The CxBR Diffusion Engine – A Tool for Modeling Human Behavior on the Battle Field

Hans Fernlund, Sven Eklund* and Avelino J. Gonzalez

Intelligent Systems Laboratory
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2450
hfe@du.se, gonzalez@pegasus.cc.ucf.edu

*Computer Engineering Department
Dalarna University
78188 Borlänge, SWEDEN
sek@du.se

Abstract

The option to automatically model the behavior of different actors during live exercise training would increase the value of the after-action-review (AAR) process. If a simulated model of the actors is available right after the live exercise training, the evaluation of their behavior would be more timely and alternative actions could also be evaluated at the same time. The CxBR Diffusion Engine merges technologies to establish a tool for automatic, on-line behavior modeling. Context Based Reasoning (CxBR) is a proven methodology to build simulated agents with human behavior. Genetic Programming (GP) provides the CxBR framework with learning capabilities to automatically create simulated agents with human behavior. The final piece in the CxBR Diffusion Engine is to provide an efficient, flexible, scaleable and mobile platform to evolve the agents' behavior. This platform is the newly developed massively parallel architecture for distributed GP. The massively parallel architecture has the potential to execute the GP linear machine code representation at a rate of up to 50,000 generations per second. Implemented in an FPGA, this architecture is highly portable and applicable to mobile, on-line applications. This paper will present a theory on how the CxBR + GP can evolve simulated agents with human behavior by observation in a massively parallel architecture. These pieces will introduce all the necessary elements to build the CxBR Diffusion Engine that could model human behavior to enable individual AAR of trainees in the training field.

Introduction

The objective of this research is to automatically model human behavior by looking at a human in action. Furthermore, the aim is to either apply these models in a simulator or to use the models to evaluate the behavior of the actor. To model humans in their natural training or operational environment, the automatic modeling tool must be portable. The aim is to have the models ready to use shortly after the monitoring process. To ensure rapid use of the models, the modeling needs to be done on-line. On-line learning is performed at the same time as the actors are performing their tasks. Here we present techniques how this could be accomplished by using a fast executing and portable GP hardware architecture to evolve human performance knowledge in a CxBR framework very quickly and easily. Before describing the technical details, we begin by discussing what could be gained by modeling from observation.

Learning by Observation

Inspired by how humans and other mammals learn by observation, the machine learning community has developed a number of theories on learning by observation,

applied in various areas. The interest in this research is to investigate learning human behavior by observation. The intent is not only to use the observations to learn, but also to learn the behavior of the observed entity. A number of advantages could be gained by using learning by observation instead of traditional knowledge acquisition and development methods:

- Reduce time and cost of development, debugging and maintenance.
- More accurate, realistic and refined representation of human behavior.
- Potential to incorporate learning from experience.
- Potential to incorporate new features of human-ness, such as emotions.
- Relaxing operators programming skills.
- Reduction of problem domains.
- Develop simulated entities in real time.

The research described in this paper defines learning by observation as follows:

> *The agent adopts the behavior of the observed entity only through the use of data collected through observation.*

To create a behavioral model with vide variety of human features there must exist an efficient modeling framework. Context-Based Reasoning (CxBR) was proposed by Gonzalez and Ahlers (1998) to have many of these features. Using CxBR as a framework satisfies the prerequisites for implementation of human behavior. If the model is to be created automatically, the framework needs to be equipped with a learning paradigm that will work in conjunction with the framework without disturbing the supported human features. The learning paradigm used in this research is Genetic Programming (GP).

Context-Based Reasoning

Gonzalez and Ahlers (1994) presented Context-Based Reasoning (CxBR) as a modeling technique that can efficiently represent the behavior of humans in intelligent software agents. Later results showed that it is especially well suited to modeling tactical behavior.

CxBR is based on the idea that:

- A recognized situation calls for a set of actions and procedures that properly address the current situation.
- As a mission evolves, a transition to another set of actions and procedures may be required to address the new situation.
- Things that are likely to happen while under the current situation are limited by the current situation itself.

CxBR encapsulates into hierarchically-organized *contexts* the knowledge about appropriate actions and/or procedures as well as compatible new situations.

