# Introducing Autonomy Oriented Computation

Jiming Liu
Dept. of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
jiming@comp.hkbu.edu.hk

Kwok Ching Tsui
Dept. of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
tsuikc@comp.hkbu.edu.hk

Jianbing Wu
Intelligent Systems Group
University of Calgary
Calgary, Canada T2N 1N4
jbwu@enme.ucalgary.ca

## ABSTRACT
Examples of autonomous multi-entity systems are plentiful, both in the natural and artificial worlds. Many systems have been studied in depth and some models of these have been built in computational systems for problem solving. Central to these computational systems is the notion of autonomy. This article introduces the concept of *autonomy oriented computation* - a bottom-up problem solving paradigm in which autonomy is the basis of modeling. It aims to model, explain and predict behavior(s) in a complex adaptive system.

## 1. INTRODUCTION
Traffic on motorways is notoriously busy but most drivers seem to have learned the type of skill to avoid almost all kinds of collision, with only few exceptions. It is almost equally surprising in a busy market where traffic consists of a mixture of vehicles, ranging from fast-moving trucks to slow-moving push-carts, from bicycles to pedestrians. Everybody, be it drivers or people on foot, has developed the mode of travel that they can co-exist harmoniously while moving in different directions.

Brokers in stock markets seem also to have developed a highly sophisticated 'herding' behavior to buy and sell in the wake of market information. The balance between species of lifeform in an ecosystem is equally complex and yet all of them seems to be settled into a dynamic equilibrium, most of the times. These scenarios point to a common phenomenon that can be observed in everyday life - lots of independent minds can sometimes maintain order in a global view despite the lack of communication, central planning or prior arrangement.

In contrast, team sports players such as basketball players spent many hours practicing team formations before they make 'magical' passes. Without such intensive practice, players will not be able to get the cue from others and defeat is imminent. Even when a team has been playing together for a long time, secret signs have to be given before the desired results can be achieved. In the presence of a group of independent minds, team sports must be significantly different from motorway traffic so that different behavior results.

The World is full of autonomous individuals, be it human, plants, animals or physical objects. Depending on the level where observation is made, these islands, populations or societies, may consist of single individual or many similar individuals. Human being is a good example. A population of human being consists of many individuals and each individual is composed of many biological subsystems and so forth. Some of these multi-entity systems tend to remain in a dynamically stable state, capable of adapting to changes in the environment, e.g. an ecosystem in a swamp and traffic on a highway. Others seem to be chaotic without any order, e.g. the stock market, which seems to be unpredictable. No matter at what level we look at a multi-entity system, lots of activities are happening inside them, both between and within the members of the system.

### 1.1 Characteristics of Autonomy
These multi-entity systems are sometimes called complex adaptive systems (CAS) and they have the following characteristics:

1. **Autonomous**: the systems' elements are rational individuals that will act independently. In other words, a central controller for directing and coordinating individual elements is absent;
2. **Emergent**: They exhibit, often not simple, behaviors, that are not present or pre-defined in the behavior of the autonomous entities within the CAS;
3. **Adaptive**: They often change their behavior in response to changes in the environment in which they are situated; and
4. **Self-organized**: They are able to organize the elements to achieve the above behaviors.

### 1.2 Autonomy Defined
According to the American Heritage Dictionary of the English Language, *autonomy* is defined as (1) the condition or quality of being autonomous; independence, (2) self-government or the right of self-government; self-determination and self-directed. All these relates to freedom from the control by others with respect to local or internal affairs. In the

context of Artificial Intelligence, autonomy has been one of the key elements in many research fields such as Intelligent Agents and Artificial Life [30, 37], to name but a few. A similar definition can be stated as follows.

Definition 1. *Autonomy is a self-governed, self-determined and self-directed entity with respect to its own action and internal state, free from the* ***explicit*** *control of another entity.*

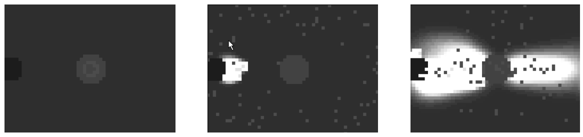
This is an endogenous view of an autonomy. In other words, the internal affair of an entity is protected from the influence of others in the way similar to that of an object in the Software Engineering sense. It is also worth noting that only direct perturbation is prohibited; indirect influence is allowed and even encouraged (to be discussed below). The underlying assumption is that all entities are able to make rational decisions, subject to the limitation of the available information.

In terms of exogenous behavior, autonomy can exhibits one of three kinds of behaviors: purposeful behavior, emergent behavior, emergent purposeful behavior. By purposeful we mean an entity acts according to a goal and the resultant behavior. Such behavior can be defined with the attributes and capabilities of the entity concerned. In contrast, emergent behavior is some system behavior not inherent in the individual entities of a CAS. It can only result from the interaction between individual entities. However, it is not a collective behavior as it does not involve sharing of power to accomplish a goal. The most complex of all exogenous behavior of autonomy is emergent purposeful behavior, which is a combination of the former two behaviors. In other words, emergent purposeful behavior is a goal-directed emergent behavior that is not achievable at the individual entity level.

## 1.3 Autonomy in Natural Systems

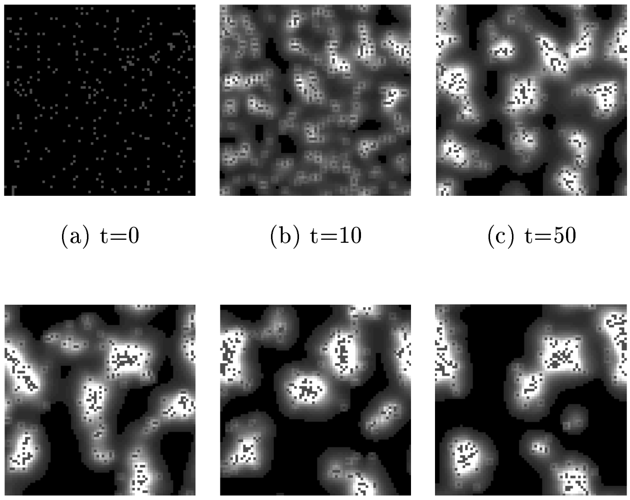
Food foraging in an ant colony, for example, is an individual task as well as a group task [19]. Ants start off with some kind of random walk in the absence of any information about food source,. Once a food source is found, an ant will lay some pheromone along the path between the food source and the nest as a signal to other ants. When other ants in the locality of the path, while looking for food themselves, found a higher concentration of pheromone along a certain direction, it may decide (with high likelihood) to follow the path. Once food is found, the ants will add more pheromone to the path. Therefore, more ants are 'recruited' until the food source is exhausted and the pheromone evaporated. This is a purposeful behavior.

