Autonomy Oriented Computing (AOC): Formulating Computational Systems with Autonomous Components

Jiming Liu, *Senior Member, IEEE*, Xiaolong Jin, Kwok Ching Tsui, *Member, IEEE*

Abstract

Autonomous multi-entity systems are plentiful in natural and artificial worlds. Many systems have been studied in depth and some models of them have been built as computational systems for problem solving. Central to these computational systems is the notion of *autonomy*. This article surveys research work done along this direction and proposes *autonomy oriented computing* (AOC) as a paradigm to describe systems for solving hard computational problems and for characterizing the behaviors of a complex system. AOC differs from major complex system related studies such as artificial life, simulated evolution, and multi-agent systems in that AOC is not just intended to replicate complex behavior, emulate evolution, or coordinate the functioning of many interacting agents. AOC emphasizes the modeling of autonomy in the entities of a complex system and the *self-organization* of them in achieving a specific goal. Through examining implemented applications, we identify three main approaches to AOC. Specifically, we provide a detailed description of the AOC framework with formal definitions of essential constructs and their interrelationships, including the notions of emergent autonomy, self-organization, and the interactions among entities and environment.

Index Terms

Autonomy Oriented Computing (AOC); Synthetic autonomy; Emergent autonomy; Self-organization; Autonomous entities; Multi-agent systems.

I. Introduction

Nature is full of complex systems, some of which have been extensively studied from different angles and with different objectives [1–8]. Some researchers want to understand the working mechanism of a complex system concerned. Immunologists, for example, want to know the way the human immune system reacts to antigens [3]. Similarly, economists want to know the factors contributing to the ups and downs in share prices [4]. The knowledge gained in this way helps scientists predict future system behavior. Others studying complex system behavior want to simulate the observed complex behavior and formulate problem solving strategies for hard computational problems such as global optimization. Computer scientists and mathematicians have formulated various algorithms based

Jiming Liu, Xiaolong Jin, and Kwok Ching Tsui are with Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong, E-mail: {jiming,jxl,tsuikc}@comp.hkbu.edu.hk.

on natural evolution to solve their problems at hand [5–8]. In general, one wants to be able to *explain*, *predict*, *reconstruct*, and *deploy* a complex system.

Techniques for complex systems modeling can broadly be divided into top-down and bottom-up approaches. Top-down approaches start from the high-level characterization of a system and use various tools, such as ordinary and partial differential equations [9–11]. These approaches generally treat every part of a complex system generically and tend to model average cases well, where behavioral difference of the individuals is minimal and can be ignored [12]. However, this approach is not always applicable. For example, the distribution of antibody in the human immune system tends to be heterogeneous. Therefore, the use of differential equations cannot accurately describe the emergent behavior and dynamics in such biological system [3].

Bottom-up approaches [7], [13–19], on the other hand, start with the smallest and simplest entities of a complex system and model their behavior as follows:

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

This article surveys research work on formulating computational systems based on the notion of autonomy and proposes a bottom-up approach called *autonomy oriented computing* (AOC) as a paradigm to describe systems for solving hard computational problems and for characterizing the behaviors of a complex system. The organization of the paper is schematically shown in Figure 1.

II. Background on Related Approaches

Autonomy oriented computing draws on and further enhances several related research areas. This section highlights some of the key relationships and distinctions between them.

Artificial Life (Alife) emphasizes the simulation of life in a computer setting. It, therefore, falls short of its use as a computational method for problem solving. On the other hand, an AOC does not necessarily need to produce life-like behaviors as natural phenomena are usually abstracted and simplified. Similar to Alife, *Agent Based Simulation* (ABS) is intended to reproduce certain observed complex behaviors in the social context and perhaps find an explanation to the observed phenomena. However, unlike AOC, the behavior model of agents in ABS does not necessarily have a natural/biological analogy. Moreover, there is no computational problem to be solved in ABS.

There are at least three examples of multi-entity computational systems. *Multi-Agent Systems* (MAS) for distributed decision making [20], [21] is one example. Work in this field attempts to solve a computational task by delegating responsibilities to groups of agents.

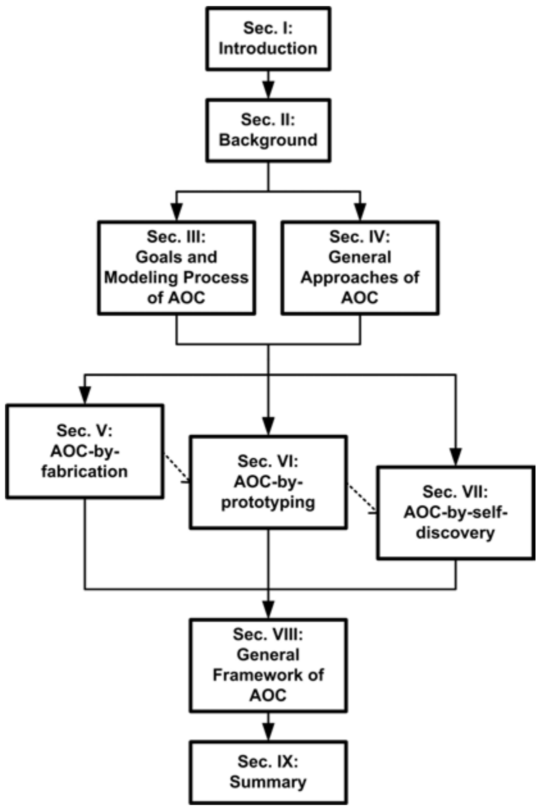

Sec. I: Introduction
Sec. II: Background
Sec. III: Goals and Modeling Process of AOC
Sec. IV: General Approaches of AOC
Sec. V: AOC-by-fabrication
Sec. VI: AOC-by-prototyping
Sec. VII: AOC-by-self-discovery
Sec. VIII: General Framework of AOC
Sec. IX: Summary

Fig. 1. The organization of the paper.

In other words, MAS takes a divide-and-conquer approach to system modeling and the latest work on agent oriented information systems continues with this trend [22], [23]. 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 [24]. The behavior of an individual agent is preprogrammed and can sometimes be complex. Complicated issues such as negotiation and coordination are of paramount importance. These factors are usually part of the systems design and hence require a lot of human intervention.

Another example of multi-entity computation is *Ecology of Computation* [25], [26] where a population of heterogeneous computational systems share partial solutions to the common problem at hand. These individual problem solvers tackle the problem with different methods and, therefore, have different internal representations of the problem/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 (distributed CSP) [27–29] 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 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 an AOC approach where competition and behavior amplification are used to discover the best strategy for an autonomous entity. Furthermore, an AOC approach does not need the above hierarchy to find a solution as all autonomous entities are treated as peers 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 updates.

III. Goals and Modeling Process of Autonomy Oriented Computing

Autonomy oriented computing (AOC) refers to bottom-up computational approaches, which are intended to solve hard computational problems or characterize behaviors of complex systems. In more detail, AOC has three goals: The first goal is *to reproduce life-like behavior in computation*. With detailed knowledge of the underlying mechanism, simplified life-like behavior can be used as model for a general-purpose problem solving technique. Replication of behavior is not the end, but rather the means, of these computational algorithms; the second goal is *to understand the underlying mechanism of a real-world complex system* by hypothesizing and repeated experimentation. The end product of these simulations is a better understanding of or explanations to the real working mechanism of the modeled system; the third goal concerns *the emergence of a problem solver in the absence of human intervention*. In other words, self-adaptive algorithms are desired.

To build an AOC-based model, the following is a list of common steps:

i. observe macroscopic behaviors of a natural system;
ii. design entities with desired synthetic behaviors as well as an environment where entities reside;
iii. observe macroscopic behaviors of the artificial system;
iv. validate the behaviors of the artificial system against the natural counterpart;
v. modify (ii) in view of (iv);
vi. repeat (iii)-(v) until satisfactory;
vii. find out a model/origin of (i) in terms of (ii) or apply the derived model to solve problems.

From the above steps, we note that an AOC system mainly contains a population of autonomous entities and the rest of the system is referred to as the environment. Concentrating on entity and environment, the formulation of an AOC model involves three phases (see Figure 2). The first phase, **natural system identification**, can be viewed as the precursor to actual systems modeling and concerns the selection of an appropriate analogy from the natural and physical world. There are two tasks involved: *identify desired system behaviors* and *identify system parameters*. Choosing the right analogy is the key to the success of the AOC-based system and the right system usually presents itself through its behaviors. Once an appropriate analogy is chosen, details such as the number of entities to run and the length of time to run the simulation need to be decided.

The second phase, **artificial system construction**, involves all elements in the AOC-based system. This phase is divided into two major sub-phases: autonomous entity modeling and environment modeling. The *identify contributing entities* task is the first and the most important task in this phase. Designers are required to choose the level of

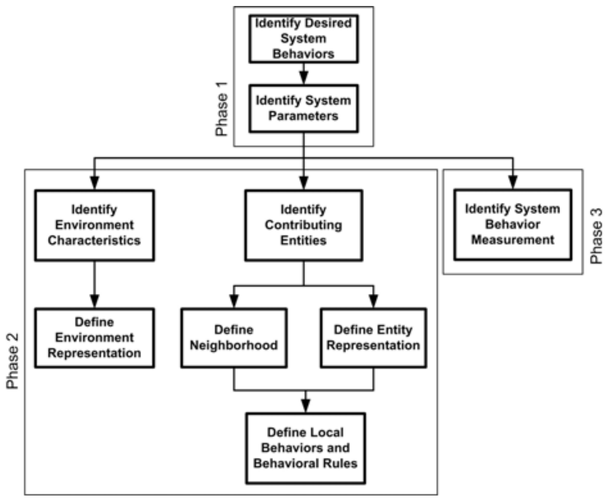

Phase 1
Identify Desired System Behaviors
Identify System Parameters
Phase 2
Identify Environment Characteristics
Identify Contributing Entities
Define Environment Representation
Define Neighborhood
Define Entity Representation
Define Local Behaviors and Behavioral Rules
Phase 3
Identify System Behavior Measurement

Fig. 2. A block diagram of major components of an AOC model.

detail to be modeled, that is appropriate to the problem at hand. The *define neighborhood* task defines a certain measurement (e.g., distance) in the solution space within which local interactions can occur and local information can be collected. The *define entity representation* task handles how to characterize an entity, including its states and goals etc. The last task concerning the entities, *define local behaviors and behavioral rules*, defines the ways in which an autonomous entity reacts to various information it has collected within its neighborhood and the ways in which it adapts its local behaviors and behavioral rules.

The tasks that concern the environment are *identify environment characteristics* and *define environment representation*. The former task concerns the role the environment plays in conveying the knowledge shared between the autonomous entities. The latter task addresses the characterization of the environment.

