Adaption in Distributed Systems - An Evolutionary Approach

Stephan Otto, Stefan Kirn
University of Hohenheim
Schwerzstrasse 35
Stuttgart, Germany
{ottosn,kirn}@uni-hohenheim.de

ABSTRACT

There is a trend towards networked and distributed systems, complicating the design process of self-adaptive software. Logistics networks can be seen as a distributed system that have to adapt to requirements of companies and customers in a flexible and fast manner. When constructing and planning logistic networks different aspects of complexity have to be considered: the number of stores, intermediate stores and transport entities that are required at every stage in a supply chain as well as the sufficient size of every store or transport entity. This paper presents an approach that simulates adaptive logistic networks using a multi-agent system (MAS) based on Evolutionary Computation (EC). Our approach uses fully decentralized operators for reproduction like mutation, recombination and selection, regulated by market mechanisms. The novelty of this approach lies in the decentralized bottom-up adaption method for decentralized systems and we use a logistic scenario as an example. Our proposed method is based on a formal model explaining how adaption occurs in the number and strategies of agents and thus of logistic networks. The implementation and experimental results are given to illustrate the expected outcomes.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Distributed Artificial Intelligence, Multiagent systems*

General Terms

Algorithms

Keywords

Multi-agent systems, Adaption / Self-Adaption, Evolutionary Local Search, Artificial life, Simulated Adaptive Behaviour

1. INTRODUCTION

With the rise of global enterprise networks, single companies need to react fast and flexible in order to remain in dynamic markets. In this context, simply adapting intra company processes is insufficient to preserve competitiveness. Instead focus broadening towards cross-company structures takes place [13]. These structures can be seen as networks consisting of nodes connected by edges with flow of objects (e.g. material, information, money, goods) between the nodes. In general, the flow in these networks can be seen as intertwining of movement along the edges and (temporarily) storing in the nodes. Regardless, of the objects flowing through such networks, we can see them as logistic networks [17]. In this paper we are focusing on goods and information as objects which pass through logistic networks. At this point different concurrent objectives occur:

- *Trade-off between resource allocation and costs:* Fast and failsafe flow of objects through the network requires an oversized (and possibly unlimited) network to cope with all tasks. This results in huge costs due to high resource usage (e.g. number of stores, transport entities). On the contrary a reduced number of nodes in the network may cause delay and/or failures.
- *Reduction of costs by grouping several streams of goods (consolidation) causes complexity.* Although it is more effective to bundle different streams in logistic networks, this induces mutual dependencies thus increasing complexity (e.g. in coordination).
- *Capacity restrictions at every node in a logistic network:* A balance between the diversity of goods for storage and the capacity of every node has to be found. Finally an increased variety of goods leads to more complex supply processes requiring extended storage facilities and operations.

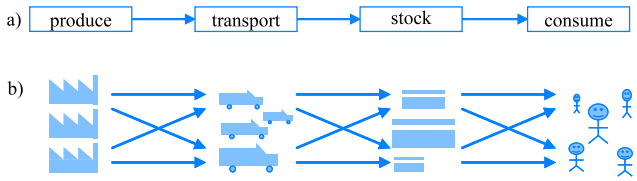

Figure 1: a) Distribution process from producer to consumer, b) Structure of a logistic network according to the process in (a)

Typically the flow inside logistic networks is organized in processes that describes the sequence of activities in the network. So, according to the process in figure 1a) we have four steps that determine the flow of goods through the network starting with production of goods followed by transportation, then to stocking and finally sale to customers. In figure 1b) a possible logistic network consisting of producers, trucks for transportation and stores for selling according

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

to the process shown in (a) is given. An actor may have different capabilities and parameters determining his behaviour or strategy (store size, transport capacity, etc.) and thus have an influence on the complete network. The goal in logistics is to streamline the network structure while improving service to the customer [2]. The adaption of transport networks does not only interest supply chain management, but also introduces a lot of other fields related to computer science. In the former case an ineffective network leads to higher transportation and inventory costs and in the latter case the optimization of such networks is NP-complete [10]. Typically in constructing logistic networks the overall problem is split into several (also NP-complete) parts and solved sequentially [10]. Common sub problems are in the estimation of the number and location of stores, the correlation of customers to stores, the capacity and product range of stores, and the required transport capacity to deliver objects through the network. All of these issues are related to any stage in the logistic process and solving correlated sub problems independently may prevent the finding of a global optimum [13].

An inherent characteristic of distributed problems like logistic network adaption is the fact that nobody is aware of the complete overview of the whole system. This is based on two reasons. The first reason is that nodes are limited by how much they can communicate and process. Any central entity would need to know (all) information which leads to communication problems due to bandwidth limitations or high computational cost [4]. The second reason is simply information hiding, which means, that not every information can be given to a central control due to intellectual property or security reasons [8, 3, 11].

In this paper we present a multi-agent system that attempts to optimize this distributed problem. As an example of such a distribution problem we use a logistic network based on processes illustrated in Figure 1a) in a decentralized bottom-up approach. For this we use evolutionary computation based on an economic model. We assume no central instance as a manager or coordinator; instead we use a bottom-up approach for adaption in distributed systems by simulation. Decentralization is an inherent characteristic of logistic networks [8, 13] and thus we are especially interested in exploring the extend to which a decentralized adaption can lead to an improved result (from a system designers perspective) and how an EC-enabled and marked-based agent system implementation can realize this adaption appropriately. This paper presents an approach on how distributed systems can be controlled without any central entity.