Slime mold is a single cell organism that behaves like an amoeba as an individual. However, this is true only when the condition is favorable and food is plentiful. When food becomes scarce, slime mold cells will gather around to form a cluster and stop reproducing. They started to behave like a single multicellular organism until food, for example, is found. They then will split like a flower stalk bearing a ball of spores. The spores, and hence the slime-mold cells, will eventually separate from the crowd and (re-)live as individuals. This is a good example of emergent purposeful



(a) (b) (c)

**Figure 1: Ant food foraging simulation with pheromone laying as recruitment strategy. (a) Ants start to explore the environment with random walk from colony (center); (b) First few ants discovered the food source (left) and started the lay the pheromone trail (white); (c) A phermome trail fully established between food source and colony and significantly more ants (grey squares) started to gather around the trail**



(a) t=0 (b) t=10 (c) t=50

(d) t=150 (e) t=300 (f) t=600

**Figure 2: Slime-mold cells cluster together due to unfavorable conditions**

behavior. Similar to the pheromone laying behavior of the ants, slime-molds cells will produce a chemical called cAMP and emit it into the environment. They are also attracted by the same chemical. Therefore, in the absence of any vision, slime-mold cells can find each other by simply following the cAMP concentration gradient and remain at the place where cAMP concentration is the highest. This process is purely local to the cells as they cannot sense anything far away.

## 1.4 Man-Made Systems

Mathematical modeling has been used successfully in many areas. However, it requires detailed understanding of the system to be modeled, which often is missing, before ordinary differential equations can be formulated. By way of examples, the benefit of a bottom-up approach rather than a top-down approach is illustrated.

### *1.4.1 Transportation System*

Traffic jams are commonly found in metropolitan and have been studied using many methods. Transportation systems involve two major parties: Drivers who exhibit various kinds of social behavior and physical systems such as cars, traffic lights, and road that draws the boundaries on the limits of a transportation system. Any work attempting to study the dynamics of a transportation system needs to consider the effects that these parties can exert on it. The result of such studies is a better understanding of the transportation system so that the cause of certain phenomenon, such as traffic jam, can be explained and the effect of plans, such as the addition of extra set of traffic lights, can be predicted.

Traditional methods of predicting traffic patterns are based on statistical models that treat the traffic system as a homogeneous fluid. These methods produces predictions of traffic patterns at the global level without considering the difference in behavior of individual drivers - only an average driver is represented. Researchers at Los Alamos National Laboratory devised a system called the Transportation Analysis Simulation System (TRANSIMS) [48] to offer an alternative to the traditional methods. TRANSIMS is essentially a cellular automata. Cars are represented by cells in system and they change state, such as start or stop, according to its current state and that of its neighbors. The system has successfully mimicked the actual observed traffic patterns of Albuquerque, NM, USA and some large metropolitan [28].

Helbing and Huberman have also applied a similar simulation technique to study the evolution of various traffic patterns out of disorganized vehicle flow by competitive interactions [23]. Specifically, they reported studies of traffic involving a diverse set of vehicles, such as cars and lorries, along a two-lane highway with different velocities. Vehicles are characterized by different desired velocities, which affect the way drivers behave in the presence of traffic on the highway. By including simple behavior rules such as follow-the-leader and lane-changing, different phenomena such as traffic jam and coherent solid mass can be observed and agreed with empirical data for Dutch traffic.

### *1.4.2 Flocking of Birds*

Flocks of birds and schools of fish do not have a leader to decide where to go but yet are able to fly or swim as a 'mass' as if they have planned their route already. The modeling of such behavior clearly cannot be performed by a top-down approach. Instead, Reynolds has simulated perfectly these behavior in his boid flock model [50]. Each individual, or boid, in the flock is characterized by three simple steering behaviors that govern the boid movement. These steering behavior are based on the positions and velocities of a boid's nearby flockmates, they are:

- *Separation*: steer to avoid crowding local flockmates
- *Alignment*: steer towards the average heading of local flockmates
- *Cohesion*: steer to move toward the average position of local flockmates

Neighborhood is defined by distance and the angle from the direction of flight. It limits the 'visibility' of the boids in a way similar to the murky waters obscure the vision of fishes beyond a certain distance. Advanced features such as predictive obstacle avoidance and goal seeking are added to simulate some flock behaviors. The boid flocking model has been used in producing animation in films. Similar idea has been used to visualize information cluster [45].

### *1.4.3 Financial Market*

Financial market has long been recognized as dynamic and unpredictable with many fluctuations. A lot of its behaviors are against the predictions of traditional economic theories that assume a perfect free market controlled only by supply and demand and are based on equilibrium and efficiency. Ormerod has pointed out some examples such as the success of technologically inferior VHS video system, an inefficient QWERTY keyboard, and low budget films that are supported by infamous casts [42]. The success of these products cannot be explained by common economic theories. However, in light of the ant colony analogy, where individuals' decisions to buy are affected in part by others who have or have not bought. The effect of positive feedback magnifies to an extent that drives other competitors bankrupt. With the assumptions that the role of financial markets is to allocate resource, Farmer has likened it to a biological ecology and has studied analytically few common trading strategies [16]. For example, the following analogies are drawn: species and trading strategy, individual organism and trader, mutation and recombination and creation of new strategies, among others. These analogies are helpful for understanding and explaining market boom-bust cycles due to different strategies under given conditions. Rieck has also conducted empirical study on the evolutionary simulation of asset trading strategies [52].