The third phase, **performance measurement**, concerns the evaluation criteria for comparing the artificial system manifested by the AOC-based system with its natural counterpart. This relates to problem-solving and provides an indication to modify the current set of individual behaviors and behavioral rules. The end of this phase will trigger the next cycle, if necessary, and involves modifications to some or all AOC elements defined in the previous cycle.

IV. General Approaches to Autonomy Oriented Computing

AOC employs autonomy as the core model of any complex system behavior. They are intended to reconstruct, explain, and predict the behavior of such systems, which is hard to model or compute using top-down approaches. Local interactions between the autonomous entities are the primary driving force 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 a certain natural phenomenon. The process of abstraction inevitably involves certain simplification of the natural counterpart. An abstracted version of some natural phenomena is the starting point of AOC such that the problem at hand can be recasted. There are three different approaches to AOC and they are described in more detail, with examples, in the following three sections.

- *AOC-by-fabrication* is intended to replicate certain self-organized behaviors observable in the real-world to form a general-purpose problem solver. The operating mechanism is more or less known and may be simplified during the modeling process. Research in Alife is related to this AOC approach up to the behavior replication stage. Nature-inspired techniques such as Genetic Algorithms and Ant Systems are typical examples of such an extension.
- *AOC-by-prototyping* is devoted to understanding the operating mechanism underlying a complex system to be modeled by simulating the observed behavior, through characterizing a society of autonomous entities. Usually, AOC-by-prototyping involves a trial-and-error process to eliminate the difference between a prototype and its natural counterpart. Examples of this approach include the study of Internet ecology, traffic jams, and Web log analysis. This AOC approach relates to multi-agent approaches to complex systems in Distributed Artificial Intelligence.
- *AOC-by-self-discovery* concentrates on the automatic discovery of a solution. The trial-and-error process of an AOC-by-prototyping algorithm 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 that affects the local behavior of an entity. Some evolutionary algorithms that exhibit self-adaptive capability are examples of this approach.

To put the above three approaches to AOC into a framework of system modeling techniques, we can view AOC-by-fabrication as the basic skill one needs to master. One can then build on these methods to explore, with trial-and-error, the many possibilities of explaining certain observed phenomena. This is exactly what the AOC-by-prototyping methods attempt to accomplish. AOC-by-self-discovery methods, on the other hand, try to eliminate the repeated trial-and-error process that often comes with other AOC approaches by adjusting the system parameters automatically.

V. AOC-BY-FABRICATION

AOC-by-fabrication is characterized by some known working mechanisms of a natural phenomenon to be abstracted and upon which models are built. Work in the field of Alife provides a firm foundation for such endeavor, as the definition of Alife shows.

“The study of man-made systems that exhibit behaviors characteristic of natural living systems.” [13]

“…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. ” [30]

Some well known instances of Alife include visual arts [31], L-System [32], [33], and Tierra [34]. These systems are intended to replicate some natural behaviors.

Building on the experience of systems modeling, AOC algorithms are used for solving hard computational problems. For example, in the commonly used version of genetic algorithm [7] 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 in the gene pool. Genetic programming [35] further modifies the chromosome representation to trees. Evolutionary programming [5] and evolution strategy [6], on the other hand, are closer to asexual reproduction with the addition of constraints on mutation and the introduction of mutation operator evolution respectively. Despite these simplifications and modifications, evolutionary algorithms capture the essence of natural evolution and are proven global optimization techniques.

Another successful algorithm that has been applied in similar domains is the Ant System [8]. It mimics the food foraging behavior of ants [36]. Other examples include the Selfish Gene Algorithm [37] and the Culture Algorithm [38] which simulate the Selfish Gene principle [39] and the development of culture in a society, respectively. The following example is intended to demonstrate the steps in formulating and using AOC-by-fabrication algorithms.

A. Feature Search and Extraction

The specific search problem to be considered in this example is the following: An environment contains a homogeneous region with the same physical feature characteristics. This region is referred to as a goal region. The feature characteristics of the goal can be evaluated based on some measurements. Here, the term *measurement* is taken as a generic notion; the specific quantity that it refers to will depend on the nature of applications. For instance, it may refer to the grey-level intensity of an image in the case of image processing. The task of autonomous entities in the environment is to search the feature locations of the goal region. The entities can recognize and distinguish feature locations, if encountered, and then decide and execute their next step reactive behaviors.

Autonomy Oriented Models in Feature Extraction

In the AOC-based approach, an entity checks its neighboring environments, *i.e.*, small circles as in Figure 3(a), and selects its behavior according to the concentration of elements in the neighboring region. If the concentration is within a certain range, then the current location satisfies a triggering condition. This further activates the reproduction mechanism of the entity.

Taking a border-tracing entity for example, if an entity of border sensitive class reaches a border position, then this entity will inhabit at the border and proceed to reproduce both within its immediate neighboring region and inside a large region, as illustrated in Figures 3(b) and (c).

Image Segmentation

Image segmentation requires identification of homogeneous regions 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

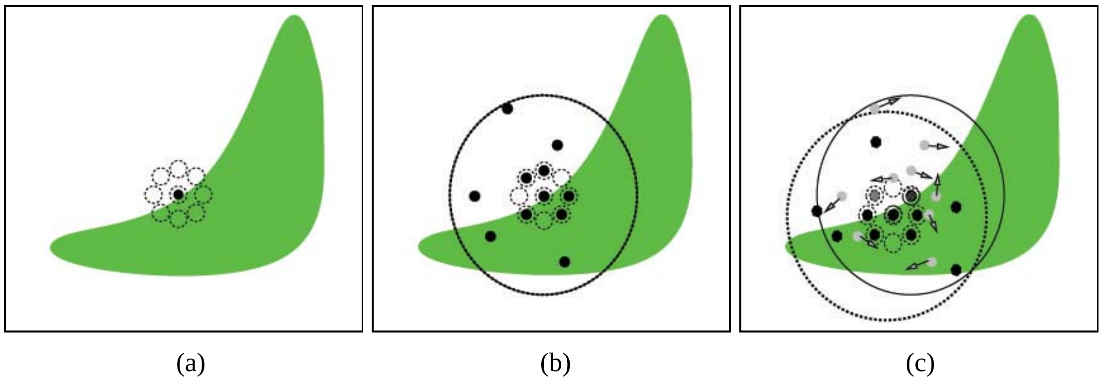


Fig. 3. An illustration of the behaviors of feature search entities. (a) As an entity, which is marked as a solid circle, moves to a new location in its local environment, it senses its neighboring locations, marked by dotted circles in this example. (b) When a triggering condition has been satisfied, as the location of the entity is right next to the border of a shaded region, the entity will self-reproduce some offspring entities within its local region. (c) At the following time step, the offspring will diffuse to new locations. By doing so, some of them will encounter new border feature locations as well and thereafter self-reproduce more entities. On the other hand, the entities that cannot find any border features after a given number of diffusion steps will be automatically turned off [40].

merging of homogeneous ones [41], [42]. An autonomy oriented approach has been used to tackle the same task [43]. 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 delivers 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 lifespan (a predefined number of steps) or wanders 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 are determined by their respective behavioral vectors, which contain 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 the herding behavior that considers only local information. A similar technique has also been applied to a feature extraction task such as border-tracing and edge detection [44].

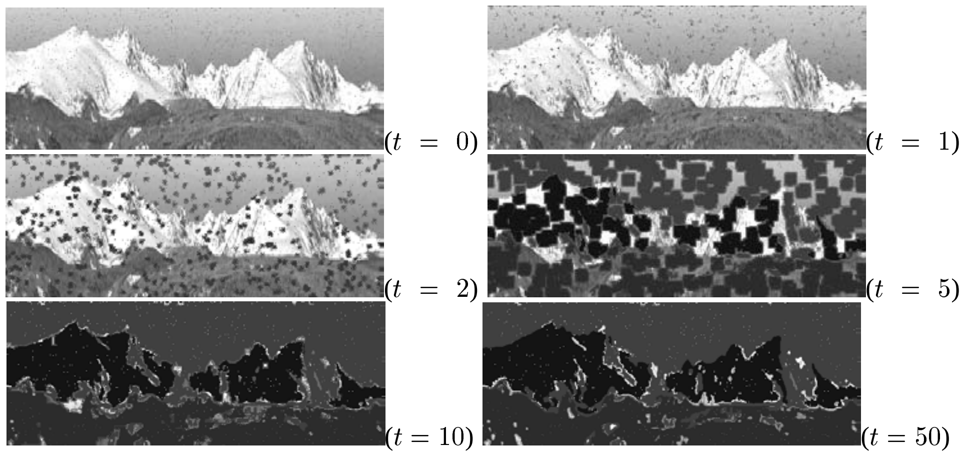


Fig. 4. Segmenting a landscape image that contains three complex-shaped regions.

TABLE I

THE NUMBER OF ACTIVE ENTITIES USED IN LABELING THE HOMOGENEOUS REGIONS OF A LANDSCAPE IMAGE.

Class	**# of active entities used** **($time\ step = 1 \sim 50$)**
Class-1	47,037
Class-2	75,473
Class-3	48,837

Examination

In order to examine the effectiveness of multiple classes of entities in the simultaneous detection of significant image segments, Liu *et al.* [40], [43], [44] have conducted several experiments in which various classes of entities are defined and used to extract different homogeneous regions from an image. Figure 4 presents an example of the AOC-based image segmentation task, where 1,500 entities (500 from each of three classes) are randomly distributed over the given image. Figure 4 presents a series of intermediate steps during collective image segmentation. Figure 4 ($t = 50$) gives the resultant markers as produced by the different classes of entities.

Computational Efficiency

In AOC-based image segmentation, the *computational costs* required can be estimated by counting how many active entities are being used over time (*i.e.*, the entities whose ages do not exceed a given lifespan). For the above-mentioned collective image segmentation task, Liu *et al.* have counted the number of active entities in each class that have been involved over a period of 50 time steps, as given in Table I. It can readily be noted that for each class, the number of active entities (*i.e.*, computational steps) involved in extracting a homogeneous region is less than the size of the given image, $526 \times 197 = 103,622$.

B. Summary

The AOC-by-fabrication approach focuses on building a mapping between a real problem and a natural phenomenon/system, the working mechanism behind which is usually more or less known (see Figure 5). In the mapping, the synthetic entities and their parameters (e.g., states, behaviors etc.) respectively correspond to the natural life-forms and their properties. Ideally, some special states of the natural phenomenon/system correspond to the solutions of the real problem.