The remainder of this paper is organized as follows. In section 2 we review previous work related to logistic network structure optimization and evolutionary computation in multi-agent systems. In section 3 we present our model of a market-based agent system. Section 4 presents our decentralized optimization method by combining evolutionary computation and multi-agent systems. Using this model we specify two emergent effects that occur within such a distributed system formed by agents: adaption of the number of agents and adaption of agent's strategies to a given scenario. We show how the number of agents is regulated by market mechanisms and how resource efficient strategies dominate. In section 5 we shortly describe our current implementation and present first experimental results. Finally the conclusion and future plans are mentioned in section 6.

2. RELATED WORK

For specific sub problems like depot location search, known as *p-median-problems*, methods exist that are basically heuristic [10]. One of them described as 'fast interchange heuristic' by Whitaker in [20] solves the *p-median-problem*, but capacity is not considered, which can be seen as unrealistic. A more extensive survey is presented in [7, 13] whereby only top-down algorithms are considered. First results of an independent market study indicate that most of the available supply chain management software tools support centrally organized networks and lack a collaborative planning and execution support [9] and thus also use a top-down approach with central coordination, which is not applicable to our problem.

So far, very little work has been conducted on evolutionary computation in multi-agent systems concerning logistics. An evolutionary approach using economic agents for studying strategies in electronic markets is presented by [1]. Here trading agents are investigated by studying heterogeneous strategies in large electronic markets. A traditional 'central' evolutionary algorithm is used and the process steps hard wired in the agent system makes it inappropriate for the present problem. A decentralized EC-enabled MAS framework is presented in [19] where local selection occurs and has been used in [5, 18, 16]. In [5, 18] Smith and Eymann investigate negotiation strategies in a supply chain for the production of cabints but concentrate merely on self-organizing coordination effects. Explicit control is not provided. Further, they consider only coordination effects between agents but not between agents and the environment. Previous EC-enabled MAS approaches use agents that are specific in their function and hence the re-usability of agents in different logistic scenarios or processes is not supported. Further, the use of centralized infrastructure like yellow pages may induce a bottleneck as the agent population, the communication load and the information in the system grows. [16] uses agents that compete for shared resources directly from the environment. Resources are not traded between the agents and subsequently replenished to the environment. In our sample logistic scenario (figure 1) this approach is inadequate, because agents not only interact with the environment but also perform intermediate interactions.

To the best of our knowledge, there is no multi-agent system approach that adapts logistic networks by using evolutionary computation that supports processes and a decentralized market based control directed from the environment. In this paper we use a scalable architecture of a multi-agent system without any central control that can run up to several thousands of agents.

3. MODEL

3.1 Multi-Agent System

In order to model and study distributed systems, we use a multi-agent system:

DEFINITION 1 (MULTI-AGENT SYSTEM (MAS)). *A multi-agent system $MAS = (A, U)$ consists of a finite set of agents $A = \{a_1, ..., a_i\}$ embedded in an environment U.*

Since a MAS consists of agents, we give the following definition for an agent:

DEFINITION 2 (AGENT). *An agent $a = (I, O, f, S, c)$ consists of a finite set of sensory inputs $I = \{i_1, ..., i_n\}$, a finite set of effector outputs $O = \{o_1, ..., o_m\}$, a function $f : I \rightarrow O$ which maps sensory inputs to effector outputs, a finite set of strategy parameters S determining f and the agents current funds $c \in R$.*

An agent a is an autonomous sensor effector entity whose function f_a can adapt to both, environment U and other agents $\{A \setminus a\}$. The strategy S_a represents a set of parameters that are assigned to an agent a. We consider, the agent's function f_a to be determined by S_a and thus, the strategy S_a of agent a determines the behavior

of agent a itself. Access to particular parameters is given by the following notation: $S_a(parameter1)$ denotes $parameter1$.

3.2 Distributed Optimization Problem

This section describes the formalization of the distributed optimization problem. Given a set of variables $\{s_1, s_2, \ldots, s_p\}$, each assigned to an agent a. $S_A = (s_1, s_2, \ldots, s_p) \in S_A^*$ is called the strategy vector of the MAS and S_A^* is called the search space. The distributed optimization problem (DOP) is given by:

DEFINITION 3 (DISTRIBUTED OPTIMIZATION PROBLEM). *A distributed optimization problem (DOP) is given by:*

$$\textit{minimize} \quad F(S_A), \textit{subject} \quad C(MAS)$$

where F is not known and therefore can not be calculated by any single problem solver.

The set $C(MAS)$ contains constraints of the DOP distributed over the set of agents A and the environment U. The combination of the local strategy vectors of all agents $S_A = \sum_A s_a \quad a \in A$ form together the strategy vector of the problem. The DOP becomes even harder, if constraints vary over time t. The dimensions of S_A and therefore of S_A^* are not fixed, rather they may vary over time, as agents enter or leave the system.

3.3 Integrating an Economic Perspective

Referring to the presented model of distributed optimization problem, we can simulate a network of distributed and connected actors with a multi-agent system. We consider an agent a seeking to effectively exploit a limited set of resources according to its Strategy S_a. Thus a network is composed of autonomous agents each acting to maximize its funds c_a based on local assumptions and limited information. In our example, the only good with intrinsic value to an agent are funds. The value of other stocks is based on the possibility to convert them to money at some future time by selling or transporting them. Furthermore the actions of one agent distort the utilities experienced by other agents. Thus the logistic network can also be seen as a complex dynamic network of trading agents.

