous move and the heuristics about the agent event queues just as described in the previous sections. For instance, if the food stock has increased and the previous society modification was the hiring of more specialist of a certain type, say farmers, then the group representative now also tries to hire more farmers. If, on the other hand, the food stock had decreased, the monitor agent tries to undo the previous action (here, to fire the additional farmers). If furthermore for instance, the agents' communication queues are packed, the monitor agent changes the communication protocol between members to a simple one (e.g., from an complex auction

mechanism to Contract Net Protocol). These modifications are performed by emitting a *society-modifying action* to the world server.
As agents can decide, whether or not to join or leave a certain groups, the outcome of a group modification process is hardly predictable. This is in particular true for cases in which a group representative chooses to enlarge its group: there may be no agent of the wanted type available or they may refuse to join the group. Therefore, the representative cannot be sure that all of his society modifying actions have been performed successfully. This justifies the presented approach in Section 5, always to perform a *society-observing action* before a new *society-modifying action.*
So, not only group formation evolves naturally from goal optimization, but also agent characteristics such as *selfishness, social behavior* or *solitary behavior*, concepts well studied but mostly represented in an explicit manner. Furthermore, agent groups and subgroups can be characterized by propagating member characteristics.

7 Conclusion and Future Work

In this paper, we have presented an approach to incorporate bottleneck analysis to a mechanism that enables self-adaption of a society of artificial agents without giving up the paradigm of agent autonomy. In detail, our approach makes use of analyzing event queues that agents and world server process during the run of an application. We have demonstrated how society structure control is integrated naturally into the layout of the generic agent framework *SIF* enabling the use of agents of any architecture. In particular, we have introduced *monitor agents* in order to represent agent groups and society *explicitly* and we have shown how these agents can perceive and modify the society. For the sake of clarification, we have presented a simulation scenario of behavior of human societies, the *colonization scenario* where these concepts can be used canonically.
Future work will concentrate on the introduction of a machine learning component to this approach, mentioned in Section 5, and of further validating this approach by introducing several societies competing against each other. These societies have different policies to operate, some might simulate communist policies, others might simulate market oriented policies, some may have the adaption mechanism at their disposal, others may not. The goal then is compare society policies on an abstract level, similar to the comparison of the iterated prisoners' dilemma problem in [Axe84].

Acknowledgments

We would like to thank the members of the MAS group at DFKI and to Donald Steiner at Siemens AG for fruitful discussions. In particular, the self-adaption mechanism has been derived under continuous discussions with Christoph G. Jung. *SIF* was developed in cooperation with Petra Funk, Jürgen Lind, and Michael Schillo.

References

[Axe84] R. Axelrod. *The Evolution of Cooperation.* Basic Books, 1984.

[DP95] J. Doran and M. Palmer. *Artificial Societies*, chapter The EOS project: integrating two models of Palaeolihic social change, pages 103 –125. VCL Press, 1995.

[FGLS98] P. Funk, C. Gerber, J. Lind, and M. Schillo. Social Interaction Framework—A Generic Testbed for Social Agents. Research Report, German Research Center for Artificial Intelligence, 1998. To Appear.

[FMP$^+$97] K. Fischer, J.P. Müller, M. Pischel, C. Gerber, and B. Chaib-draa. A Simulation Approach based on Negotiation and Cooperation between Agents: A Case Study. *Submitted to: ACM Transaction on Modeling and Computer Simulation: Special Issue on Simulation of Scalable Systems*, 1997.

[Fre95] E. Frese. *Grundlagen der Organisation.* Gabler Verlag, 6th edition, 1995.

[Ger97] C. Gerber. An Artificial Agent Society is more than a Collection of "Social" Agents. In *Socially Intelligent Agents - Papers from the 1997 AAAI Fall Symposium.* Technical Report FS-97-02, AAAI, 1997.

[GJ97] C. Gerber and C. Jung. Towards the Bounded Optimal Agent Society. In *Distributed Cognitive Systems: Proceedings of the VKS'97 Workshop.* Dokument D-97-08, DFKI GmbH, 1997.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimalization by simulated annealing. *Science*, 220:671, 1983.

[Mit97] T. Mitchel. *Machine Learning.* Mc Graw Hill, 1997.

[Mül96] J. P. Müller. *An Architecture for Dynamically Interacting Agents.* PhD thesis, Universität des Saarlandes, Saarbrücken, 1996.

[Nas88] D. Nasser. How to run a focus group. *Public Relations Journal*, 44:33–34, 1988.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[RW91] S J. Russell and E. Wefald. *Do the Right Thing.* MIT Press, 1991.

Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies

Christian Gerber

TM-98-0
Technical Mem