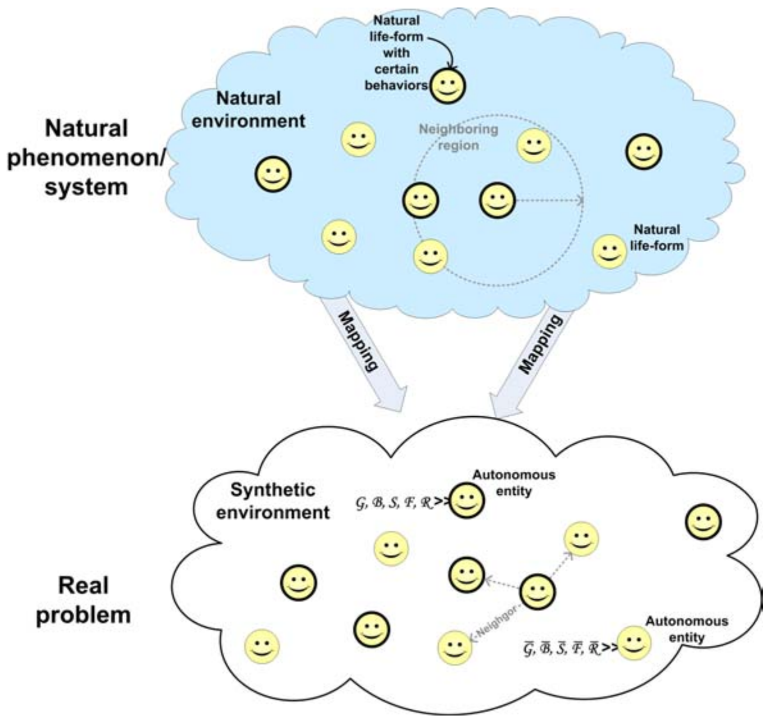

Natural life-form with certain behaviors
Natural environment
Natural phenomenon/ system
Neighboring region
Natural life-form
Mapping
Mapping
Synthetic environment
$\mathcal{G}, \mathcal{B}, \mathcal{S}, \mathcal{F}, \mathcal{R}$>>
Autonomous entity
Real problem
Neighgor
$\overline{\mathcal{G}}, \overline{\mathcal{B}}, \overline{\mathcal{S}}, \overline{\mathcal{F}}, \overline{\mathcal{R}}$>>
Autonomous entity

Fig. 5. The schematic diagram of the AOC-by-fabrication approach, which is intended to build a mapping between a real problem and a natural phenomenon/system. In the figure, entities are characterized by a group of parameters (e.g., $\mathcal{G}$, $\mathcal{B}$, $\mathcal{S}$, $\mathcal{F}$, and $\mathcal{R}$), the meanings of which will be given in Section VIII, Definitions 3-9.

From the above example, we can note the following common characteristics of the AOC-by-fabrication approach:

1. There is a population of individuals, each of which is mainly characterized by sets of goals, states, behaviors, behavioral rules etc. Individuals may be homogeneous or heterogeneous. Even in the homogeneous case, individuals may differ by some detailed parameters;
2. The composition of the population may change over time, by the process analogous to birth (amplification of the desired behavior) and death (elimination of the undesired behavior). But, in some applications, the population has the fixed number of entities;
3. Interactions between individuals are local, neither global information nor central executive to control behaviors or interactions is needed;

4. 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;
5. Local goals of individuals drive the selection of local behaviors at each step;
6. The global goal of the whole system is represented by a universal fitness function which gives a measure of the progress of the computation.

VI. AOC-by-Prototyping

With the help of a blueprint, engineers can build a model of a system in an orderly fashion. 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 behaviors 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. 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 developing and testing theories about society and social phenomena [15], [45]. Others methods have also been used to study traffic flow [46], traffic jam [47], [48], crowd control [49], and crowd dynamics [50].

A. Understanding Self-Organized Regularities

The Internet is growing in size, *i.e.*, the 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.

Researchers have identified several interesting, self-organized regularities related to the Web, ranging from the growth and evolution of the Web to the usage patterns in Web surfing. Many regularities are best represented by characteristic distributions following either a Zipf-like law [51] or a power law, that is, the probability P of a variant taking value k is proportional to $k^{-\alpha}$ where α is from 0 to 2.

Random-walk models [52], [53] and Markov-chain models [54], [55] have been used to simulate statistical regularities as empirically observed from the Web. However, these models do not relate the emergent regularities to the dynamic interactions between users and the Web, nor do they reflect the inter-relationships between user behavior and the contents or structure of the World Wide Web. The issues of user interest and motivation to navigate on the Web are among the most important factors that directly determine the navigation behaviors of users [56].

With an AOC-based approach to regularity characterization, Liu and Zhang [57], [58] have taken one step further by proposing a new computational model of Web surfing that takes into account the characteristics of users, such as interest profiles, motivations, and navigation strategies. By doing so, they attempt to answer the following questions:

- Is it possible to experimentally observe regularities similar to empirical Web regularities if we formulate the aggregation of user motivation? In other words, is it possible to account for empirical regularities from the point of view of motivation aggregation?

- Are there any navigation strategies or decision-making processes involved that determine the emergence of Web regularities, such as the distributions of user navigation depth?
- Will different navigation strategies or decision-making processes lead to different emergent regularities?
- Will the distribution of Web contents as well as page structure affect emergent regularities?

In order to answer the above questions, they have developed a white-box model. The features of this autonomy oriented model are: First, it incorporates the behavioral characteristics of Web users with measurable and adjustable attributes; second, it exhibits the empirical regularities as found in Web log data; and third, the operations in the model correspond to those in the real-world Web surfing.

AOC-Based Regularity Characterization

In the AOC-based regularity characterization, we view users as *information foraging entities* inhabiting in the Web space. The Web space is a collection of websites connected by hyperlinks. Each website contains certain information contents, and each hyperlink between two websites signifies certain content similarity between them. The contents contained in a website can be characterized using a multi-dimensional *content vector* where each component corresponds to the relative information weight on a certain topic. In order to build an artificial Web space that characterizes the topologies as well as connectivities of the real-world Web, we introduce the notion of an *artificial website* that may cover contents related to several topics and each topic may include a certain number of Web pages. Such a website may also be linked to other websites of similar or different topics through URLs. Thus, the Web space is generated by assigning topics to each Web page according to a certain statistical distribution. The variance of the distribution controls how similar all the pages are.

As users are the main subject to be studied, they are simulated in the system by associating with them an *interest vector*, again generated randomly based on a statistical distribution with certain variance. Therefore, there is a parameter to control the degree of overlap in interest between users.

Motivational Support Aggregation

When an information foraging entity finds certain websites in which the content is close to its interested topic(s), it will become more ready to *forage* to the websites at the next level, that is, it gets more *motivated* to *surf* deeper. On the other hand, when the entity does not find any interesting information after some foraging steps or it has found sufficient contents satisfying its interests, it will stop foraging and leave the Web space. In order to model such a motivation-driven foraging behavior, here we introduce a support function, P_t, which serves as the driving force for an entity to forage further. When the entity has found some useful information, it will get rewarded, and thus the support value will be increased. As the support value exceeds a certain threshold, which implies that the entity has obtained a sufficient amount of useful information, the entity will stop foraging. In other words, the entity is satisfied with what it has found. On the contrary, if the support value is too low, the entity will lose its motivation to forage further and thus leave the Web space.

TABLE II

THE DETAILED INFORMATION CORRESPONDING TO FIGURE 6.

Subfigure	**Statistic**	**Data source**	**Distribution**	**Power**	**Linear regression residual**	**Entities' satisfaction rate**[1]
(a)	foraging depth (step)	Recurrent entities	Power-law	$\beta_c = -1.843$	$\sigma = 0.011$	$\delta = 0.340$
(c)		Rational entities	Power-law	$\beta_c = -2.179$	$\sigma = 0.020$	$\delta = 0.197$
(e)		NASA Web Log	Power-law	$\beta_c = -2.669$	$\sigma = 1.174$	–
(b)	link-click-frequency[2]	Recurrent entities	Power-law	$\beta_l = -1.396$	$\sigma = 0.041$	–
(d)		Rational entities	Power-law	$\beta_l = -1.987$	$\sigma = 0.039$	–
(f)		NASA Web Log	Power-law	$\beta_l = -1.620$	$\sigma = 0.020$	–

[1] the ratio of the number of satisfied entities to the total number of entities that have surfed on the Web.
[2] 'link click' refers to the total times for which entities pass through a link.

Specifically, the support function is defined as follows:

$$P_{t+1} = P_t + \theta \times \Delta M_t + \phi \times \Delta R_t, \tag{1}$$

where

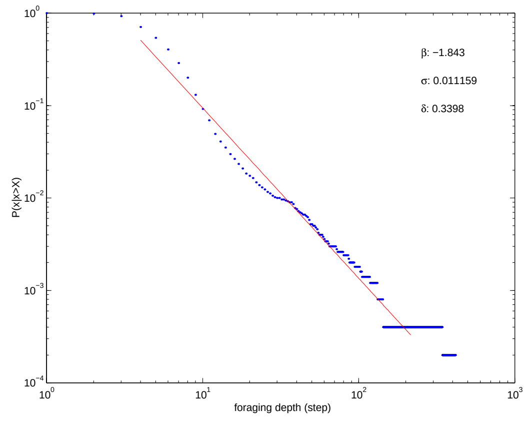
P_t:	support value at step t,
ΔM_t:	motivational loss at step t,
ΔR_t:	reward received at step t,
θ, ϕ:	coefficients of motivation and reward.

Validation of Autonomy Oriented Models

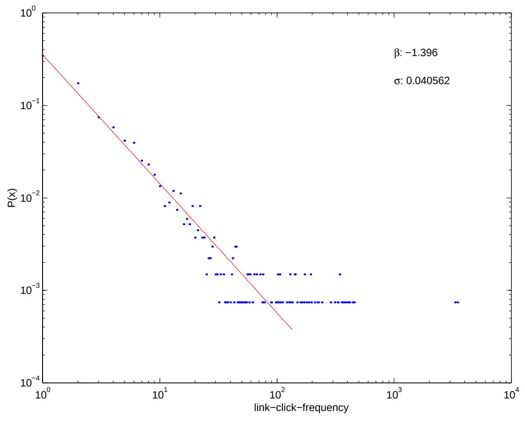
In order to validate their autonomy oriented model, Liu and Zhang [57], [58] have conducted several experiments. In the experiments, two measurements are of particular interests: the surfing-depth (step) distribution and the rank-frequency distribution of link clicks. The frequency of link clicks refers to the number of times for which a link is passed through by the entities. It is also called *link-click-frequency*. In the experiments, foraging entities are categorized into three different groups: *recurrent user* who is familiar with the Web structure, *rational user* who is new to the website but knows clearly what he/she is looking for, and *random user* who has no strong intention to get anything and just 'wanders' around.

Figures 6(a)-(d) present the statistical distributions of foraging depth and link-click-frequency obtained for recurrent and rational entities, respectively. It is interesting to observe from Figures 6(b) and (d) that the distributions of link-click-frequency exhibit a power law. A very similar result on the distribution of website popularity has been empirically observed and reported in [59].