## 1.5 Organization of the Article

Many models have been built using some analogy from nature and computational systems for problem solving have been formulated with them. The aim of this article is to survey the application of autonomoy as the basis of system modeling and we call this *Autonomoy Oriented Computation* (AOC).

The article is structured as follows. The concept of autonomy oriented computation (AOC) is then discussed where details of three major approaches of AOC and their examples are given. Alongside these are the principles, typical applications, and simulation evironments related to AOC (section 2). Some research issues, both theoretical and practical, are also described (section 3). The article concludes with a summary of the main pioints.

# 2. AUTONOMY ORIENTED COMPUTATION

Autonomy oriented computation (AOC) refers to computation algorithms that employ autonomy as the core model of any complex system. They aim at modeling, explaining and predicting the behavior of such systems, which are hard to model using top-down approaches. Local interaction between the autonomous entities is the primary driving force of AOC. An abstracted version of some natural phenomenon is the starting point of AOC so that the problem at hand can be recasted. There are three different approaches of AOC and are applicable in different circumstances. This section describes these methods in detail alongside some examples.

## 2.1 Scope of AOC

Formulation of an autonomy oriented computational system involves an appropriate analogy, which normally comes from nature. Employing such an analogy, therefore, requires identification, abstraction and reproduction of certain natural phenomenon. The process of abstraction inevitably involves certain simplification of the natural counterpart. For example, in the commonly used version of genetic algorithm [26] in the family of evolutionary algorithms, the process of sexual evolution is simplified to selection, recombination and mutation, without the explicit identification of male and female, for example, in the gene pool. Genetic programming [31] further modifies the chromosome representation to tree. Evolutionary programming [17] and evolution strategy [55], on the other hand, is closer to asexual reproduction with the addition of constraints on mutation and the introduction of mutation operator evolution respectively. Despite these simplification and modification, evolutionary algorithms capture the essence of natural evolution and are proven global optimization techniques.

DEFINITION 2. *Synthetic Autonomy is defined as an abstracted, desired, observable natural phenomena and is the fundamental building block of an autonomy oriented computational system.*

Any computational system built using synthetic autonomy involves the interaction between them. This is similar to the messages that are passed between objects in an object-oriented program. Both the interaction and the messages are the 'glue' that help put the building blocks together to form a coherent system. The computational system exhibits either emergent behavior or emergent purposeful behavior that is not pre-programmed in the system and its components.

DEFINITION 3. *Emergent Autonomy is defined as the observable, self-induced behavior of an autonomy oriented computational system built using synthetic autonomy as the basic building block.*

It is not difficult to see that a computational system derived from many levels of abstraction and modeling. If a human society were to be modeled as a computational system, abstraction can possibly occurs at the level of: population, individual, biological system, cell, molecule and atom. Note that autonomy according to definition 1 is present at all these levels. Moreover, the emergent autonomy obtained at, say, the cell level, is the foundation for building emergent autonomy at the biological system level. As a result, another definition of autonomy in the context of a computational system using that of synthetic autonomy and emergent autonomy given above can be stated as:

DEFINITION 4. *Autonomy is a self-governed, self-determined and self-directed emergent autonomy with respect to its own action and internal state, built from synthetic autonomy, and is the basic synthetic autonomy for producing emergent autonomy, free from the explicit control of another autonomy.*

This multi-level view of an autonomy encompasses the subsumption architecture [5] in that complex behavior can be built up from multiple levels of simpler, and relatively more primitive, behavior.

## 2.2 Approaches of AOC

AOC algorithms can be divided into three main approaches. The first approach – AOC-by-fabrication – aims at replicating certain self-organized collective behavior observable in the real world. The operating mechanism is more or less known and may be simplified during the modeling process. Research in the Artificial Life community falls well into this approach. The natural application of this type of AOC algorithms is as a problem solving technique, such as optimization algorithm. Nature-inspired techniques such as Genetic Algorithm and Ant system are typical examples of such extension.

The second approach – AOC-by-prototyping – aims at understanding the operating mechanism underlying the system to be modeled by simulating the observed phenomenon, through modeling a society of autonomous entities. Examples include the study of Internet ecology, traffic jams and web log analysis. This approach of AOC algorithm relates to multiagent approach to complex system in Distributed Artificial Intelligence.

The third approach – AOC-by-self-discovery – aims at the discovery of a solution by employing autonomy. The trial-and-error process of the AOC-by-prototyping AOC is replaced by autonomy in the system. In other words, the distance measure between the desired emergent behavior and the current emergent behavior of the system in question becomes part of the environmental information used by the autonomy. Some evolutionary algorithms that exhibit self-adaptive capability are examples of this approach.

AOC algorithms share a basic form with many possible variants. The following sections attempt to characterize these three classes of AOC algorithms, where more detailed illustrations are given for each case.

### *2.2.1 AOC-by-Fabrication*

Works in the field of Artificial Life (ALife) fall into the AOC-by-fabrication approach of AOC. AOC-by-fabrication is characterized by some known working mechanism of the natural phenomenon to be abstracted. The definitions given by the inventor of ALife on two separate occasions are:

"The study of man-made systems that exhibit behaviors characteristic of natural living systems." [32]

"Artificial Life is a field of study devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomena, and recreating these dynamics in other physical media – such as computers – making them accessible to new kinds of experimental manipulation and testing. ...In addition to providing new ways to study the biological phenomena associated with life here on Earth, *life-as-we-know-it*, Artificial Life allows us to extend our studies to the larger domain of "bio-logic" of possible life, *life-as-it-could-be* ..." [33]

Some well known instances of ALife include visual arts [56], L-System [46, 47] and Tierra [49]. These systems aims at the replication of some natural behaviors.