Mission Contexts define the mission to be undertaken by the agent. While it does not control the agent per se, the Mission Context defines the scope of the mission, its goals, the plan, and the constraints imposed (time, weather, rules of engagement, etc). The *Major Context* is the primary control element for the agent. It contains functions, rules and a list of compatible next Major Contexts. Identification of a new situation can now be simplified because only a limited number of all situations are possible under the currently active context. *Sub-Contexts* are abstractions of functions performed by the Major Context which may be too complex for one function, or that may be employed by other Major Contexts. This encourages re-usability. Sub-Contexts are activated by rules in the active Major Context. They will de-activate themselves upon completion of their actions.

One and only one specific Major Context is always active for each agent, making it the sole controller of the agent. When the situation changes, a transition to another Major Context may be required to properly address the emerging situation. For example, an automobile may enter an interstate highway, requiring a transition to an **InterstateDriving** Major Context. Transitions between contexts are triggered by events in the environment – some planned, others unplanned. Expert performers are able to recognize and identify the transition points quickly and effectively.

CxBR is a very intuitive, efficient and effective representation technique for human behavior. For one, CxBR was specifically designed to model tactical human behavior. As such, it provides the important hierarchical organization of contexts. A full description of CxBR can be found in Gonzalez and Ahlers (1998).

Genetic Programming

Genetic Programming (GP) is derived from Genetic Algorithms and both are stochastic search algorithms. The search process looks for the best suitable program that will solve the problem at hand. The target system for the GP could be a CPU, a compiler, a simulation or anything else that could execute the pre-defined instructions, from now on referred to as a *program*. GP evolves source code representing a program that can address a specific problem. This makes it very suitable for use together with CxBR. GP can build complete software programs that support the internal construction of the CxBR (i.e. the context-base).

To make GP work, some basic requirements must be satisfied. First, we need to have a set of individuals (i.e. programs that represent different solutions to the problem). Furthermore, all the individuals need to be evaluated in

some manner as to what degree they are able to solve the problem. The features of the individuals with better suitability would preferably be preserved and survive, or breed new individuals to the next generation. The next GP step would be to evolve the individuals (i.e. reproduction) in some manner to preserve the "good" features and develop even better individuals. The most common genetic operators are *crossover* and *mutation*. They support the development and evolution of the individuals. Evolving a program with GP can be described in three steps, see Figure 1.

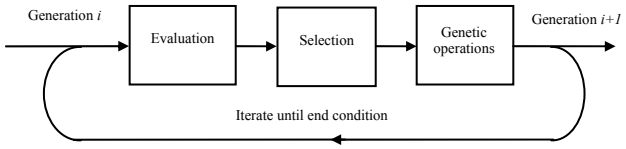

Figure 1: The three steps of GP

The criteria for stopping the evolutionary process can be a maximum number of evaluations made, a maximum number of generations evolved, the fitness reaching a certain level, or other measurable criteria given. When the evolutionary process is finished in GP, there then exists a program that will solve the problem.

Since GP builds source code, it could be used to incorporate knowledge in any context level or in any instances within CxBR where human behavior is encoded. This means that we could choose to implement learning in any specific part of CxBR and construct the knowledge therein.

Towards automation

Human behavior within CxBR can be categorized in two groups: action rules and sentinel rules. Rules, in this case describes knowledge containers. These containers can contain production rules, functions, operators and complex data structures. At the level of any Context, the sentinel rules determine whether their own Context will be active for that specific context level. The structure of these sentinel rules are similar in all contexts at all levels. Each context has its own set of sentinel rules that determine whether this context should still remain active or if it should turn over the control to another context at the same level or at a higher level. This can be viewed as state transition rules where each state (i.e. context) has its own transition table. At the Major Context level, the action set tends to be more a collection of Sub-Contexts and less of other functions, rules or variables. At lower context levels, the action set is less composed of Sub-Contexts and at the lowest context level, there are no Sub-Context calls. Experience has shown that three or at most four, levels of contexts, including the Mission Context, are normally sufficient.

From the discussion above, we can conclude that if we want to incorporate learning into CxBR, the learning paradigm must be able to learn the applicable behavior in a specific context (i.e. action rules) and also the appropriate context switches (i.e. sentinel rules). The type of learning in those two knowledge containers is usually quite different. To learn specific action patterns, the learning regularly becomes somewhat of a regression problem where the model is trying to minimize the discrepancies to the human's performance. When it comes to choose the right context for the current situation, the learning process is more of a classification problem. A learning paradigm that can adapt well to those different learning problems is GP.