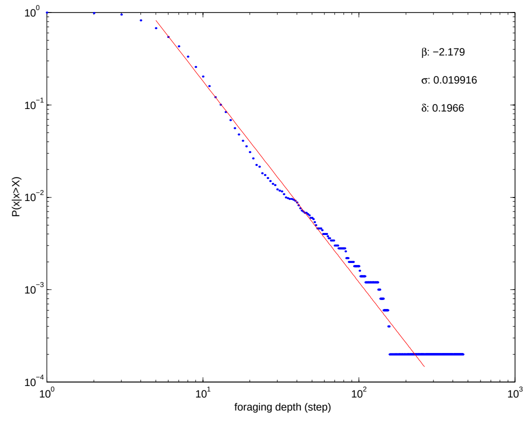
Liu and Zhang [57], [58] have also used real-world Web log datasets to compare their corresponding empirical distributions with those produced by the information foraging entities. The first dataset is a NASA Web log that recorded all HTTP requests received by the NASA Kennedy Space Center Web server in Florida from 23:59:59



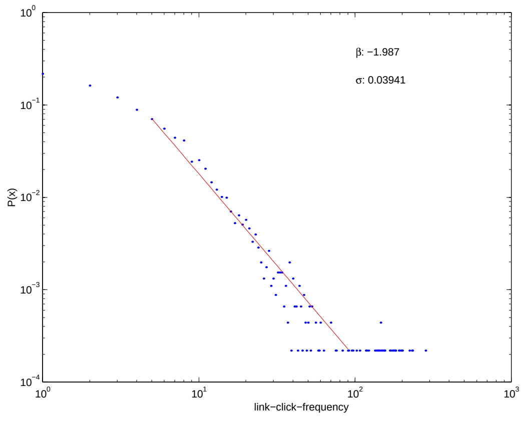
(a)



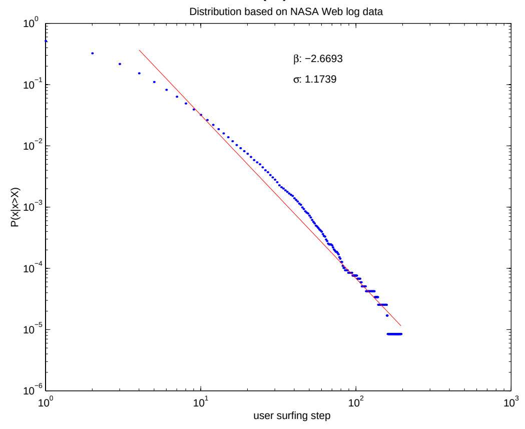
(b)



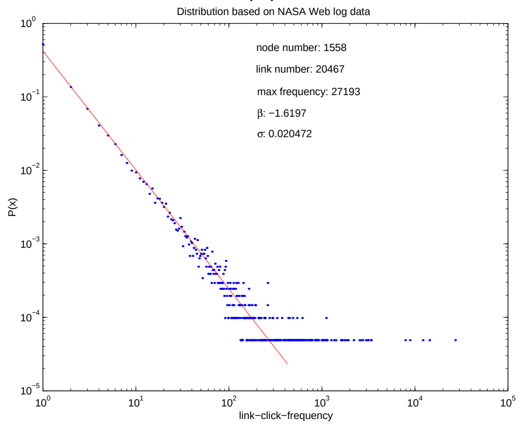
(c)



(d)

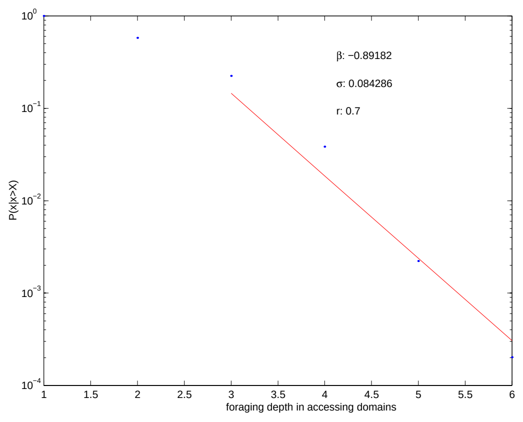


(e)

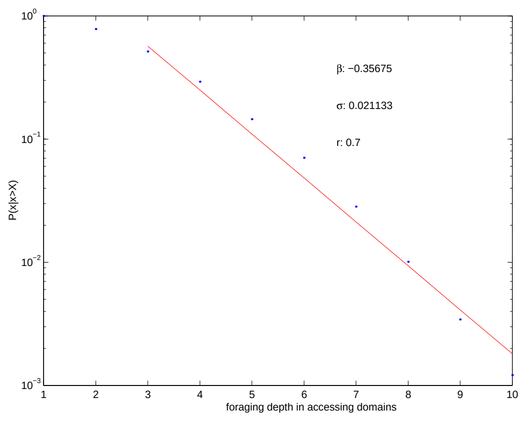


(f)

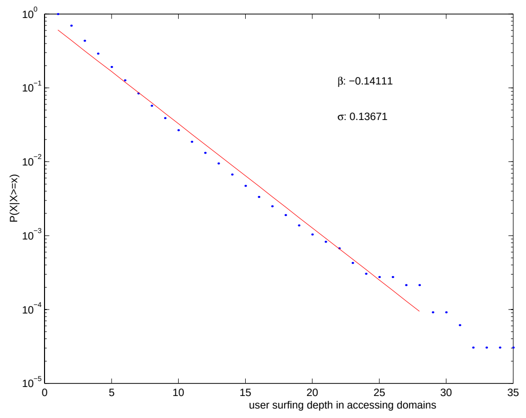
Fig. 6. Comparison between experimental data and real-world data on foraging depth and link-click-frequency [57]. '·' corresponds to experimental data and '−' corresponds to a linear-regression fitted line. Specifically, (a) and (c), (b) and (d) respectively show the cumulative distributions of foraging depth (step) and the distributions of link-click-frequency of *recurrent* entities and *rational* entities. (e) and (f) show the cumulative distribution of user surfing step and the distribution of link-click-frequency based on the real-world *NASA* Web log data (See Table II for more details).



(a)



(b)



(c)

Fig. 7. Visiting *domains*. '·' corresponds to experimental data and '$-$' corresponds to a linear-regression fitted line. (a) and (b) are the cumulative distributions of foraging depth in accessing domains by *recurrent* entities and *rational* entities, respectively. The distribution in (a) follows an exponential function with exponent $\beta_d = -0.892$ and residual $\sigma = 0.08$. The distribution in (b) also follows an exponential function with a *smaller* exponent $\beta_d = -0.357$ and residual $\sigma = 0.02$. (c) presents the cumulative distribution of user step in accessing domains in the real-world *Microsoft* Web log data. The distribution follows an exponential function with $\beta_d = -0.141$. The regression residual σ is about 0.137 [57].

August 3, 1995 to 23:59:59 August 31, 1995. The distributions of user surfing depth and link-click-frequency for the NASA dataset are shown in Figures 6(e) and (f), respectively.

From the above-mentioned synthetic and empirical results, we can note that the autonomy oriented models can readily generate power-law distributions in surfing step and link-click-frequency, which are similar to those from the real-world, and hence they offer a white-box explanation to the self-organized Web regularities.

In addition to the distributions of user steps in accessing pages and link-click-frequency, Liu and Zhang are also interested in the *distribution of user steps in accessing domains or topics* $-$ an issue of great importance that has never been studied before. Figures 7(a) and (b) presents the distributions of steps in accessing domains by recurrent and rational entities, respectively. From Figures 7(a) and (b), we can readily conclude that the cumulative probability distribution of entity steps in accessing domains follows an exponential function.

Liu and Zhang have further obtained an empirical dataset that records user behaviors in accessing the domains of a website. The dataset is a Web log file for the *Microsoft* corporate website, recording the domains or topics of `www.microsoft.com` that anonymous users visited in a one-week timeframe in February 1998. The distribution of user steps in accessing domains is shown in Figure 7 (c). If we compare Figures 7(a) and (b) with Figure 7 (c), we can note that the domain-visit regularity generated by our model characterizes the empirically observed domain-visit regularity well.

It is worth mentioning here that in [60], Yeung studied the above model in detail and validated it based on correlation dimension [61]. Yeung observed that the correlation dimension curves corresponding to the datasets obtained respectively from the above model and the NASA Web log have little difference. Therefore, he concluded that the model does show the behavior similar to that in the real world.

Parameters in Validation

In order to test the sensitivity of the parameters, Liu and Zhang [57], [58] conducted other experiments to examine the possible effects on distribution regularities while changing the number of agents, the number of domains or topics, and the parameters for the distribution of agent interest profiles and for the content distribution in the Web server. The results of those experiments revealed that altering these parameters will not change the regularities of power-law or exponential distributions as mentioned above, but only alter the shape parameters (i.e., β and σ) for the distributions. This further indicates that the distribution regularities, i.e., the power-law or exponential distributions, emerged from agent foraging behavior is stable and ubiquitous.

As for the small differences between the shapes of the synthetic and the empirical results in Figures 6 and 7, they can be eliminated by fine-tuning the related parameters. But, this is not the mostly concerned issue in Liu and Zhang's AOC-based regularity characterization.

B. Summary

AOC-by-prototyping is usually used to uncover the working mechanism behind a natural phenomenon/system. In doing so, at the beginning, a prototype will be built to characterize or simulate the natural counterpart. Then, by observing the difference between the natural phenomenon/system and the synthetic prototype, a trial-and-error process will be manually involved to fine-tune the prototype, especially the related parameters, such as adjust the behaviors of the synthetic entities. Figure 8 presents a schematic diagram of the process of AOC-by-prototyping.

In a sense, 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 is the guideline to parameter adjustment. The process can be summarized, with reference to the summary of AOC-by-fabrication in Section V-B, as follows:

1. States, evaluation function, goals, behaviors, and behavioral rules of an entity can be changed from one prototype to the next;
2. The definition of the environment can also be changed from one version to the next. Even the model can be modified completely;
3. There is an additional step to compare the synthetic model with the natural counterpart;
4. A new prototype is built by adopting steps 1 and 2 above, and repeating the whole process.