Another strand of AOC-by-fabrication goes a bit further beyond behavior replication. Genetic algorithms [26], for example, simulate natural evolution in an artificial setting and apply the algorithm to global optimization tasks. Another successful algorithm that has been applied in similar domains is the Ant System [14]. It mimics the food foraging behavior of ants [19]. Other examples include the Selfish Gene Algorithm [8] and the Culture Algorithm [51] simulates the Selfish Gene principle [12] and the development of culture in a society respectively. Application of AOC-by-fabrication to image segmentation and feature extraction is described below. Another example on solving the N-queen constraint satisfaction problem can be found elsewhere [21].

#### *Image Segmentation and Feature Extraction*

Image segmentation requires identification of a homogeneous region within the image. However, homogeneity can be at varying degree at different parts of the image. This presents problems to conventional methods such as split-and-merge that segments the image by iterative partitioning of non-homogeneous regions and simultaneous merging of homogeneous ones [43, 44]. An autonomy oriented approach has been used to tackle the same task [35]. Autonomous entities are deployed to the 2-D representation of the image. Each entity is equipped with the ability to assess the homogeneity of the region within a predefined locality. Specifically, homogeneity is defined by the relative contrast, regional mean and region standard deviation of the grey-level intensity. When a non-homogeneous region is found, the autonomous entity will diffuse to another pixel in a certain direction within the local region. In contrast, when an entity locates a homogeneous region within the range of the pixel it presently resides, it replicates (breeds) itself to give rise to certain number of offspring and deliver them to its local region in a certain direction.

The breeding behavior enables the newly created offspring to be distributed near the pixels where the region is found to be homogeneous, so that it is more likely to find the extension to the current homogeneous region. Apart from breeding, the entity will also label the pixel found to be homogeneous. If an autonomous entity fails to find a homogeneous region during its life span (a predefined number of steps) or wandered off the search space during diffusion, it will be marked inactive.

In summary, the stimulus from the pixels will direct the autonomous entities to two different behavioral tracts: breeding and pixel labeling, or diffusion and decay. The directions of breeding and diffusion is determined by their respective behavioral vector, which contains weights (between 0 and 1) of all possible directions. The weights are updated by considering the number of successful siblings in the respective directions. An entity is considered to be successful if it has found one or more pixels that are within a homogeneous region during its life time. This method of direction selection is somewhat similar to herding behavior that considers only local information.

Similar technique has also been applied to a feature extraction tasks such as border-tracing and edge detection [34]. A more difficult task that has three grey levels within an image has also been solved by deploying three different types of autonomous entities that have different homogeneity criteria.

#### *Summary*

The AOC-by-fabrication examples have the following common characteristics:

1. There is a population of homogeneous individuals, each is characterized by a set of behavior rules, and individuals differ only by the parameters of the behavior rules;

2. The composition of the population changes over time, by the process analogous to birth (amplification of the desired behavior) and death (elimination of the undesired behavior);

3. Interaction between individuals is local, no global information nor central executive to control the behavior selection is needed;

4. Behavior selection is probabilistic;

5. The environment is dynamic and acts as the center of information relating to the current status of the problem and as place holder for information sharing between individuals;

6. Local goal of individual drives the selection of local behavior;

7. Global goal is represented by a universal fitness function which gives a measure of the progress of the computation.

### *2.2.2 AOC-by-Prototyping*

With the help of a blueprint, engineers can build a model of a system in an orderly fashion. This is true for AOC-by-fabrication in the context of building a computational system using autonomy. When insufficient knowledge about the mechanism how the system works, it is difficult, if not impossible, to build such a model. Assumptions about the unknown workings have to be made in order to get the process started. Given some observable behavior of the desired system, designers can verify the model by comparing that of the model with the desired features. This process will have to be repeated several times before a good, probably not perfect, prototype is found. This is AOC-by-prototyping in the context of building autonomy oriented computational system. Apart from obtaining a working model of the desired system, an important by-product of the process is the discovery of the mechanisms that are unknown when the design process first started. This view is shared by researchers in the field of simulating societies that aims to develop and test ideas and theories about society and social phenomena [7, 13]. Examples of the AOC approach includes crowd control [22, 59] and regularities in web surfing behavior [36].

*Web Surfing Regularities*
The Internet is growing in size, i.e. number of web pages, everyday. At the same time, more and more people are getting 'connected'. A good understanding of the browsing behavior of all web surfers has many important implications to network designers and portal engineers. Liu and Zhang [36] studied the web browsing behavior of users using both synthetic data and real data obtained from Microsoft Corporate Web site.

The web was simulated by a graph where nodes represent web pages and edges represent hyperlinks connecting the web pages. The graph is generated by assigning topics to each web page according to a normal distribution. The variance of the distribution therefore controls how similar all the pages are. As users are the main subject to be studied, each of them are simulated in the system by associating with them an interest vector, again generated randomly based on a normal distribution with certain variance. Therefore, there is a parameter to control the degree of overlap in interest between users.

Each user's navigation behavior is characterized by a motivation support, which will go up if the user meets a web page that matches his/her interest; it will go down otherwise. When the motivation support reaches a certain threshold, the user will stop browsing - a situation signify the user has obtained enough information. In more detail, the motivational support is influenced by the support of the past steps, users' patience to navigate in order to find information, and a reward for finding relevant information. Users are further categorized into three different group - recurrent user who is familiar with the web structure, rational user who is new to the web site but knows clearly what he/she is looking for, and random user who has no strong intention to get anything and just 'wander' around.

The above model is validated experimentally using the synthetic data and the real data. Results show that there exists similar power law relationship between surfing depth/step distribution and the number of clicks for a link between two web pages. Using the model described above, critical factors that affect a user experience were analyzed by altering some parameters of the model. It was revealed that there exist an optimal accessibility (distance) between two web pages.