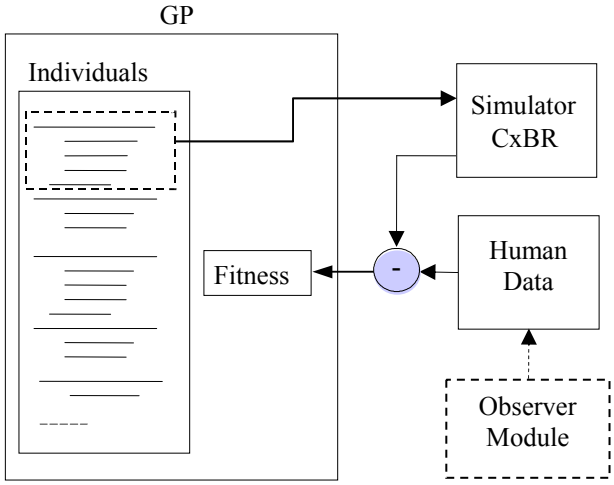

Figure 2: Learning by Observation: CxBR+GP

Instead of creating the contexts by hand, we use the GP process to build the contexts. The GP's evolutionary process provides the CxBR frame with appropriate context's action rules and sentinel rules. The individuals in the genetic population are components of the context base and a simulator is used to simulate an individual's behavior. The behavior from the simulator is then compared with the human performance, and a fitness measure is established to evaluate the models appropriateness. See Figure 2. The evolutionary process will strive to minimize the discrepancies between the performances of the contexts created by GP and the human performance. The human data is the observed human performance, or rather appropriate parts of the performance selected by the observer module. The features of CxBR and GP show that their combination could be a feasible approach to learning by observation. Through this synergistic combination, a system could be constructed

that, by observing a human, is able to build a context base for simulated entities that exhibit human behaviors.

The *Observer Module* in Figure 2 has not yet been implemented. In the results described later the objective was to investigate the feasibility of integrating CxBR and GP to achieve our stated objectives. The intended use of the observational module is to select appropriate data to be used in the learning process. Since the computational task was too complex to employ the complete data set this has been done manually so far.

A positive feature of using GP as the learning algorithm is that it preserves many of the features of CxBR that makes it an appropriate paradigm to model human behavior. The tools GP uses to store the knowledge learned is the same tools as a programmer or knowledge engineer would have used developing the knowledge base - source code statements. The agents are able to express their action in a way that is understandable to humans. This implies that the knowledge is easy to interpret if we want to include communicative features within the agents. It is easy to interpret source code and convert it to written language.

Parallel GP

To implement the CxBR+GP concept for constructing human performance models through observation quickly and effectively, we propose a, hardware Diffusion Engine. To achieve this, rapidly on the field, two main issues need to be addressed: mobility and computational complexity. GP is highly computationally complex. To apply GP to a complex problem such as model human behavior and to do it rapidly increase the computational complexity even more. The approach taken here is to present a new massively parallel architecture implemented in hardware to address the computational complexity and yet be able to present a mobile solution.

By distributing independent parts of the genetic algorithm to several processing elements that work in parallel, it is possible to speed up the calculations significantly. This can be done in several ways. The most obvious is to use the independence between individuals, at least during the evaluation step, and evaluate the individuals in parallel. Traditionally, parallel models have also been categorized by the way they handle the population. The choice between a global and a distributed population is mainly a choice on how individuals are accessed and how much communication this will require. The choice between a global and a distributed population is also a decision on GP's selection pressure, since smaller, distributed populations have higher selection pressure, resulting in faster (sometimes premature) convergence.

The Diffusion Model

One of the most interesting parallel models is the diffusion model. This fine-grained model distributes its individuals evenly over a topology of processing elements. It can be interpreted as a global population laid out on a structure of processing elements, where the spatial distribution of the individuals defines the subpopulations or, more appropriate, the neighborhoods.

Every node holds only a few individuals, most often only one or two, and the number of nodes is normally high, making it a massively parallel model with the potential of reaching high speed-ups. The migration in the Diffusion Model is implicit, where fit individuals are allowed to "diffuse" throughout the population.