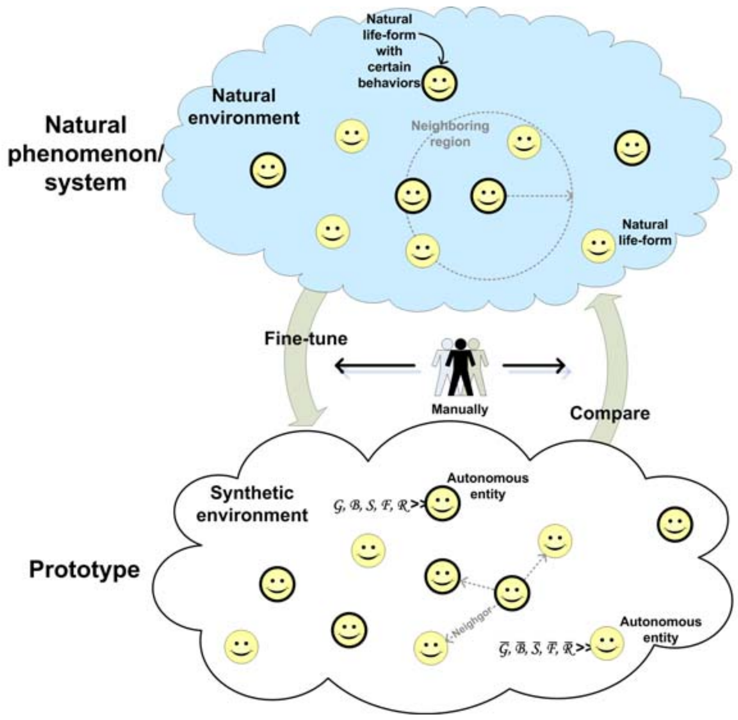

Natural life-form with certain behaviors
Natural environment
Natural phenomenon/ system
Neighboring region
Natural life-form
Fine-tune
Manually
Compare
Synthetic environment
Autonomous entity
$\mathcal{G}, \mathcal{B}, \mathcal{S}, \mathcal{F}, \mathcal{R}$>>
Prototype
Neighgor
Autonomous entity
$\overline{\mathcal{G}}, \overline{\mathcal{B}}, \overline{\mathcal{S}}, \overline{\mathcal{F}}, \overline{\mathcal{R}}$>>

Fig. 8. The schematic diagram of the process of AOC-by-prototyping, where the trial-and-error process, i.e., iterative *fine-tune* and *compare* steps, is manually operated (symbolized with a human sign in the figure).

VII. AOC-BY-SELF-DISCOVERY

AOC-by-self-discovery emphasizes the ability of an AOC-based computational system to find its own way to achieve what AOC-by-prototyping can do. The ultimate goal is to have a fully automated algorithm that can adjust its own parameters for different application domains. In other words, the AOC itself becomes autonomous.

Adaptive evolutionary algorithms [62–65] have been proposed that automatically tune some of the parameters related to the algorithms, such as mutation rate [6], [66–70], mutation operator [71–74], crossover rate [75], [76], crossover operator [77], [78], and population size [70]. They usually track changes in progress measures, such as online and offline performance [64], the ratio of average fitness to the best fitness, and the ratio of the worst fitness to average fitness, among others.

Meta-EA is another group of self-improving EAs that does not rely on the specific instruction of the designer [79]. An EA [80–83] or another intelligent system [84–87] is used to control a population of EA in the way similar to an EA optimizing the parameters of the problem at hand.

Other examples of AOC-by-self-discovery include the evolution of emergent computation where a GA is used to evolve the rules of a cellular automata for a synchronization task [17], [18], [88] and for generating test patterns for hardware circuits [89]. A variant of the latter example employs a selfish gene algorithm [90].

A. Global Optimization

Global optimization [91] is dedicated to finding the optimal solution to a function $F(x)$ where $x = \{x_1, x_2, ..., x_n\}^T$ is an n-dimensional vector representing the parameters of function. The optimal solution is then represented by $F(x^*)$ such that

$$\forall x,\ F(x^*) \leq F(x). \tag{2}$$

Many algorithms have been developed over the years to tackle this problem [92]- [94]. This is a challenging task because: First, the landscape of the function to be optimized in unknown and search algorithms are likely to be trapped in suboptimal solutions; second, the dimensionality of the function presents problems to a search algorithm. The task is further complicated if one needs to adjust the various parameters of the search algorithms.

Search by Diffusion

Inspired by diffusion in nature, Tsui and Liu [95–97] have developed an AOC-based method, called *Evolutionary Diffusion Optimization* (EDO), to tackle the global optimization task. A population of entities are used to represent candidate solutions to the optimization problem in hand. The goal is to build a collective view of the landscape of the search space by sharing information among entities. Specifically, each entity performs search in its local proximity and captures the 'direction' information of the landscape in a *probability matrix* – the likelihood estimate of success with respect to the direction of the motion for each object variable.

Autonomy Oriented Evolutionary Diffusion Model

EDO defines three types of local behaviors for each entity, namely *diffuse*, *reproduce*, and *aging*. Free ranging entities that are searching for a position better than their birthplaces are called *active* entities. Those entities that have already become parents are called *inactive* entities.

Entities in EDO explore uncharted locations in the solution space by *diffusion*. *Rational move* refers to the kind of diffusion where an entity modifies its object vector by drawing a random number for each dimension of the object vector. Each random number is then used to choose between the set of fixed steps according to the probability matrix. Adjustment is then made by adding the product of the number of steps and the size of a step to the entry in question in the object vector. This process is repeated until all dimensions of the object vector are covered. The updated object vector then becomes the new position of the entity in the solution space.

As an entity becomes older, it becomes more eager to find a better position. It, therefore, will probabilistically decide to act wild and take a *Random Walk* with probability given by $exp(-\frac{lifespan-age}{\alpha})$ where α is a scaling factor. An entity will first choose randomly a direction of diffusion for each dimension, *i.e,* either no change, towards the upper bound, or towards lower bound. In case a move is to be made, a new value between the chosen bound and the current value is then picked randomly. The process ends when all dimensions of the object vector are updated.

At the end of an iteration, the fitness of all active entities are compared with that of their parents, which have (temporarily) become stationary. All entities with higher fitness will *reproduce* via asexual reproduction – a reproducing entity replicates itself a number of times and sends the new entities off to a new location by *rational move*. Parents and their offspring share the same probability matrix. Only when an entity becomes a parent then a new probability matrix will be created for it, which is an exact copy of the parent's updated one. Sharing the probability matrix between parents and siblings enables entities from the same family to learn from each other's successes as well as failures.

Aging is the process by which consistently unsuccessful entities are eliminated from the system. This is controlled by a *lifespan* parameter. Exceptions are granted to those entities whose ages have reached the set limit but that have the above-average fitness. On the other hand, entities whose fitness is at the lower 25% of the population will be eliminated before their lifespan expires.

Adaptation

Search algorithms need a scheme to implement the strategy that says:'good' moves need to be rewarded while 'bad' moves should be discouraged. All entities in EDO maintain a close link between parents and offspring via sharing the probability matrix. Therefore, it is very easy for EDO to implement the above strategy and EDO has two feedback mechanisms for updating the probability matrix of parent. *Positive feedback* increases the value of the entry in the probability matrix that corresponds to the 'good' move. The update rule is:

$$p'_{ij} = \frac{p_{ij} + \delta}{1 + \delta}, \tag{3}$$

where p_{ij} refers to the probability of object variable i and step multiplier j that relates to the previous movement(s) and δ is a fixed adjustment factor. In contrast, *negative feedback* reduces the relevant probabilities that relate to the 'bad' move using:

$$p'_{ij} = p_{ij} \times (1 - \delta). \tag{4}$$

While negative feedback is exercised after each diffusion, positive feedback can only take place after an entity has become a parent. Note also that all probabilities are normalized using their respective sum after updating.

EDO also adapts the *step_size* parameter, which determines the amount of change during *diffusion*, over time based on the performance measurement of the population. *Step_size* is reduced if the population has not improved over a period of time. Conversely, if the population has been improving continuously for some time, *step_size* is increased. The rationale is that the entities in the neighborhood of a minimum need to make finer steps for careful exploitation, while using a larger *step_size* during a period of continuous improvement attempts to speed up the search.

Illustration

Figure 9 shows the distribution of entities used to optimize $F_1(x) = \sum_{i=1}^{2} x_i^2$. Initially, five entities are positioned randomly in the solution space - three with x-value around -80, one around -20, and one around zero - but none is within the central 20 by 20 zone around the origin. As the search progresses to generation 10, more and more entities appear inside the central zone. This trend continues between generation 10 and generation 80.

The diameter of the occupied area increases between generations 10 and 30 due to entities exploring the solution space, and shrinks between generations 30 and 80 when the direction information is being learned. It can also be observed that there is an increase in the number of entities along the axes, which can be considered as suboptimal solutions since one of the two variables is at its lowest value with respect to F_1.

The number of entities peaks at around generation 80 but drops drastically in the next 10 generations. This is because entities not around the origin or along the axes have much lower fitness than the population average and are gradually eliminated. The second phase of search sees entities along the axes being eliminated as they are occupying a suboptimal position. As all entities continue to move towards the origin, those entities that are further away from the origin are eliminated. The last four plots in Figure 9 zoom in to the origin with an increasing scale where the majority of the best entities are located.

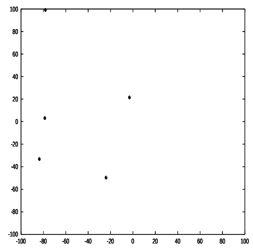
B. Summary

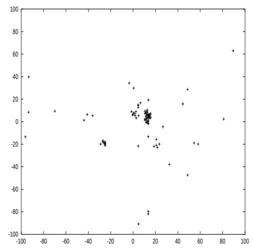
AOC-by-self-discovery can be used not only to build a mapping between a real problem and a certain natural phenomenon/system, but also to reveal the operating mechanism behind a natural phenomenon/system. In general, it combines the uses of AOC-by-fabrication and AOC-by-prototyping. In its implementation, AOC-by-self-discovery is the same as AOC-by-prototyping except that the process of trial-and-error in AOC-by-self-discovery is automated (See Figure 10). The full automation of the prototyping process is achieved by having an autonomous entity to control another level of autonomous entities. The example described above shows that AOC-by-self-discovery is indeed a viable proposition. The steps for engineering this kind of AOC algorithm is the same as those stated in Section V-B with the addition of one rule:

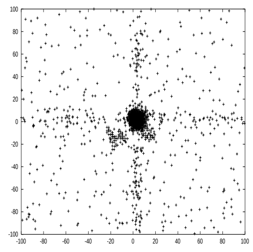
1. System parameters are self-adapted according to some performance measurements.