Referring to the process in figure 1a) consumers order goods from stock, which in turn order from transport service and finally the transport service orders goods from the producer itself. Vice versa the flow of goods follow the process from producer to transport to store and finally to customers who ordered the goods. A transfer of goods between two agents always implies the flow of money in the opposite direction. The exchange of money is a payment $\mathcal{P}$ for the sender and a receipt $\mathcal{R}$ for the receiver. All parameters of such a transaction are characterized by the strategies of both involved agents e.g. the number of goods traded.

In our model we assume a discrete timeline, where all actions take place at consecutive steps. Further, agents have to pay a tax $\mathcal{T}_a(t)$ to the environment U at every time t for their actions and according to their strategy parameters S. So, for example the tax may be calculated e.g. by memory usage, CPU usage, the number of messages sent to other agents, the store size of a store agent or the transport capacity of a transport agent. We denote tax as a function

$$\mathcal{T}_a(t) = \mathcal{T}_{S_a}(t) + \mathcal{T}_{fa}(t)$$

that calculates the tax of an agent dependent on the agents strategy parameters and actions performed by its function at time t. The tax is set by the environment U (which is usually controlled by the user). Note, that actions are taxed as and when they occur and strategy parameters will be taxed periodically. As an example lets assume the tax as $\mathcal{T}_a(t) = S_a(storesize) * 0.5$. So for example a small logistic agent $a1$ with $S_{a1}(storesize) = 10$ has to pay 5 units of fund to U at time t which is less tax than a big logistic agent $a2$ with store size $S_{a2}(storesize) = 100$: $\mathcal{T}_{a1}(t) = 5 < \mathcal{T}_{a2}(t) = 50$. Given the tax, and the flow of money we can calculate the profit π_a an agent a receives at time t by

$$\pi_a(t) = \mathcal{R}_a(t) - \mathcal{P}_a(t) - \mathcal{T}_a(t) \tag{1}$$

where $\mathcal{P}_a(t)$ denotes the payment, a has to pay to other agents in order to execute its strategy S_a, $\mathcal{T}_a(t)$ denotes tax turned over to the environment and finally a may have receipts $\mathcal{R}_a(t)$ from previously sent goods. Based on the profit $\pi_a(t)$, an agent accumulates funds c_a over time expressed by:

$$c_a(t+1) = c_a(t) + \pi_a(t) \tag{2}$$

where $c_a(t+1)$ denotes the funds of agent a at time $(t+1)$, $c_a(t)$ denotes a's funds at time t and $\pi_a(t)$ is the profit of a at time t. Money can not be 'created' by the agent, rather it is provided by the environment U which provides a specific amount of funds and tries to order a specified quantity of goods. Therefore U can be seen as a demand to the economic agents in the logistic network and limits the amount of money.

4. EVOLUTIONARY COMPUTATION AS DECENTRALIZED OPTIMIZATION METHOD

The aim of this paper is to show an optimization approach for distributed problems lined out in section 3.2. Before we go on to show our method we will epitomize our assumptions on a system, that applies this method:

- All the problem solving nodes have the ability to communicate
- There is no central manager
- There is something like money (or energy) that will be provided by the environment

The previous section has shown a market-based multi-agent system that consists of economic agents who have to solve a global task collaboratively, simultaneously fulfilling previous prerequisites. In this section we re-use basic evolutionary algorithm theories of Holland [12, 6] concurrently with a decentralized economic agent perspective given in [18] as well as our economic market perspective as we believe that a decentralized market mechanism can be a profound approach to replace the central fitness calculation in evolutionary algorithms. We want to analyze how market demand by U affects emergent adaption and allows to control agents in a logistic network consisting of single actors with a local point of view.

There are a variety of evolutionary computational models that have been proposed and studied that we will refer to as evolutionary algorithms [6, 12]. In our agent based approach we turn the typical EA software design on its head by breaking up any central instance and move the genetic representation as well as the operators to the agents respective individuals itself [19]:

Local reproduction by an agent includes all steps and operations necessary to produce new offspring, such as searching for a mate, recombination and mutation. We introduce a constant θ that serves as a threshold and enables agents to reproduce. Whenever the fund c_a of an agent a exceed this threshold ($c_a \geq \theta$) the agent will reproduce. Advantages of local versus global reproduction are discussed in [16] in detail. As we want to focus on emerging global effects

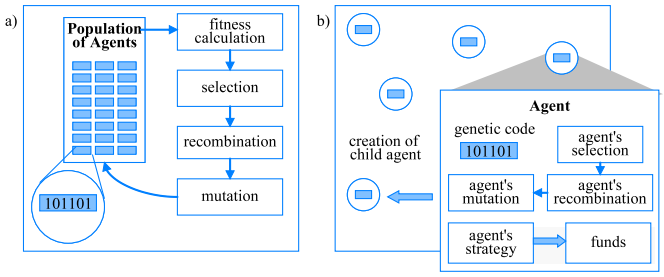

Figure 2: a) Typical centralized structure of a genetic algorithm, b) Distributed operators and genetic code in EC enabled agent system

of local reproduction, based on [15] we have formulated an agents algorithm motivated by modeling ecologies of organisms adapting in natural environments as follows:

```
initialize f according to S and get c                    1
do forever {                                             2
   get sensory inputs I                                  3
   execute f                                             4
   set effector outputs O                                5
   calculate c using equation (2)                        6
   if ( c ≥ θ)                                           7
      reproduce                                          8
      split c with child                                 9
   elseif ( c < 0)                                       10
      die                                                11
}                                                        12
```