This is possible since the neighborhoods are defined as the closest proximity around each node over some topology. Since every node is in the center of exactly one neighborhood, these neighborhoods will overlap, making every node part of several neighborhoods. Selection is performed in parallel within these local neighborhoods, and only the center node will be updated in every neighborhood.

In figure 3 the neighborhoods consist of five nodes. During evaluation all nodes evaluate their individuals in parallel. The second step of selection in then done, in parallel, in all neighborhoods. In the example of Figure 3, this means that every node is part of selection processes in five different neighborhoods. However, the selected individual will only update the center node of that neighborhood. Finally, the third step of genetic operations is also done in parallel in all nodes.

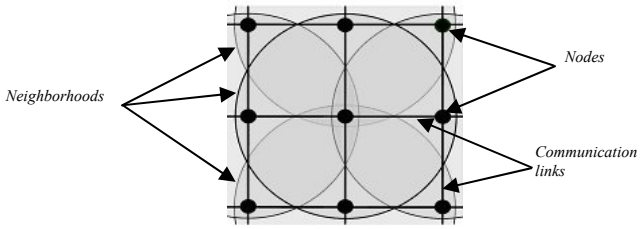

Figure 3: Diffusion Model with a 2D linear topology

An important benefit of the Diffusion Model is that it is well-suited for VLSI implementation since the nodes are simple, regular and mainly use local communication. Since every node has its own local communication links over the selected topology (1D, 2D, etc), the communication bandwidth of the system can be made to scale nicely with a growing number of nodes. Furthermore, the nodes operate synchronously and have small, distributed memories,

which also make the Diffusion Model suitable for implementation in VLSI.

In Schwehm, (1996) several other advantages of the Diffusion Model are concluded, most notably the absence of an explicit migration parameter and a potentially higher parallelism than in other parallel models.

The Diffusion Architecture

As noted above, the Diffusion Model is a promising model as such, but it is even more so when considering hardware implementation. VLSI technology, as noted by Kung and Leiserson (1979) during work on systolic arrays, benefits from architectures with small elements, regular structure, local communication, synchronous operation, low I/O and small local memories – all of which are features of the Diffusion Model.

The node

A good starting point when describing the diffusion architecture is to define its basic building block; the node. An architecture based on the Diffusion Model consists of a large number of such nodes. All nodes are identical and are connected to their neighboring nodes over some topology. See Figure 4. They all hold their own unique data (representing their individuals) and they all have the capability to evaluate their individuals and perform the GP algorithm in parallel with the other nodes. See Figure 5.

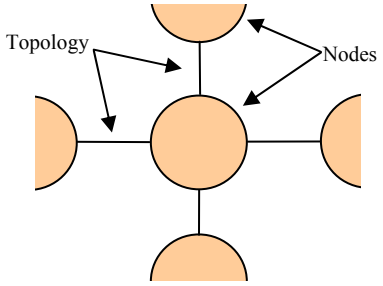

Figure 4: The node in a two-dimensional topology

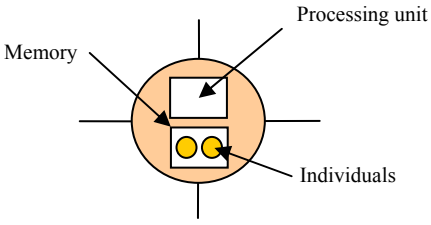

Figure 5: The two major parts of the diffusion node.

Topology

The topology is the "geography" of the communication network in the Diffusion Model - the way nodes are connected to each other. Since fit individuals spread throughout the population over this topology, it is likely that the topology will play an important role in the implementation.

The most common topologies are one- and two-dimensional networks even though higher order ones and tree-like structures are possible. In Figure 6 three common topologies are shown.

Please note that the "edges" of the network of nodes has to be dealt with. Most often the edges are connected "wrap-around" and thus making a one-dimensional line a ring and the two-dimensional grid a donut-shaped toroid.

One of the three major objectives of the design is to make the system scaleable. When implementing the topology in physical chips, the I/O-capability of the chips has to be considered. For instance, implementing 10.000 nodes with an X-net topology in chips that can hold 50*50 = 2.500 nodes would require four chips. Since nodes and their individuals are independent of each other (besides the communication over the X-net), this scales nicely over the four chips that divide the work equally. The wiring needed between the chips, however, is likely to be beyond the capacity of the chips (4 sides * 50 nodes * 3 links = 600 logical links are needed from each chip). Some kind of supporting communication hardware between the chips would probably be needed in this case.