*Summary*
AOC-by-prototyping can be seen as an iterated application of AOC-by-fabrication with the addition of parameter tuning at each iteration. The difference between the desired behavior and the actual behavior of a prototype then becomes the guideline to parameter adjustment. The process can be summarized, with reference to the summary of AOC-by-fabrication in section 2.2.1, as follows:

1. Behavior rules can be changed from one prototype to the next
2. Interaction rules can be changed from one version to the next
3. There is an additional step to compare the desired behavior with the current emergent behavior
4. A new prototype is built by adopting steps 1 and 2 above and repeat the whole process

### 2.2.3 *AOC-by-Self-Discovery*
AOC-by-self-discovery emphasizes the ability of an AOC to find its own way to achieve what AOC-by-prototyping can do. The ultimate goal is to have an fully automated algorithm that can adjust its own parameters for different application domain. In other words, the AOC itself becomes an autonomy. Examples include adaptive evoltionary algorithms [1, 2, 18, 20, 24] and the evolution of emergent behavior [10, 11, 27]. The latter is described in detail below to illustrates the characteristics of this approach to AOC.

*Evolution of Emergent Computation*
An example of AOC-by-self-discovery is the evolution of emergent computation [10, 11, 27]. A one dimensional binary cellular automata of $N$ cells was used as the basic information processing engine. Each cell is a two-state machine that can change only between 1 or 0. It has visibility of the state information of neighbors within a certain radius (3 in the reported work). The task is a synchronization task that requires the cellular automata to reach a configuration such that all cells are able to oscillates between all 0s and 1s on successive time steps. This is global information processing in the absence of a centralized coordination as no cell in the cellular automata has knowledge of what state all other cells are in. State update rules are implemented as a lookup table that encodes all the combination of the states a cell, states of its neighbors within its visibility range and this cell's next state. If a lookup table that can perform the required task is crafted by hand, this will be an AOC-by-fabrication example. The work described here went a step further by allowing a genetic algorithm to evolve the lookup table. An initial population of 100 randomly generated rules were used to evolve the optimal rule. It was found that the resultant rule is able to build up blocks of cells that are in the same state and start to expand. It can be understood as propagation of a flip message along the block boundaries until the whole CA reaches the goal state. The same combination of genetic algorithm and cellular automata has been applied to generate test patterns for hardware circuit [6]. A variant of this method is to use the selfish gene algorithm in place of the genetic algorithm [9].

*Summary*
Full automation of the prototyping process is achieved by having an autonomy to model another level of autonomies. The above examples show that AOC-by-self-discovery is indeed a viable proposition. The steps for engineering the two levels of autonomy are the same as those stated in section 2.2.1. While the two levels of autonomy share the same goal (or fitness function), they differ in the actual object to be optimized. For the higher (meta-) level autonomy, the target object is the parameter set of the lower level autonomy. On the other hand, the goal of the lower level autonomy is task oriented.

### 2.2.4 *Related Work*
Autonomy oriented computation certainly shares a lot of commonality with other research areas. This section at-

tempts to draw attentions to some clear distinctions between them.

*Artificial Life* emphasizes on the simulation of life in a computer setting. It, therefore, falls short of its use as a computation method for problem solving. On the other hand, an AOC does not necessarily need to produce lifelike behavior as natural phenomena are usually abstracted and simplified. *Agent Based Simulation* (ABS) shares similar goal with AOC-by-prototyping - finding explanation to observed phenomena. Again, there is no computation problem to be solved in ABS.

There are at least three examples of multi-entity computational system. *Multi-agent Systems* for distributed decision making [15, 54] is one example. Works in this field attempt to solve a computation task by delegating responsibilities to groups of agents. These agents usually are *heterogeneous* entities and different groups have different roles. For example, in the Zeus collaborative agent building framework, agents are divided into utility agents, such as name server, facilitators and visualizer, and domain-level agents [41]. In relation to this, behavior of individual is preprogrammed and can sometimes be complex. Complicated issues such as negotiation and coordination are paramount importance. All these are usually part of the system design and hence require a lot of human intervention.

Another example of multi-entity computation is *Ecology of Computation* [25, 29] where a population of heterogeneous computational system shares partial solutions to the common problem at hand. These individual problem solvers tackle the problem with different methods and, therefore, have different internal representation of the problem and/or solution. While it is shown that this approach is efficient in solving a problem, the coordination needs to be articulated carefully so that different internal representations can be translated properly.

*Distributed Constraint Satisfaction Problem* (CSP) [61, 62, 63] was proposed to solve CSP problems in a distributed manner. Specifically, the asynchronous backtracking algorithm assigns a variable to an agent and a directed graph called a constraint network is constructed to represent the relationships between the variables. The asynchronous weak commitment search algorithm enhanced the above algorithm by adding a dynamic priority hierarchy. In the event of conflicts, the hierarchy structure is changed so that a sub-optimal solution can be constructed first and then the final solution incrementally. This is different from the AOC approach where competition and behavior amplification is used to discover the best strategy for an autonomous entity. Furthermore, AOC does not need this hierarchy to find a solution as all autonomous entities are treated equally and only local neighbors (might change over time) matter in checking conflicts. While the distributed CSP algorithms are event-driven (act upon the receipt of messages from other agents), AOC is clock-driven with synchronous update.

*Swarm Intelligence* [3, 4] is perhaps the most similar to AOC-by-fabrication in that it emphasizes the social insect metaphor for problem solving through local interaction of relatively simple entities. It has not yet been applied to areas that other AOC approaches have been used. Extending swarm intelligence to explaining complex behavior and discovery of problem solvers should be studied.

While the three approaches of autonomy oriented computation varies in their goals, they represent a natural progression from detailed engineering work to an automated process. The common features can be summarized by the following principles for the benefit of those who wants to develop new algorithms using the autonomy oriented approach.

**Homogeneity** Self-similar individuals that have similar behavior rules are the main component of a complex adaptive system. Their rule-set may only differ in the parameters of the rules but not the structure of the rules.