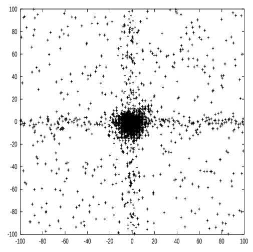
VIII. General Framework of Autonomy Oriented Computing

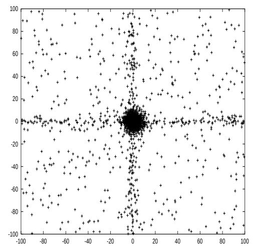
In the preceding three sections, we have illustrated three AOC-based approaches through examples, which actually correspond to three goals of AOC, respectively. Although the resulting AOC systems are implemented with different approaches, they have the similar structure and operating mechanism. In this section, we generalize the previous examples and provide a common framework for AOC systems. Specifically, (1) we present the formal definitions of some key elements, such as environment, entity, and interaction. These definitions are meant to show

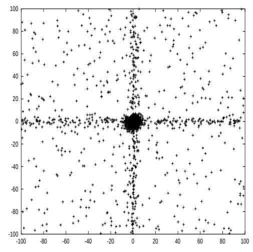

(a) Start

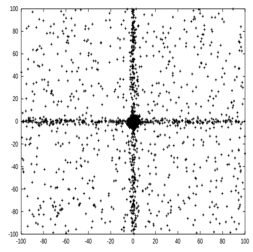

(b) Iteration 10

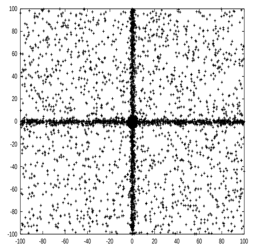

(c) Iteration 20

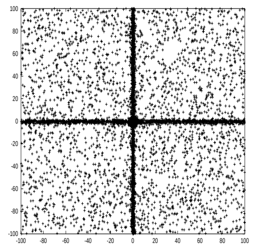

(d) Iteration 30

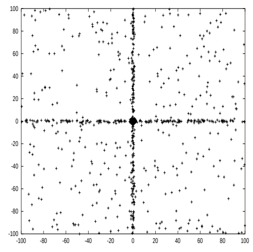

(e) Iteration 40

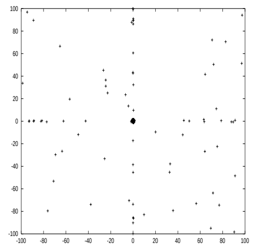

(f) Iteration 50

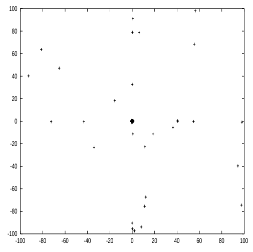

(g) Iteration 60

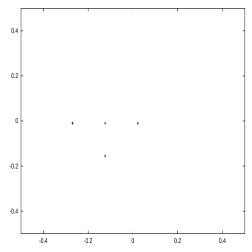

(h) Iteration 70

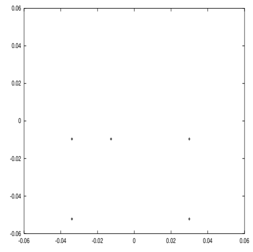

(i) Iteration 80

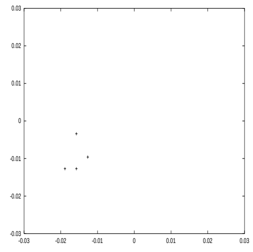

(j) Iteration 90


(k) Iteration 100


(l) Iteration 110


(m) Iteration 120


(n) Iteration 160


(o) Iteration 200

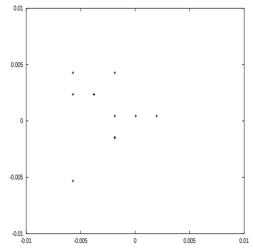

(p) Iteration 220

Fig. 9. Distributions of entities in the solution space for optimizing the 2-D version of F_1. Plots (a)-(l) show the range between ± 100 in both axes. The ranges for plots (m)-(p) are ± 0.5, ± 0.06, ± 0.02, and ± 0.01, respectively. The number of entities shown in plots (m)-(p) are 4 out of 5, 5 out of 5, 4 out of 6, and 11 out of 14, respectively.

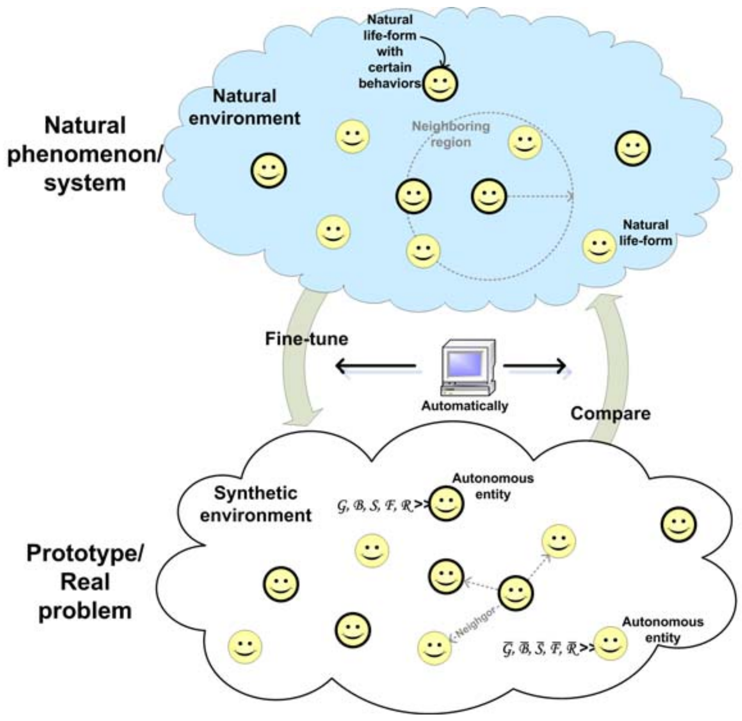

Natural life-form with certain behaviors
Natural environment
Natural phenomenon/ system
Neighboring region
Natural life-form
Fine-tune
Automatically
Compare
Synthetic environment
Autonomous entity
$\mathcal{G}, \mathcal{B}, \mathcal{S}, \mathcal{F}, \mathcal{R}$>>
Prototype/ Real problem
Neighbor
Autonomous entity
$\mathcal{G}, \mathcal{B}, \mathcal{S}, \mathcal{F}, \mathcal{R}$>>

Fig. 10. The schematic diagram of the process of AOC-by-self-discovery. As compared with AOC-by-prototyping (See Figure 8), here the trial-and-error process, i.e., repeated *fine-tune* and *compare* steps, is automatically implemented by the system (Symbolized with a computer sign in the figure).

the interrelationships among common concepts involved in AOC; (2) based on the definitions, we highlight the essence of AOC systems, i.e., the process of self-organization of entities; (3) finally, we formulate the examples under this framework.

As we have known, an AOC system naturally contains an environment and a population of autonomous entities.Then, we can formally define it as follows.

Definition 1 *An AOC system is a tuple $\langle\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}, \mathbf{E}, \mathbf{\Phi}\rangle$, where $\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}$ denotes a group of autonomous entities; $\mathbf{E}$ is an environment in which entities $\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}$ reside; $\mathbf{\Phi}$ is the system objective function, which is usually a nonlinear function of entity states.*

A. Elements of an AOC System

1) The Environment: As one of the main components in an AOC system, an environment usually plays three roles. Firstly, it serves as the domain in which autonomous entities roam. This is a static view of the environment. Secondly, the environment acts as the *noticeboard* where the autonomous entities can read and/or post their local information. In this dynamic view, the environment is changing all the time. For example, in the AOC-based feature search and extraction, an entity can read the grey-scale intensity of the pixel where it stays. On the other hand, if the entity finds the pixel belongs to a homogeneous region, it will post information, i.e., put a label on the pixel.

In this sense, the environment can also be regarded as an indirect communication medium among entities. Thirdly, the environment keeps a central clock that helps synchronize the behaviors of all autonomous entities, if necessary.

Definition 2 *Environment* $\mathbf{E}$ *is characterized by a set* $\mathcal{ES} = \{es_1, \cdots, es_{N_{\mathcal{ES}}}\}$, *where each* $es_i \in D_{es_i}$ *represents a static or dynamic attribute. At each moment,* $\mathcal{ES}$ *also represents the current state of environment* $\mathbf{E}$. *Therefore, the state space of* $\mathbf{E}$ *is given by* $D_{\mathcal{ES}} = D_{es_1} \times \cdots \times D_{es_{N_{\mathcal{ES}}}}$.

In the AOC-based feature search and extraction, for example, the features of an environment are mathematically described using two attributes: the static one records the grey-scale intensity corresponding to each pixel in the image, while the dynamic one records a label for each pixel to show whether or not the pixel locates in a certain homogeneous region.

2) Autonomous Entities: An autonomous entity modifies its own state, exerts changes to the environment and/or affects other entities. Central to an autonomous entity is its *local behaviors* and *behavioral rules* that governs how it should act or react to the information received from the environment and its neighbors. The local behavior and behavioral rules determine the next state to which the entity will transit.

Definition 3 *An autonomous entity* $\mathbf{e}$ *is a tuple* $\langle \mathcal{S}, \mathcal{F}, \mathcal{G}, \mathcal{B}, \mathcal{R} \rangle$, *where* $\mathcal{S}$ *describes the current state of entity* e; $\mathcal{F}$ *is an evaluation function;* $\mathcal{G}$ *is the goal set of entity* $\mathbf{e}$; $\mathcal{B}$ *and* $\mathcal{R}$ *are behaviors and behavioral rules, respectively.*

According to the differences in $\mathcal{S}, \mathcal{F}, \mathcal{G}, \mathcal{B}$, and $\mathcal{R}$, entities in a AOC system can be categorized into different classes. In Figures 5, 8, and 10, there are two classes of synthetic entities, i.e., bold face class and regular face class. Before further description, let us define the neighbors of an entity.

Definition 4 *The neighbors of entity* $\mathbf{e}$ *are a group of entities* $\mathcal{L}^e = \{l_1^e, \cdots, l_{N_{\mathcal{L}}}^e\}$. *The relationship (e.g., distance) between each* l_i^e *and entity* $\mathbf{e}$ *satisfies an application-dependent constraint.*

In different AOC systems, the neighbors of an entity can be fixed or dynamically changed. For example, in the Boids system [98], the others in a boid's vision field are seen as its neighbors. Since boids are flying everywhere, their neighbors are dynamically changed over time.

At each moment, an entity is in a certain state. It, according to its behavioral rules, selects and performs its behaviors so as to achieve certain goals with respect to its state. While doing so, it needs to interact with its neighbors and/or its environment to get the necessary information. In the following, we will further describe the notions of state, evaluation function, goal, local behavior, and behavioral rule for an autonomous entity.

Definition 5 *State* $\mathcal{S}$ *of autonomous entity* $\mathbf{e}$ *is characterized by a set of static or dynamic attributes, i.e.,* $\mathcal{S} = \{s_1, \cdots, s_{N_{\mathcal{S}}}\}$, *where* $s_i \in D_{s_i}$. *Thus,* $D_{\mathcal{S}} = D_{s_1} \times \cdots \times D_{s_{N_{\mathcal{S}}}}$ *corresponds to the state space of entity* $\mathbf{e}$.