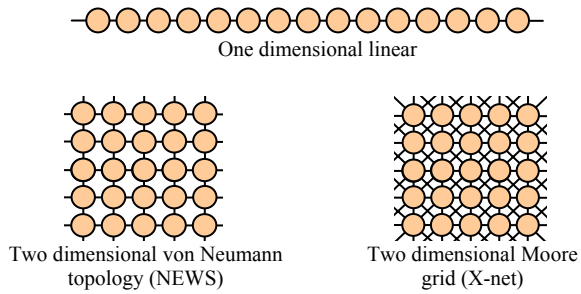

Figure 6: Three "mesh-like" topologies

Neighborhood shapes

Upon the topology of the system, neighborhoods are defined, both by their shape and size. It is within these neighborhoods that selection is performed. Every node is in the center of exactly one neighborhood, but it is also part of, and contributes to, several other neighborhoods, depending on the neighborhood shape. It is only the center node of each neighborhood which is updated by the algorithm.

It is the overlapping of these neighborhoods that enable the implicit movement or diffusion of fit individuals throughout the population/topology. Larger neighborhoods will have a greater overlap and will therefore spread fit individuals faster than smaller ones. Also, using topologies with different dimensionality will spread fit individuals differently.

The distributed GP

In the distributed population model of the Diffusion Model the basic steps of the genetic algorithm from Figure 1 are performed in parallel in all nodes. Since the population is distributed and communication between individuals is limited, these basic steps are somewhat different than in the standard GP.

Fitness evaluation

The node and the representation of its individuals have to incorporate the ability to evaluate very different individuals even though all the nodes have to obey identical control signals from the single control unit. This could be solved by either a very general representation (data parallel evaluation), by an embedded evaluation unit under local node control or by a combination of both.

Selection algorithms

Traditional selection algorithms such as roulette or ranking selection require global fitness or ranking averages to be calculated, distributed and maintained. Implemented in a parallel model, this often introduces a communication bottleneck which will limit the performance of the parallel system. By using a local selection pool only, that is, the local neighborhood of the Diffusion Model, this communication problem can be managed satisfactorily. However, this also means that the standard GP is changed for most selection algorithms and it is not obvious how this will affect the overall performance of the system.

Genetic operations

Mutation is a stochastic change of a single individual, independent of all other individuals. It can therefore be performed in parallel over all nodes without any communication.

Crossover is the recombination of two individuals and may therefore require some communication, depending on the population density and on the selection algorithm. The implementation is dependent on representation but also on the type of crossover.

Representation

The three steps described above are all heavily dependent on the choice of representation. The choice of representation for the individuals is also central in order to achieve the main objectives of effectiveness, flexibility, scalability and mobility.

Since the evaluation of the individuals is responsible for a large portion of the total execution time (for most applications), the representation has to be fast to interpret and evaluate. It has also to be storage efficiently ("hardware friendly") since fewer transistors per individual means more individuals (or nodes) per chip which most often translates to shorter overall execution time. Fewer transistors per individual could also be translated into smaller size with the same number of individuals (nodes), which benefits the objective of mobility. Finally, the representation has to be flexible enough to efficiently support many different applications with minimal changes to the design.

We propose the use of linear machine code as GP-representation. Clearly, it is storage efficient and it is very efficient to evaluate for a CPU. Using general purpose registers it also supports reuse of results and easy parameter passing.

Since the code and therefore also the behavior differ between individuals, each node has to have its own resource to evaluate its individuals (which basically is the whole point of a parallel system). Using linear machine code GP it is natural to equip every node with its own CPU.

This also means that the system can quite easily adapt to new applications. The solution space of the application (the space of all possible solutions) is simply defined by the Instruction Set Architecture, ISA. By redefining the CPU and its ISA the architecture can be fine-tuned for each application. Using a hardware description language like VHDL, this is easily accomplished.

A general structure of the node architecture is shown in Figure 7.