**Simplicity** Behavior model of each individual in the population is simple.

**Locality** Interaction is strictly local although the notion of locality can be physical as well as logical.

**Implicity** Another form of interaction comes from the implicit knowledge sharing between individual via a common environment. This environment can also have dynamic behaviors such as spreading the shared news or aging the news contributed by the individuals.

**Uncertainty** Behavior such as rule selection is not deterministic. There is always a certain degree of randomness and uncertainty in the decision-making process.

**Amplification** Desirable behavior is amplified while undesirable ones are eliminated via mechanisms such as birth and death. This is also the result of positive feedback from the environment.

**Recursion** Complex behaviors are aggregated from ineractions of some simple autonomy after many iterations. The result of each iteration is an emergent autonomy, which can become a synthetic autonomy (basic building block) of another emergent autonomy.

**Scalability** Cutting across all the AOC approaches is the issue of scalability. AOC seeks to take a bold view of scalability requirements, as it envisions an environment able to scale up or down according to the ever changing needs of a dynamic complex system.

**Openness** New types of individuls/entities can be accommodated seamlessly into the complex systems.

Radomness is an important factor in any self-organising system such as those formulated using the AOC principles. On

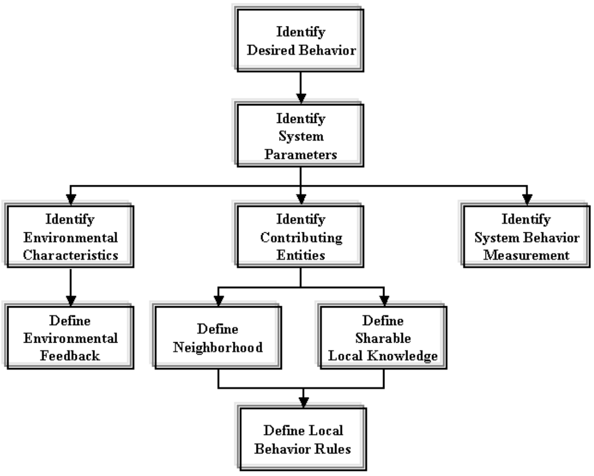



**Figure 3: Block diagram of major components of an AOC that need to be defined**

one hand, it helps the autonomous entities in the system to explore new areas in the solution space. On the other hand, it introduced certain degree of uncertainty regarding the outcome of the simulations. Therefore, it is important to have a concrete assessment of the progress in an AOC system.

## 2.3 Applications

A handful of examples on the applications of autonomy oriented computation are described in this article. The AOC-by-fabrication examples fall into the area of global optimisation, constraint satisfaction problem and search. These problems have several commonalities. Firstly, there are a collection of homogeneous entities. If AOC-by-fabrication were to be used to solve a particular problem, the right level of abstraction has to be chosen so that homogeneous entities can be identified. This does not, however, exclude the repeated application of this technique to suit the need of any application that can be best modeled by multiple levels of abstraction. Secondly, there is some relationships between the homogeneous entities, in the form of constraints, such as limitations on the location inhabitable by a queen in an N-queen problem [21]. Thirdly, a performance measurement is available to assess the quality of any solution. It can be used in the AOC as a driving force to direct the autonomous entities.

There are two basic stages of modeling in an AOC algorithm (see Figure 3). The first part is individual behavior modeling and the second stage is system behavior modeling. These two stages are considered on top of the basic system setup such as the number of autonomous entities required and simulation environment parameters. The major tasks at the first stage is the modeling of autonomous behavior of an AOC's basic components, namely the autonomous entities and the environment. This includes defining the capabilities of an autonomous entity and the characteristics of the environment in relation to the problem at hand. The next thing to define is the neighborhood measurement and the kind and form of information to be shared between neighbors. The second stage relates to problem-solving and the main items to define is the distance measurement between the desired behavoir and the system behavior produced by the current set of individual behavior rules. The end of the second stage will tirgger the next cycle which involves behavior rule modification.

# 3. RESEARCH ISSUES

Research in Autonomy Oriented Computation has been active in the past decade or so with many successful applications. However, many areas need to be explored in order to exploit the benefits of AOC. This sections sketches some directions, both theoretical and practical, of future research.

## 3.1 Theoretical Issues

### *3.1.1 Concept, Methodology & Theory*

New approaches to system dynamics and performance measurement are particularly needed so that clearer guidelines can be developed to help practitioners to gain better insight into the AOC, and AOC-by-self-discovery in particular. Measure of emergence, evolvability and self-organization, tractability and scalability of AOC are useful for tracking the progress of an AOC.

Some pioneer work in these areas has already started to appear. Wright et al. has suggested a measure of emergence based on information theory [60]. Ronald et al. has also defined a qualitative emergence test based on a surprise factor obtained from two viewpoints of the an emergent behavior [53]. Standish has studied the relationship between the size all equivalent descriptions needed to encode the complexity mesure of a complex system [58]. While Nehaniv has studied a hierarchical complexity measure for a biological system [40]. In a separate study, Nahaniv has defined evolvability as the rate of complexity [39].

Theory on the formation of roles and social structure in a community of autonomies would expand the capability of a homogeneous autonomies to heterogeneous ones.

### *3.1.2 Algorithm Comparison*

Strength and weaknesses of AOC need to be assessed formally by comparing with other multi-agent paradigms to establish a clear insight into the benefits of AOC. Benchmark problems should also be identified for this purpose.

## 3.2 Practical Issues

There are a few decisions need to be made when implementing an autonomy oriented computation. They include: hardware and software environment, update schedule, management services and visualization. This section discusses the relevancy of these issues to the three approaches of AOC.

### *3.2.1 Hardware and Software Environment*

Autonomy oriented computation usually involves more than one object in the system. These objects include the autonomy and the dynamic environment. Implementing the computational system in a single processor machine requires the support of virtual parallelism. Modern operating systems and programming languages support multi-threading. This allows slicing up CPU cycles and allocate them to the individual processes that represent objects in the system. When multiple processors are available, individuals can be allocated to different processor on the same machine or across the network of processors with the support of some facilities such as Parallel Virtual Machine[1] (PVM) and Message Passing Interface (MPI) [2]. However, it requires a central control program to coordinate task allocation and result consolidation. This makes parallel implementation of, for example, genetic algorithm possible without requiring expensive parallel machine.