As we have seen in Section V, the state of an autonomous entity in the AOC-based feature search and extraction is characterized by its lifespan as well as its current position, age, and activity, where lifespan is pre-defined and fixed; position, age, and activity are dynamically changed.

Before an entity uses its behavioral rules to select its behavior, it needs to assess its current condition, including its own state (internal state) and/or those of its neighbors and environment. In some applications, while selecting its behavior, an entity needs to assess its choices, i.e., possible states at the next step.

Definition 6 *The assessment of conditions is performed by an entity using one of the following evaluation functions. They assess:*

- *Internal state:*

$$\mathcal{F} : \hat{D}_{\mathcal{S}} \longrightarrow R, \tag{5}$$

- *State of its environment:*

$$\mathcal{F} : \hat{D}_{\mathcal{ES}} \longrightarrow R, \tag{6}$$

- *States of its neighbors:*

$$\mathcal{F} : \prod_{l_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{l_i^e}}) \longrightarrow R, \tag{7}$$

- *Internal state and that of its environment:*

$$\mathcal{F} : \hat{D}_{\mathcal{S}} \times \hat{D}_{\mathcal{ES}} \longrightarrow R, \tag{8}$$

- *Internal state and those of its neighbors:*

$$\mathcal{F} : \hat{D}_{\mathcal{S}} \times \prod_{l_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{l_i^e}}) \longrightarrow R, \tag{9}$$

where R is the value domain of $\mathcal{F}$ (e.g., the set of real numbers); $\hat{D}_{\mathcal{S}}$ is a Cartesian product of elements in a subset of $\{D_{s_i}\}$, i.e., $\hat{D}_{\mathcal{S}} \subseteq D_{\mathcal{S}}$; $\hat{D}_{\mathcal{ES}}$ is also a Cartesian product, $\hat{D}_{\mathcal{ES}} \subseteq D_{\mathcal{ES}}$; l_i^e denotes the i^{th} neighbor of entity $\mathbf{e}$; $\mathcal{S}^{l_i^e}$ and $D_{\mathcal{S}^{l_i^e}}$ denote the current state and the state space of entity l_i^e, respectively; $\hat{D}_{\mathcal{S}^{l_i^e}}$ is a subset in $D_{\mathcal{S}^{l_i^e}}$; Π is the Cartesian product operator.

Note that in the above definition, we use $\hat{D}_{\mathcal{S}}$, $\hat{D}_{\mathcal{ES}}$, and $\hat{D}_{\mathcal{S}^{l_i^e}}$, not $D_{\mathcal{S}}$, $D_{\mathcal{ES}}$, and $D_{\mathcal{S}^{l_i^e}}$. This is because the evaluation function $\mathcal{F}$ is based on a subset of attributes, which represents the entity as well as its neighbors and environment. Again, let us take the AOC-based feature search and extraction as an example. The evaluation function in this example is based on the position of an entity (i.e., the pixel where the entity stays currently) and the gray-scale intensity of the corresponding pixel and its neighboring region. Here, the evaluation function is independent of age, activity, and lifespan, although they are also used to characterize the entity's state.

Generally speaking, the behavior of an entity in AOC systems is goal-directed. The goal of an entity is defined as follows:

Definition 7 *Entity $\mathbf{e}$ can be engaged in a set of goals over time, as denoted by $\mathcal{G} = \{g_1, \cdots, g_{N_\mathcal{G}}\}$. Each goal g_i is to achieve a certain state $\mathcal{S}'$ such that evaluation function $\mathcal{F}$ takes a certain pre-defined value α, i.e., $g_i = \{\mathcal{S}' | \mathcal{F}(\cdot) = \alpha\}$, where α is a constant.*

In an AOC system, at a given moment each entity $\mathbf{e}$ usually has only one goal and all entities may share the same goal. Although, the self-organizing behavior of an autonomous entity is goal-directed, the entity does not explicitly know the global goal of the whole system and the task that the system is performing.

Definition 8 *The local behavior set of entity $\mathbf{e}$ is $\mathcal{B} = \{b_1, \cdots, b_{N_\mathcal{B}}\}$. Each behavior b_i is a mapping in one of the following forms:*

- *Self-reproduce:*
$$b_i : \mathbf{e} \to \mathbf{e}^m, \tag{10}$$
which is a reproduction-like behavior. It denotes that entity $\mathbf{e}$ replicates itself m times (i.e., breed m offspring);
- *Die:*
$$b_i : \mathbf{e} \to \emptyset, \tag{11}$$
which denotes that entity $\mathbf{e}$ vanishes from the environment;
- *Change internal state:*
$$b_i : \hat{D}_\mathcal{S} \to \hat{D}_\mathcal{S}, \tag{12}$$
- *Change the state of its environment:*
$$b_i : \hat{D}_{\mathcal{ES}} \to \hat{D}_{\mathcal{ES}}, \tag{13}$$
- *Change internal state and that of its environment:*
$$b_i : \hat{D}_\mathcal{S} \times \hat{D}_{\mathcal{ES}} \to \hat{D}_\mathcal{S} \times \hat{D}_{\mathcal{ES}}, \tag{14}$$
- *Change internal state and those of its neighbors:*
$$b_i : \hat{D}_\mathcal{S} \times \prod_{l_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{l_i^e}}) \to \hat{D}_\mathcal{S} \times \prod_{l_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{l_i^e}}), \tag{15}$$
where $\hat{D}_\mathcal{S}$, $\hat{D}_{\mathcal{ES}}$, $\hat{D}_{\mathcal{S}^{l_i^e}}$, and Π have the same meanings as in Definition 6.

Definition 9 *The behavioral rule set of entity $\mathbf{e}$ is $\mathcal{R} = \{r_1, \cdots, r_{N_\mathcal{R}}\}$. Each behavioral rule r_i is to select one or more local behaviors to perform at each step. Behavioral rules can be classified into two types:*

- *Evaluation-based rules:*
$$r_i : Dom(\mathcal{F}) \to \{\hat{\mathcal{B}}\}, \tag{16}$$
where $Dom(\mathcal{F})$ denotes the value domain of evaluation function $\mathcal{F}$; $\hat{\mathcal{B}} \subseteq \mathcal{B}$; $\{\hat{\mathcal{B}}\}$ is a set of subsets in $\mathcal{B}$.
- *Probability-based rules:*
$$r_i : [0,1] \to \{\hat{\mathcal{B}}\}, \tag{17}$$
where each subset $\hat{\mathcal{B}}$ is assigned a probability, $p_{\hat{\mathcal{B}}}$, which may be fixed or dynamically changed over time. This type of rules probabilistically select behaviors from $\mathcal{B}$.

For example, in the AOC-based feature search and extraction, an entity has four local behaviors, namely, breeding, pixel labeling, diffusion, and decay. At each step, it chooses to breed some offspring and label its current position, or diffuse to another position and decay, based on the assessment to its state. Here, the behavioral rule is evaluation-based.

3) System Objective Function: In an AOC system, the *system objective function*, $\mathbf{\Phi}$, guides the system to evolve towards desired states/patterns.

Definition 10 *In an AOC system,* system objective function $\mathbf{\Phi}$ *is defined as a function of the states of some entities. For different applications, the definitions of* $\mathbf{\Phi}$ *can be categorized into two types. Let* $\{\mathbf{e}_i\}$ *be the set of entities.*

- *State-oriented:*
$$\mathbf{\Phi} : \prod_{e_k \in \{\mathbf{e}_i\}} \hat{D}_{\mathcal{S}^{e_k}} \longrightarrow R^m, \tag{18}$$
where $\hat{D}_{\mathcal{S}^{e_k}}$ *is a subset of the state space,* $D_{\mathcal{S}^{e_k}}$*, of entity* $\mathbf{e}_k$*, and* m *is the dimensionality of the value domain of* $\mathbf{\Phi}$*.*
- *Process-oriented:*
$$\mathbf{\Phi} : \Big\{ \prod_{e_k \in \{\mathbf{e}_i\}} \hat{D}_{\mathcal{S}^{e_k}} \Big\}^{\Pi} \longrightarrow R^m, \tag{19}$$
where $\{\cdot\}^{\Pi}$ *denotes multiplying the elements inside the braces a finite number of times.*

It should be pointed out the $\mathbf{\Phi}$ is usually a nonlinear function. In the above definition, the first type of $\mathbf{\Phi}$ concerns the state of an AOC system, which is measured with an m-dimensional vector. Specifically, the system aims at certain desired states. With the second type of $\mathbf{\Phi}$, an AOC system attempts to exhibit certain desired characteristics or patterns in its evolutionary process.

B. Interactions in an AOC System

The emergent behavior of an AOC system originates from its internal interactions. This subsection addresses the interactions in an AOC system. Generally speaking, there are two kinds of interactions, namely, *interactions between entities and their environment* and *interactions among entities*.

1) Interactions between Entities and their Environment: The interactions between an entity and its environment are implemented through the state change caused by the entity's behaviors.

Definition 11 *The interactions between entity* $\mathbf{e}$ *and its environment* $\mathbf{E}$ *are defined through a set of mappings* $\{\mathcal{I}_{\mathbf{eE}}\}$*, where* $\mathcal{I}_{\mathbf{eE}}$ *has one of the following forms:*

$$\mathcal{I}_{\mathbf{eE}} : \hat{D}_{\mathcal{ES}} \rightarrow_{b_i} \hat{D}_{\mathcal{ES}}, \tag{20}$$

or

$$\mathcal{I}_{\mathbf{eE}} : \hat{D}_{\mathcal{S}} \times \hat{D}_{\mathcal{ES}} \rightarrow_{b_i} \hat{D}_{\mathcal{S}} \times \hat{D}_{\mathcal{ES}}, \tag{21}$$

where '$\rightarrow_{b_i}$' indicates that $\mathcal{I}_{\mathbf{eE}}$ is in fact a behavior, b_i, of entity $\mathbf{e}$*, which is related to the state subspace $\hat{D}_{\mathcal{ES}}$ of entity* $\mathbf{e}$*'s environment (See Definition 8, Equations 13 and 14).*

Figure 11 presents a schematic diagram of interactions between two entities $\mathbf{e}_A$, $\mathbf{e}_B$ and their environment $\mathbf{E}$.