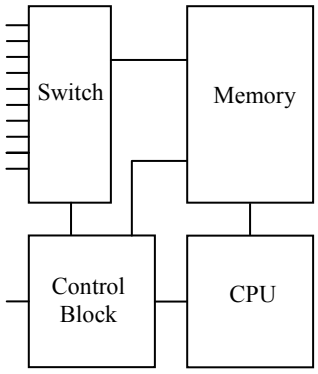

Figure 7: The four blocks of the diffusion node

Experimental Results

The validity of combining CxBR and GP to learn human behavior only from observational data will be shown in Fernlund's upcoming dissertation. The new approach to learning by observation was applied to simulated automobiles. The experiments collected data from humans operating a simulated automobile and the new approach evolved a stable agent, with all behavior pattern learned, able to operate in a simulated environment. One of the most interesting results was that the automated model was as good as a model created by traditional means (i.e. manually by an independent knowledge engineer/programmer).

The Diffusion Model has been extensively studied through simulations in order to define some of its most important parameters, for instance selection algorithm, topology and neighborhood size. Please refer to Eklund (2003a) for a detailed description of these results.

The Diffusion Model has also been tested on several benchmark applications. In Eklund (2003b) the model proved to work well on the very difficult task of handwritten character recognition. In Eklund (2003c), the model was shown to outperform some of the most well-known models for time-series forecasting.

Recently, a first prototype of the diffusion architecture has been implemented in VLSI. Preliminary results indicate that one node of the diffusion architecture, holding one individual, can be implemented using roughly 30,000 gates. Using a symbolic regression problem as application and a maximum individual size of 32 words, the entire population can be updated at a rate of 50,000 generations per second, regardless of population size.

Conclusions and Further Work

In this paper we have presented the necessary theories to create a rapid, on-the-field human behavior modeling tool. Results have shown that GP and CxBR are able to evolve simulated agents with human behavior, solely by observations. The new Diffusion architecture has shown very good simulation results and the first hardware implementation is currently being built. The theories presented here fulfill the prerequisites for enabling the creation of tool that could be used to rapidly build human behavior models on the field.

Even if the different parts show good results, further research is needed to make them work together. First the CxBR and GP module needs to modify the tree structure representation of the individuals to suit the linear machine code representation of the diffusion architecture. To make the theory of learning by observation complete, which also will make the approach more generic and widely applicable, further research needs to be conducted within the observation module. This module will completely automate the modeling by observation and analysis of the environment. It is also necessary to develop the observer module to enable on-line learning. Furthermore, additional adjustment is probably needed from both hardware and the algorithmic side to complete the CxBR Diffusion Engine.

Completing the CxBR Diffusion Engine would open possibilities of model human behavior and automatically create simulated agents with the same behavior, without the observed human being aware of it. In certain situation it could be preferable to not declare the modeling to the actors prior to the experiment. The modeling could also be done in a live event and not during a simulation. The possibility could be to even model hostile troops. We could also use the models in After Action Reviews and either use the models in simulated actions or evaluate the knowledge stored in the models. Since the knowledge created by the GP module is transparent, it is possible to understand and use the knowledge learned.

References

Eklund, S, "Empirical Studies of Neighbourhood Shapes in the Massively Parallel Diffusion Model", Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence no. 2507, ISBN 3-540-00124-7, Springer-Verlag, 2003a.

Eklund, S, "Handwritten character recognition using a massively parallel GP Engine in VLSI", IFAC Conference on Intelligent Control and Signal Processing, Faro, Portugal, April 2003b.

Eklund, S, "Time series forecasting using massively parallel genetic programming", The Fifth International Workshop on Nature Inspired Distributed Computing, NIDISC'03, Nice, France, April 2003c.

Gonzalez A. J. and Ahlers, R. H. "A Novel Paradigm for Representing Tactical Knowledge in Intelligent Simulated Opponents" Proceeding of the IAE/AIE Conference, May 31-June 3, 1994, Austin, Texas.

Gonzalez A. J. and Ahlers, R. H. "Context-Based Representation of Intelligent Behavior in Training Simulations" Transactions of the Society for Computer Simulation International, volume 15, no 4, December 1998

Kung, H. T, Leiserson, E, "Systolic Arrays (for VLSI)" in I. S. Duff and G. Stewart (editors), Sparse Matrix Proceedings, Knoxville, Academic Press, 1979.

Schwehm, M, "Parallel Population Models for Genetic Algorithms", Universitat Erlangen-Nürnberg, 1996.