With the popularity of network programming, mobile agents can be sent to run on any machine over the Internet, provided permission is granted by the host. Running simulated evolution in this way has been attempted [57] and is a good starting for those pursuing this line of research.

### *3.2.2 Update Schedule*

Individual entity changes state at every step based on its current state and that of its neighbors. In a synchronous update scenario, the current state of all individuals are frozen to allow them to obtain state information and change state, if appropriate. The current state of the whole system is then updated and the system clock ticks, marking the beginning of the next time step. In a parallel implementation, synchronization might become an overhead too big to handle. Asynchronous update can therefore allow processes on each processor to proceed as if it were a single processor machine. However, the choice of update schedule and the choice of hardware platform is related. If a multi-process hardware environment is chosen, synchronous update would slow the simulation down as all processes have to start and stop at the same time.

### *3.2.3 Management Services*

With the vast amount of autonomous entities in the system, autonomy oriented computation needs to keep track of the creation and deletion of objects. Moreover, a centralized messaging mechanism is needed to facilitate message passing between object. A central clock is also required to help the autonomous entities to synchronize with state update, no matter it is synchronous or asynchronous. Some centralized whiteboard may also be needed if the autonomous entities were to share information in an implicit way and to contain a global view of the system's status. This whiteboard may also be used to simulate the dynamic environment in systems such as the Ant system [14].

### *3.2.4 Visualization*

Visualization is a good way for people running simulations to 'see' what is going on with the experiment. Items of interest that are related to individual autonomy include actual movement, state, actions taken, fitness, age, etc. On the other hand, some global information such as population size, best fitness, average fitness, etc, alongside any progress measurement such as measure of emergence, evolvability, diversity and convergence, are also of interest to modelers. Visual display of all these information is of tremendous help to modelers to obtain a quick view of the system.

## 4. SUMMARY

We have outlined in the article a new computational method called Autonomy Oriented Computation based on autonomy and local interaction between autonomies. The usage of AOC is vast as judged from many examples described above. AOC has three approaches with different objectives. AOC-by-fabrication is similar to construction with a blueprint where known phenomena is abstracted and replicated as a computational problem solving technique. AOC-by-prototyping presents a trial-and-error approach to find explanations to some observations via an autonomy oriented system. Human involvement is certain intensive to fine-tune system parameters. AOC-by-self-discovery, on the other hand, is an autonomous problem solver that can tune its own settings to suit the problem at hand. It requires less human intervention than AOC-by-prototyping but represents the highest degree of uncertainty as not until a solution is found, it is difficult to tell when it will stop.

This article has also discussed issues related to implementation of an AOC algorithm and has compared AOC with some related work. There are still many issues to be addressed by the research community, some of which have also been described. This article has just scratched the tip of an iceberg and hope to stimulate more interests in this excitingly new field.

## 5. REFERENCES


[1] A. J. Angeline. Adaptive and self-adaptive evolutionary computations. In M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, and T. Fukuda, editors, *Computation Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, Piscataway, NJ, 1995.

[2] T. Bäck. Self-adaptation. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages C7.1:1–15. Institute of Physics Publishing and Oxford University Press, 1997.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, New York, NJ, 1999.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406:39–42, 6 July 2000.

[5] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[6] S. Chiusano, F. Corno, P. Prinetto, and M. S. Reorda. Cellular automata for sequential test pattern generation. In *Proceedings of IEEE VLSI Test Symposium*, pages 60–65, April, 1997.

[7] R. Conte and N. Gilbert. Introduction: Computer simulation for social theory. In N. Gilbert and R. Conte, editors, *Artificial Societies*, chapter 1, pages 1–15. UCL Press, London, 1995.

[8] F. Corno, M. S. Reorda, and G. Squillero. The selfish gene algorithm: A new evolutionary optimization strategy. In *Proceedings of SAC'98: 13th Annual ACM Symposium on Applied Computing*, pages 349–355, February 1998.


[1] website: www.epm.ornl.gov/pvm/

[2] website: www.erc.msstate.edu/labs/hpcl/projects/mpi/

[9] F. Corno, M. S. Reorda, and G. Squillero. Exploiting the selfish gene algorithm for evolving hardware cellular automata. In *Proceedings of Congress of Evolutionary Computation (CEC2000)*, pages 1401–1406, 2000.

[10] J. P. Crutchfield and M. Mitchell. The evolution of emergent computation. *Proceedings of the National Academy of Sciences, USA*, 92(23):10742–10746, 1995.

[11] R. Das, J. P. C. abd Melanie Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995.

[12] R. Dawkins. *The Selfish Gene*. Oxford University Press, 1989.

[13] J. Doran and N. Gilbert. Simulating societies: An introduction. In N. Gilbert and J. Doran, editors, *Simulating Societies: The Computer Simulation of Social Phenomena*, chapter 1, pages 1–18. UCL Press, London, 1994.

[14] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):1–13, 1996.

[15] E. H. Durfee. Distributed problem solving and planning. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press, 1999.

[16] J. D. Farmer. Market force, ecology, and evolution. Working Paper 98-12-117E, Santa Fe Institute, 14 February 2000.

[17] L. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulation Evolution*. Wiley, NY, 1966.

[18] B. Freisleben. Metaevolutionary approaches. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages C7.2:1–8. Institute of Physics Publishing and Oxford University Press, 1997.

[19] S. Goss, R. Beckers, J. L. Deneubourg, S. Aron, and J. M. Pasteels. How trail laying and trail following can solve foraging problems for ant colonies. In R. N. Hughes, editor, *Behavioural Mechanisms of Food Selection*, volume G 20 of *NATO-ASI Series*. Springer-Verlag, 1990.