Listing 1: Pseudo code executed in every agent continuously

An agent is set up with its strategy S, which can be seen as the agents genes and funds c from his parent. The initial population is created by assigning a random strategy and an initial reservoir of funds to every agent (line 1). Once started, an agent will continuously follow his strategy by executing f. Whenever c reaches θ the agent will reproduce (line 7 - 9) and create a new agent. This behavior can be translated to indicate the entry of new companies into the market imitating the strategy of existing and successful companies. In the opposite case, if an agent dies, his funds drop below zero ($c < 0$, line 10), construed to signify the bankruptcy of an inefficient company. Therefore the notion of 'generation number' does not exist and instead a perpetual reproduction and replacement of agents takes place. This can be regarded as a combination of biological and economical ideas, because no economic agent would create its own competition. However in biology it is the case.

4.1 Adaption of the Number of Agents

Let us consider a population of agents A, each agent $a \in A$ performing his strategy by executing f_a and receiving profit π_a according to equation (1). Since the profit of an agent may vary from positive to negative thereby leading to the creation of new agents or the death of existing ones. The stepwise creation of a number of new agents $|A'_a(t+1)|$ from an existing agent a (with positive profit) can be expressed in the following way:

$$|A'_a(t+1)| = \lfloor \frac{c_a(t) + \pi_a(t)}{\theta} \rfloor \tag{3}$$

If $c_a(t) + \pi_a(t) < \theta$, the funds $c_a(t)$ is increased/decreased by positive/negative profit $\pi_a(t)$ without leading to the birth of a new agent. To calculate the long-term population growth, we can summarize the profit of every agent beginning with time $t = 0$. According to equation 2 the funds c_a at time $t+1$ can be rewritten as $c_a(t+1) = c_a(t) + \pi_a(t)$ and thus we can express the overall profit of agent a until time $t+1$ as $\tilde{\pi}_a(t+1) = \sum_{i=0)}^{t} \pi_a(i)$. This leads to the calculation of the number of agents created overall $|\tilde{A}'_a|$ until time $t+1$ by a:

$$|\tilde{A}'_a(t+1)| = \lfloor \frac{\tilde{\pi}_a(t)}{\theta} \rfloor \tag{4}$$

For long running systems we approximate the creation of agents in equation 4 from discrete by a continuous adaption in population growth with a result varying at most by 1:

$$|\tilde{A}'_a(t+1)| \quad \leq \quad \frac{\tilde{\pi}_a(t)}{\theta} \quad \leq \quad |\tilde{A}'_a(t+1)| + 1 \tag{5}$$

For sufficient long runs we replace in the following investigation the stepwise adaption by a continuous adaption. In other words, for a particular agent a we use the ratio $\frac{\pi_a(t)}{\theta}$ as an expression of the estimated creation of new agents at time t. In fact, the number of agents increase/decrease stepwise and not continuously, depending on the distribution of $\Pi(t)$ and the funds $C_A = \sum_{a \in A} c_a$ in the population. Using continuous simplification makes it easier to describe population-based effects in the following sections. By summarizing the profit of all agents we get the population-wide profit at time t of the whole agent population by

$$\Pi(t) = \sum_{a \in A} \pi_a(t) \tag{6}$$

Dividing the overall profit $\Pi(t)$ by θ we get the average number of new/dying agents at time $t+1$ (again, averaged for sufficient long runs).

$$|A(t+1)| = |A(t)| + \frac{\Pi(t)}{\theta} \tag{7}$$

Thus $|A(t+1)|$ vary by factor $\frac{\Pi}{\theta}$ compared to $|A(t)|$, where $|A(t)|$ is the number of agents in A at time t. If $\frac{\Pi}{\theta}$ is positive (negative), the number of agents will increase (decrease) depending on Π. With this model we can prove the following theorem:

THEOREM 1. *The number of agents will be adapted so that the overall long-term profit* Π *in the agent population will stay around* $\Pi_0 = 0$.

PROOF. We set the maximum global profit to $\Pi_{max} = \mathcal{R}(\mathcal{D})$ where demand $\mathcal{D}$ is completely turned over in payments $\mathcal{P}$ to the agents. Since agents have to pay tax in each time step, we denote the overall tax with $\mathcal{T}_A(t) = \sum_{a \in A} \mathcal{T}_a(t)$. The first part of the proof corresponds to the case where the number of agents grow and thus approximate Π to Π_0 in future time. We assume a positive profit $\Pi(t) = \Pi_{max} - \mathcal{T}_A(t) > 0$ at time t, gained by agent population $A(t)$. Thus the agent population will grow by $\frac{\Pi(t)}{\theta}$. This leads to the creation of a new set of agents $A'(t)$ and the agent population in the next time step is determined by $A(t+1) = A(t) \cup A'(t)$ in $t+1$. $\Pi(t+1)$ can now be calculated by $\Pi(t+1) = \Pi(t) - \mathcal{T}_{A'}(t+1)$. If $\Pi(t) > \mathcal{T}_{A'}(t+1)$ holds true, the number of agents will further increase by reduced rate $\frac{\Pi(t+1)}{\theta}$. But if $\mathcal{T}_{A'}(t+1) > \Pi(t)$, the profit in next time step will be negative: $\Pi(t+1) < 0$. Thus, according to equation 7 there will be decay in the number of agents. The second part of this proof follows the same argument, but uses opposite signs. Once the number of agents increases, so that $\Pi < 0$, a decay of agents occur. □

This emergent behavior can be observed in real scenarios [8] where actors enter and leave the market. The rate at which agents

enter or leave the market is directly correlated with the overall profit (see equation 7) and the market supply $\mathcal{S}$ and demand $\mathcal{D}$ in such a scenario. Even without any central control this effect can be observed.

4.2 Distributed Problem Optimization by Spread of Successful Strategies

Since in our system no central algorithm can rank and compare all agent funds, no central selection based on fitness can be calculated. Therefore our logistic network must evolve fitter strategies in an emergent self organizing way. In this section we attempt to make an analogy to *Learning Classifier Systems* (LCSs) [6], since these vary in so far that LCSs use a central control instance, additionally to their rule size remains constant.

For the following discussion we assume that strategies were reproduced in a pure way without disturbance of mutation or recombination. During reproduction, a strategy S_a is copied according to its agent a's fund c_a, only if $\pi_a > 0$. More precisely we have only to consider the set of agents with positive profit, to calculate a reproducing probability of S_a. We denote the positive profit $\pi_a^+(t)$ of an agent a with $(\pi_a^+(t) = \pi_a(t) > 0)$ and the set of agents with positive profit with $A^+(t) = \{a | a \in A(t) \wedge \pi_a^+(t)\}$ at time t. The overall positive profit $\Pi^+(t)$ of the whole population $A(t)$ at time t is given by

$$\Pi^+(t) = \sum_{a \in A^+(t)} \pi_a^+(t) \tag{8}$$

As the number of agents changes according to constant fluctuation within the market, we denote the set of newly created agents with A' and the set of eliminated agents with $\bar{A}$. The proportion of a strategy S_a in $A(t)$ is $p_a(t) = \frac{1}{|A(t)|}$, where $|A(t)|$ denotes the number of agents in $A(t)$. Since we want to estimate the ratio of S_a in the population at the next time step, we have first to determine the population itself, as it can grow or shrink. The set of agents in the next time step is determined with $A(t+1) = A(t) \setminus \bar{A}(t) \cup A'(t)$. Using this notation, we can estimate the ratio of strategy S_a by:

$$p_a(t+1) = \frac{1 + \frac{\pi_a^+(t)}{\theta}}{|A(t) \cup A'(t) \setminus \bar{A}(t)|} \tag{9}$$

where $1 + \frac{\pi_a^+(t)}{\theta}$ is the original strategy plus the estimated additional quantity of S_a that together forms the number of examples of strategy S_a in $A(t+1)$. Dividing the number of strategies S_a by number of agents $|A(t+1)|$ in $A(t+1)$ we get a proportion of S_a in the whole agent population. If we recognize that the average profit equals to zero after a sufficient amount of time and for long running simulation we can set $\Pi = \Pi_0 = 0$. According to equation 7 the number of agents in the next time step can be rewritten by $|A(t+1)| = |A(t)|$ and we may rewrite the proportion of S_a at time step $t+1$ as follows:

$$p_a(t+1) = \frac{1 + \frac{\pi_a^+(t)}{\theta}}{|A(t)|} \tag{10}$$

It follows, that based on the proportion $p_a(t+1) > p_a(t)$ for a positive profit π_a^+ the part of S_a in A grows. In words, the proportion of a particular strategy S_a grows as the ratio of the average profit π_a of agent a. Strategies causing a positive profit will receive an increasing number of replications in the next time step, while strategies generating a negative profit will receive a decreasing number of copies.

It follows that strategies inducing a positive profit on the one hand may have to pay less tax compared to below zero profit strategies due to the usage of less resources or a better resource utilization. Both are needed to streamline logistic networks while simultaneously improving service to the customer. Based on equation 1 one can see that tax (T) is that part of an agents profit determining variable which is not explicitly related to the flow of goods. With tax the models basic conditions can be set resp. controlled and the agent population will adapt to it, if tax is not too high. Therefore, an intrinsic property of systems using our model is the constant search for better resource utilization.

5. IMPLEMENTATION

This section presents results obtained by using the described method within our scientific HoPIX (Hohenheimer Process Integrator eXtended) system for logistic network adaption. The agent system is DIET [14] because it provides a bottom-up approach that can run a large number of agents. Additional to the model presented in section 3, every agent has some parameters described in the following section. A so called *familyTag* which indicates the process step an agent is executing. Further, every agent owns a *partnerTag* indicating the previous process step. This tagging can be seen as property of complex adaptive systms described by Holland [12] as one of the basic characteristics in adaptive complex systems. Using tags, agents can identify and connect to each other in order to fulfill their tasks. In DIET an agent needs a family tag to connect to another agent. Specifying a tag will connect an agent to a random agent with this tag. Once connected it is possible to connect to this particulary agent again using its `AgentIdentity`. In our application we use tags to determine trading partners in the logistic network (using *partnerTag* as described above) and mating partners for the evolutionary computation.

In order to setup a flexible logistic simulation, we use two types of agents: `StorageAgents` and `TransportAgents`. These differ in the way they handle goods that flow through the network. A `StorageAgent` tries to fill its store to capacity and then waits for possible orders to deliver the goods immediately. The `TransportAgent` waits for an order from a buyer subsequently searching a `StorageAgent` who can then turn the goods over in time. Using these basic logistic functions, goods can flow through a network of logistic agents connected according via to a process. We consider every process step as either a storage or a transportation step.