[20] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[21] J. Han, J. Liu, and Q. Cai. From ALife agents to a kingdom of N queens. In J. Liu and N. Zhong, editors, *Intelligent Agent Tecnology: Systems, Methodologies andTools*, pages 110–120. World Scientific, 1999.

[22] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamic features of escape panic. *Nature*, 407:487–490, 28 September 2000.

[23] D. Helbing and B. A. Huberman. Coherent movng states in highway traffic. *Nature*, 396:738–740, 24/31 December 1998.

[24] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In T. Bäck, Z. Michalewicz, and X. Yao, editors, *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 65–69, Berlin, 1997. IEEE Press.

[25] T. Hogg and B. A. Huberman. Better than the best: The power of cooperation. In L. NAdel and D. Stein, editors, *SFI 1992 Lectures in Complex Systems*, pages 163–184. Addison-Wesley, 1993.

[26] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, 1992.

[27] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Mechanisms of emergent computation in cellular automata. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V*, pages 613–622, Berlin, 1998. Springer-Verlag.

[28] K. R. Howard. Unjamming traffic with computers. *Scientific American*, October 1997.

[29] B. A. Huberman, editor. *The Ecology of Computation*. North-Holland, Amsterdam, 1988.

[30] N. R. Jennings and M. Wooldridge. Software agents. *IEE Review*, 42(1):17–21, January 1996.

[31] J. R. Koza. *Genetic Programming : on the programming of computers by means of natural selection*. MIT Press, 1992.

[32] C. G. Langton. Artificial life. In C. G. Langton, editor, *Artificial Life*, pages 1–47. Addison-Wesley, Redwood City, CA, 1989. Volume VI of SFI Studies in the Sciences of Complexity.

[33] C. G. Langton. Preface. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages xiii–xviii. Addison-Wesley, Redwood City, CA, 1992. Volume X of SFI Studies in the Sciences of Complexity.

[34] J. Liu and Y. Y. Tang. Adaptive image segmentation with distributed behavior-based agents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):544–551, June 1999.

[35] J. Liu, Y. Y. Tang, and Y. C. Cao. An evolutionary autonomous agents approach to image feature extraction. *IEEE Transaction on Evolutionary Computation*, 1(2):141–158, July 1997.

[36] J. Liu and S. Zhang. Personal communication, 2001.

[37] P. Maes. Modeling adaptive autonomous agents. In C. G. Langton, editor, *Artificial Life: An Overview*, pages 135–162. MIT Press, 1995.

[38] C. Nehaniv, editor. *Proceedings of the Evolvability Workshop at the Seventh International Conference on the Simulation and Synthesis of Living Systems (Artificial Life 7)*, number 351, August 2000. Published as University of Hertfordshire Technical Report 351.

[39] C. L. Nehaniv. Measuring evolvability as the rate of complexity increase. In Nehaniv [38], pages 66–68. Published as University of Hertfordshire Technical Report 351.

[40] C. L. Nehaniv and J. L. Rhodes. The evolution and understanding of hierarchical complexity in biology from an algebraic perspective. *Artificial Life*, 6(1):45–67, 2000.

[41] H. S. Nwana, D. T. Ndumu, and L. C. Lee. ZEUS: An advanced tool-kit for engineering distributed multi-agent systems. In *Proceedings of PAAM98*, pages 377–391, March 1998.

[42] P. Ormerod. *Butterfly Economics*. faber and faber, 1998.

[43] T. Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, 1992.

[44] I. Pitas. *Digital Image Processing Algorithms*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[45] G. Proctor and C. Winter. Information flocking: Data visualisation in virtual worlds using emergent behaviours. In J.-C. Heudin, editor, *Proceedings of 1st International Conference on Virtual Worlds*, pages 168–176, Berlin, 1998. Springer-Verlag.

[46] P. Prusinkiewicz, M. Hammel, and R. Mech. Visual models of morphogensis: A guided tour. Online document, May 1997. at http://www.cpsc.ucalgary.ca/Redirect/bmv/vmm-deluxe/Titlepage.html.

[47] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, NY, 1990.

[48] S. Rasmussen and C. Barrett. Elements of a theory of simulation. Technical Report Working Paper 95-04-040, Santa Fe Institute, 1995.

[49] T. S. Ray. An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 371–408. Addison-Wesley, Redwood City, CA, 1992. Volume X of SFI Studies in the Sciences of Complexity.

[50] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987. SIGHGRAPH '87 Conference Proceedings.

[51] R. G. Reynolds. An introduction to cultural algorithms. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, NJ, 1994.

[52] C. Rieck. Evolutionary simulation of asset trading strategies. In E. Hillebrand and J. Stender, editors, *Many-Agent Simulation and Artificial Life*, pages 112–136. IOS Press, Amsterdam, 1994.

[53] E. M. A. Ronald, M. Sipper, and M. S. Capcarrère. Design, observation, surprise! a test of emergence. *Artificial Life*, 5(3):225–239, 1999.

[54] T. W. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press, 1999.

[55] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley : Chichester, 1981.

[56] K. Sims. Artificial evolution for computer graphics. *Computer Graphics*, 25(4):319–328, 1991.

[57] R. E. Smith and N. Taylor. A framework for 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.

[58] R. K. Standish. On complexity and emergence. Los Alamos Physics Archive arXiv:nlin.AO/0101006 (http://xxx.lanl.gov/abs/nlin/0101006), 2001.

[59] G. K. Still. *Crowd Dynamics*. PhD thesis, Mathematics Department, Warwick University, August 2000.

[60] W. A. Wright, R. E. Smith, M. Danek, and P. Greenway. A measure of emergence in an adapting, multi-agent context. In *Proceedings Supplement of SAB'2000*, pages 20–27, 2000.

[61] M. Yokoo. *Distributed Constraint Satisfaction Foundations of Cooperation in Multi-Agent Systems*. Springer Series on Agent Technology. Springer-Verlag, Berlin, 2001.

[62] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.

[63] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.