The flow through the network is determined by three facts: 1.) the number of agents who executing every process step and 2.) their transport respective store size. The latter fact is encoded as a parameter string for every `TransportAgent` agent as $S(transportcapacity)$ and `StorageAgent` as $S(storesize)$. The third fact is the provided demand and money from the environment. At the start a number of agents with initial parameter settings are created for every process-step.

The agents strategy S is represented as an `GeneticCode` object, with all parameters stored in a vector. Each parameter is encoded as an integer that can be evolved using the common variable-length binary valued crossover and mutation operators [6]. We use uniform crossover through all runs. In fixed intervals agents send their `GeneticCode` objects in a special message within further information to a randomly chosen agent pertaining to the same process step (resp. using the same *familyTag*). This allows a local selection amongst the same agent families. For local selection methods we have implemented the following four different variants:

- *fund*: the receiving agent selects his partner on the basis of funds (of the sending agent) provided together with the `GeneticCode` object. Here the receiving agent stores the `GeneticCode` provided with the highest fund for future reproduction. As the fund of an agent may vary between 0 and θ this method cannot be seen as appropriately reflecting an agent's fitness.
- *lifetime*: the recipient agent selects and stores the `GeneticCode` object provided by the sending agent with the longest lifetime. This may also not reflect an appropriate representation of success as an agent can survive a long time without creating a child.
- *nr-of-childs*: the receiving agent selects and stores the `GeneticCode` object provided by the sending agent with the most children created.
- *mutation-only*: the agent does not use any foreign `GeneticCode`, but only uses mutation.

5.1 Example Scenario: Logistics Network

As an example of a distributed optimization problem, we studied the logistic network described in section 1. We used the given process in figure 1a) where the environment participates from the consumer side. At random intervals, consumers come to the market for a single transaction given by a uniform distribution process $U[0, orderinterval]$ ranging from 0 to the maximum order interval $orderinterval$ to prevent bursts in order placement. The ordering time for one consumer is calculated by

$$t_{order} + 1 = t_{order} + U[0, orderinterval].$$

The order size $ordersize$ is fixed to a specific value. Producer agents are `StorageAgents` that 'generate' their goods traded in the logistic network determined by $producetime$. At fixed intervals $producetime$ a producer generates a unit of goods until his storage capacity $S(storesize)$ is reached. The time steps for generating goods are calculated as follows:

$$t_{producetime} + 1 = t_{producetime} + producetime.$$

For our simulation we consider non-varying transport ways of `TransportAgents`, i.e. the time needed for moving goods from source to buyer remains constant.

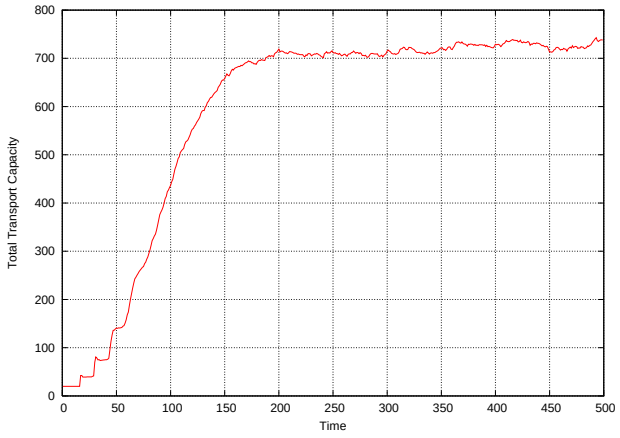

Figure 3: Summarized transport capacity

We concluded an experiment to analyze the adaption of the agent system to a given process shown in figure 1a) where the process consists of four steps: production, transportation, stocking, and consumption. The number of producers was set to 50, the time for producing a new good was set to $producetime = 500ms$. The parameters for the U are $orderinterval = 10000ms$, $ordersize = 10$, number of consumers $= 50$. We started in the initial configuration with one transport agent $a1$ and one store agent $a2$, thus the multi-agent system consists of fifty (producers) plus the two agents (i.e. one `StorageAgent`, one `TransportAgent`) that form the initial logistic network. The initial strategy parameter for transport agent $a1$ is $S_{a1}(transportcapacity) = S_{a2}(storesize) = 20$. The tax function is given with

$$T(S) = 0.06 * S(transportcapacity) + 0.06 * S(storesize).$$

Further, the number of customers and the number of producers remain constant and the `mutationprobability` is set to 0.03. The selection method is set to *fund* and θ is dependent on the agents capacity calculated as follows $\theta = S(transportcapacity) * 2$ respective $\theta = S(storesize)*2$. For the discussion of the mentioned effects (section 4) we focus on the transport agents. All results are obtained by averaging 50 simulation runs.

5.2 Simulation Results

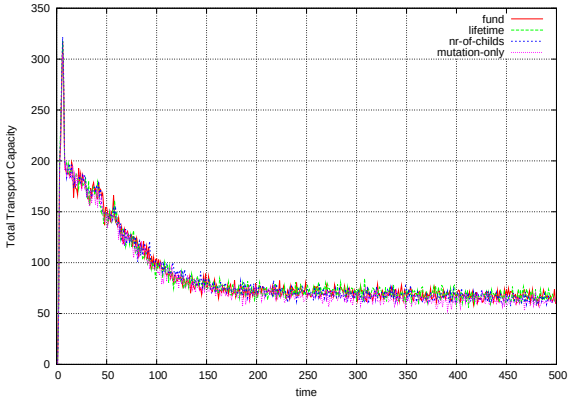

Figure 4: Total missed orders compared by different local selection strategies

In figure 3 the total transport capacity of all transport agents is shown. An interesting result was obtained at the beginning of the simulation, where the number of agents rose quickly and therefore until time step 140 the transport capacity soared steeply from an initial 20 to 715. One time step in our simulation lasts 2 seconds. This effect was induced by the high occupancy rate of every transport agent a at beginning of the simulation causing high profit π_a. After time step 140 the total transport capacity varied between 700 and 900 and due to saturation effects a balance between demand and tax T emerged. Common to all simulation runs is the visibility of the step-wise rise in the number of agents but due to averaging effects this behaviour is observable only at the beginning.

Every time a consumer agent tries to order and the store agent refuses the request (because subsequent transport agents are busy), a missed order occurs (figure 4). We executed all experiments with the four different local selection methods described in section 5. A remarkable result for this scenario was that all four methods show the same performance. We believe that this effect occurs due to the small set of parameters. Consumers place a single order to a store agent within their order interval $orderinterval$. In the starting period the number of missed orders decreases quickly from initial values between 200 and 300 down to 100 at time step 140 due to the increased total transport capacity. After the first steep

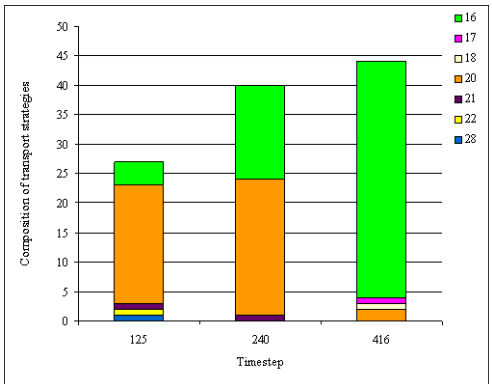

Figure 5: Adaption of transport strategies at three snapshots

decline in missed orders, a slight decreasing takes place. This effect originates from the slightly changing strategies of the transport agents by EC mechanism. Figure 5 shows the adaption of transport agent parameter $S(transportcapacity)$ (of a single run) among the population of transport agents at three selected milestones at time steps 125, 240 and 416. The chart shows a snapshot of the distribution of different transport capacities inside the population of transport agents. At time step 125 the dominant strategy is $S(transportcapacity) = 20$ whereas $S(transportcapacity) = 16$ occurs only 4 times. The last snapshot at time step 416 shows that $S(transportcapacity) = 16$ has spread out nearly in the whole population (40 out of 44 transport agents). As expected the more adapted strategies will receive an increasing number of replications in subsequent time steps. One consequence of the spread of low tax strategies is that the population size can grow while the overall transport capacity remains the same.

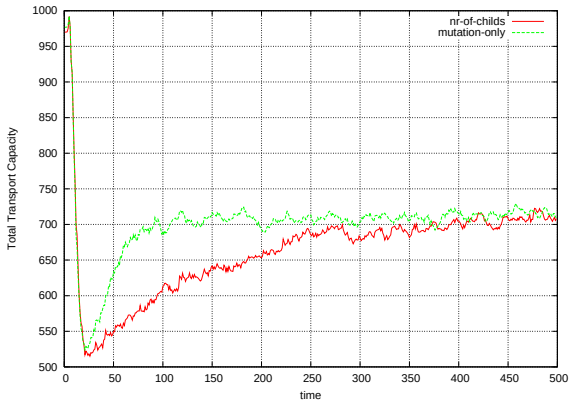

Figure 6: Adaption of the total transport capacity after initialization with 50 `TransportAgents`

We now change the settings so that at the initiation of every process step (transport, store) there are 50 agents with randomly assigned parameters as follows:

$$10 \leq S(transportcapacity) = S(storesize) \leq 30.$$

Figure 6 depicts the progress of the total transport capacity starting with a high value (970) due to a high initial transport capacity. This high number plummets steeply almost immediately after system start, because the provided funds from U do not generate enough profit Π for the whole population. After a consolidation phase the population grows and stabilizes. We compared the *mutation-only* with the *nr-of-childs* method. An interesting fact is that while *mutation-only* shows a more adaptive behaviour, in the final result they are both equal. *mutation-only* allows a quicker spread of successful strategies due to less disturbance by recombination effects. Given this fact, the total transport capacity shows similar effects as illustrated in figure 7. Here the total transport capacity with *mutation-only* shrinks quickly to the equilibrium, whereas the *nr-of-childs* method exhibit a short oscillation due to disturbance effects.

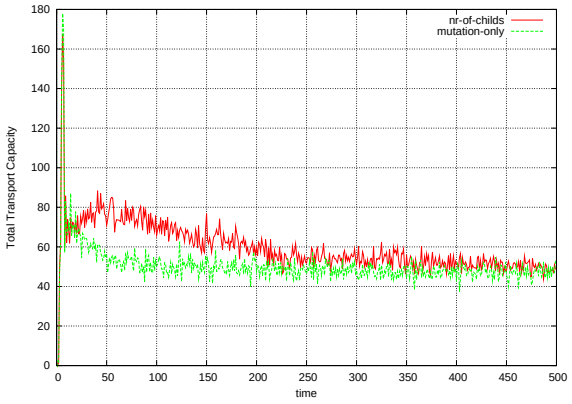

Figure 7: Total missed orders over time after initialization with 50 `TransportAgents`

6. CONCLUSION AND FURTHER DIRECTIONS

This paper has addressed the adaptivity of agent systems whereas an evolutionary computation approach is used. All necessary steps of the evolutionary algorithm have been moved to the agents itself which allows a fully decentralized approach. We have shown with our approach, that even in a decentralized system architecture without any central control an adaption of two basic properties occur. First the number of agents $|A|$ adapt according to the demand provided from environment U and second, successful strategies spread in the population of agents. The adaption process is based on funds provided by U that allows local selection and reproduction by the agents. The two effects indicate that an agent population and their resource utilization adapt to a pre-defined demand and will constantly continue to adapt thereby satisfying the demand with lower resource utilization. We have applied our method in a logistics scenario forming a process based network of agents who structure a logistic network. In this scenario objects (e.g. goods, information) flow through the network starting from the producer, crossing all steps of the process and finally reach the environment which created the original demand for goods and gives funds in return.

Further research in this line will pursue the development of a more generalized method of adapting multi-agent systems using an economics-enabled market-based computation approach. We also hope to address other useful application fields for this method, e.g. in the context of grid computing and agent systems. As discussed in section 5, our proposed logistic simulation has limitations when considering geography or real world maps, especially we use constant transport distance. Our future works includes developing a more detailed logistic scenario model, the support of the complete set of base logistic functions as well as the use of geographic maps.

7. REFERENCES

[1] A. Babanov, W. Ketter, and M. Gini. An evolutionary approach for studying heterogeneous strategies in electronic

markets. In G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Enigneering Self-Organising Systems*, 2977, pages 157–168. LNAI, Springer Verlag Berlin Heidelberg, 2004.

[2] K. Bichler and N. Schröter. *Praxisorientierte Logistik*. W. Kohlhammer GmbH Stuttgart, 2004.

[3] W. Cai, S. J. Turner, and B. P. Gan. Hierarchical federations: an architecture for information hiding. In *PADS '01: Proceedings of the fifteenth workshop on Parallel and distributed simulation*, pages 67–74, Washington, DC, USA, 2001. IEEE Computer Society.

[4] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Trends in cooperative distributed problem Solving, July 1995.

[5] T. Eymann. *AVALANCE - Ein agentenbasierter dezentraler Koordinationsmechanismus für elektronische Märkte*. PhD thesis, Universität Freiburg, 2000.

[6] D. E. Goldberg. *Genetic Algorithms in search, optimization, and machine learning*. Addison Wesley Longmann, Inc., 1989.

[7] H.-W. Graf. *Netzstrukturplanung - Ein Ansatz zur Optimierung von Transportnetzen*. PhD thesis, Universität Dortmund, Fakultät Maschinenbau, Dortmund, 2000.

[8] V. Graudina and J. Grundspenkis. Technologies and multi-agent system architectures for transportation and logistics support: An overview. *International Conference on Computer Systems and Technologies - CompSysTech*, 2005.

[9] S. Guenther and F. Laakmann. Efficient evaluation and selection of it-support based on the supply chain management task reference model. http://www.scm-ctc.de/, 2001.

[10] H. Heinrichmeyer and A. Reinholz. Entwicklung eines Bewertungsmodells für die Depotstandortoptimierung bei Servicenetzen. Technical report, Fraunhofer-Institut für Materialfluss und Logistik, Universität Dortmund, Fachbereich Informatik, Lehrstuhl für Systemanalyse, 2004.

[11] F. Hohl. A framework to protect mobile agents by using reference states. Technical Report 2000/03, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), March 2000.

[12] J. H. Holland. *Hidden Order: How Adaptation Builds Complexity*. Addison Wesley Publishing Company, September 1996.

[13] E. Jehle and M. Kaczmarek. Organisation der Planung und Steuerung in Supply Chains. Technical report, Universität Dortmund, Lehrstuhl Industriebetriebslehre, 2004.

[14] P. Marrow, M. Koubarakis, R. van Lengen, F. Valverde-Albacete, E. Bonsma, J. Cid-Suerio, A. Figueiras-Vidal, A. Gallardo-Antolín, C. Hoile, T. Koutris, H. Molina-Bulla, A. Navia-Vázquez, P. Raftopoulou, N. Skarmeas, C. Tryfonopoulos, F. Wang, and C. Xiruhaki. Agents in decentralised information ecosystems: the diet approach. In *Proceedings of the Artificial Intelligence and Simulation Behaviour Convention*, pages 109–117, March 2001.

[15] F. Menczer and R. K. Belew. Adaptive retrival agents: Internalizing local context and scaling up to the web. *Mach. Learn.*, 39(2-3):203–242, 2000.

[16] F. Menczer, M. Degeratu, and W. Street. Efficient and scalable pareto optimization by evolutionary local selection algorithms. *Evolutionary Computation*, 8(3):223–247, 2000.

[17] H. Pfohl. *Logistiksysteme*. Springer Verlag Berlin Heidelberg, 2000.

[18] R. E. Smith, C. Bonacina, P. J. Kearney, and T. Eymann. Integrating economics and genetics models in information ecosystems. In *Proceedings of the Congress on Evolutionary Computation (CEC 2000)*, La Jolla, USA, July 2000.

[19] R. E. Smith and N. Taylor. Evolutionary computation in agent-based systems. In C. Looney and J. Castaing, editors, *Proceedings of the 1998 International Conference on Intelligent Systems*, pages 221–224. ISCA Press, 1998.

[20] R. Whitaker. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. *INFOR 21*, pages 95–108, 1983.