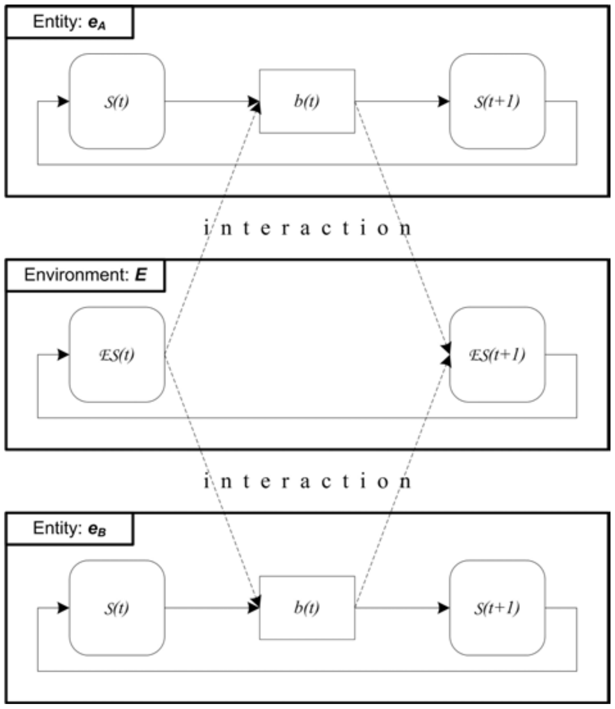


Fig. 11. Interactions (i.e., the dashed lines) between two entities $\mathbf{e_A}$, $\mathbf{e_B}$ and their environment $\mathbf{E}$, which are caused by entity behaviors (i.e., $b(t)$). The solid lines denotes the state changes of entities and their environment also cause by entity behaviors.

2) Interactions among Entities: Different AOC systems may have different fashions of interactions among their entities. Those interactions can be further categorized into two subtypes, *direct interactions* and *indirect interactions*. Which subtype of interactions will be used in an AOC system is determined by the specific applications.

Direct interactions are implemented through direct state information exchanges among entities. In an AOC system with direct interactions, each entity can interact with its *neighbors*. Figure 12 presents a schematic diagram of direct interactions between two autonomous entities $\mathbf{e_A}$ and $\mathbf{e_B}$.

Definition 12 *The direct interactions between entities* $\mathbf{e_A}$ *and* $\mathbf{e_B}$ *are a set of mapping tuples* $\{\langle\mathcal{I}_{AB},\mathcal{I}_{BA}\rangle\}$,

$$\mathcal{I}_{AB} : \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{S}^B} \rightarrow_{b^A} \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{S}^B}, \tag{22}$$

and

$$\mathcal{I}_{BA} : \hat{D}_{\mathcal{S}^B} \times \hat{D}_{\mathcal{S}^A} \rightarrow_{b^B} \hat{D}_{\mathcal{S}^B} \times \hat{D}_{\mathcal{S}^A}, \tag{23}$$

where '$\rightarrow_{b^A}$' and '$\rightarrow_{b^B}$' denote that $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$ are respectively behaviors of entities $\mathbf{e_A}$ *and* $\mathbf{e_B}$*, which are related to the states of their neighbors (See Definition 8, Equation 15); $\hat{D}_{\mathcal{S}^A}$ and $\hat{D}_{\mathcal{S}^B}$ are respectively subsets of* *state spaces of entities* $\mathbf{e_A}$ *and* $\mathbf{e_B}$.

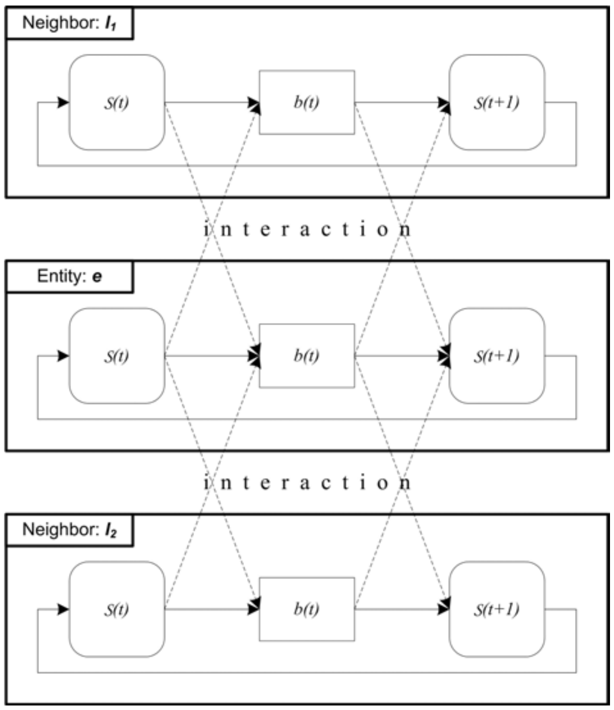


Fig. 12. Direct interactions (i.e., the dashed lines) between entity $\mathbf{e}$ and its two neighbors, namely, entities $\mathbf{l_1}$ and $\mathbf{l_2}$. The solid lines denotes the state changes of three entities cause by their behaviors.

Indirect interactions are implemented through the communication medium role of an environment. They can be separated into two stages: (1) Through the interactions between an entity and its environment, the entity will 'transfer' its information to the environment; (2) Later, while other entities behave, if necessary, they will consider the information, which has been 'transfered' to the environment by the previous agent.

Definition 13 *The indirect interactions between entities* $\mathbf{e_A}$ *and* $\mathbf{e_B}$ *are a set of mapping tuples* $\{\langle \mathcal{I}_{A,E}, \mathcal{I}_{B,E} \rangle\}$, *where interaction* $\mathcal{I}_{A,E}$ *between entity* $\mathbf{e_A}$ *and environment* $\mathbf{E}$ *occurs before interaction* $\mathcal{I}_{B,E}$ *between entity* $\mathbf{e_B}$ *and environment* $\mathbf{E}$*, namely,*

- *At time* t*:*

$$\mathcal{I}_{A,E} : \hat{D}_{\mathcal{ES}} \rightarrow_{b^A} \hat{D}_{\mathcal{ES}}, \text{ or} \tag{24}$$

$$\mathcal{I}_{A,E} : \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{ES}} \rightarrow_{b^A} \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{ES}}; \tag{25}$$

- *At time* t'*:*

$$\mathcal{I}_{B,E} : \hat{D}_{\mathcal{ES}} \rightarrow_{b^B} \hat{D}_{\mathcal{ES}}, \text{ or} \tag{26}$$

$$\mathcal{I}_{B,E} : \hat{D}_{\mathcal{S}^B} \times \hat{D}_{\mathcal{ES}} \rightarrow_{b^B} \hat{D}_{\mathcal{S}^B} \times \hat{D}_{\mathcal{ES}}, \tag{27}$$

where $t < t'$; '$\rightarrow_{b^A}$' and '$\rightarrow_{b^B}$' denote that $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$ are respectively behaviors of entities $\mathbf{e_A}$ and $\mathbf{e_B}$; $\hat{D}_{\mathcal{S}^A} \subseteq D_{\mathcal{S}^A}$ and $\hat{D}_{\mathcal{S}^B} \subseteq D_{\mathcal{S}^B}$, where $D_{\mathcal{S}^A}$ and $D_{\mathcal{S}^B}$ are state spaces of entities $\mathbf{e_A}$ and $\mathbf{e_B}$.

C. Discussions on Homogeneity, Heterogeneity, and Hierarchy of Entities

Here we discuss the homogeneity, heterogeneity, and hierarchy issues of entities in an AOC system.

In the preceding sections, we have formally defined an AOC system as well as its entities, environment, system objective function, and interactions. Based on our definitions, we can readily define entities in specific applications to be homogeneous or heterogeneous in terms of their similar/different goals, behaviors, behavioral rules, interactions etc.. As we have seen that in the AOC-based feature extraction (in Section V) and global optimization (in Section VII), entities are homogeneous, while in the AOC-based WWW regularity characterization (in Section VI) the entities can be regarded as heterogeneous.

In the presented AOC-based examples, autonomous entities are at the same level. In order to be consistent with those examples, our definitions in the previous sections do not mention the hierarchy issue of entities. However, in an AOC system, entities can actually be at different levels. In other words, entities may be organized in a hierarchical manner. Entities at different levels have different goals, behaviors, behavioral rules etc. Their interactions with others may have different intensities according to specific requirements. Entities can also be statically or dynamically formed into different groups. The entities in the same group have common goals and behavioral rules, and behave as a whole to interact with other groups or individual entities. Due to space limitation, in this paper, we will not extend our discussions to this case.

D. System Behaviors and Autonomy

Complex systems modeling using a bottom-up approach centers around local behaviors and behavioral rules of autonomous entities and complex behaviors of the whole system. The trickiest part of all modeling tasks is finding the relationship between these two kinds of behaviors. AOC adds a new dimension to the modeling process, *i.e.*, modeling and deploying autonomy-based systems. Broadly speaking, autonomy is a special attribute of entities in a complex system and is the building block of an AOC algorithm. This subsection attempts to emphasize two typical system behaviors based on the previous formulation of AOC components. The notion of autonomy in the context of a computational system is re-visited in the following paragraphs.

1) System Behaviors: Through local behaviors of its entities as well as its internal interactions, an AOC system can exhibit two kinds complex behaviors, namely *emergent behavior* and *emergent purposeful behavior*.

Definition 14 *Emergent behavior is some system behavior not inherent in individual entities. It can only result from the interactions of individual entities. Such behavior cannot be defined simply by the attributes of the entities concerned.*

Definition 15 *Emergent purposeful behavior is a goal-directed emergent behavior that is not achievable purely at the individual entity level.*

Take the AOC-based EDO as an example. As we have seen from Figure 9, in the first 80 generations, the number of entities increases rapidly so as to entities abound in the range between ± 100 in both axes, while in the next 20 generations the number of entities decreases drastically. This is an example of emergent behavior. On the other hand, (1) between generation 10 and 30, more and more entities gather around the origin, and (2) between generation 10 and 80, lots of entities cluster along the axes. Through these two emergent behaviors, the EDO system explores the optimal or suboptimal solution region. Therefore, they are emergent purposeful behaviors.

Individual entities have their local behaviors. Whether or not they are identical depends on the system concerned. If the entities of a complex system are able to adapt, the local behavior of each entity is bound to be different over time. As a result, the system behavior may also be changed. The worth noting is the fact that emergent behavior and emergent purposeful behavior may not arise just from the interactions between the basic elements but from the interactions between subsystems.

2) Autonomy Defined: Autonomy in AOC can be defined at two levels, i.e., entity level and system level.

a) Autonomy at Entity Level: The *autonomy* of an entity implies freedom from the control by others with respect to local or internal affairs, i.e., behaviors and decision making. In the context of Artificial Intelligence, autonomy has been one of the key notions in many research fields, including Intelligent Agents and Artificial Life [99], [100]. Autonomy may be defined as follows.

Definition 16 *Autonomy is an attribute of a self-governed, self-determined, and self-directed entity with respect to its own behaviors and internal states, free from the explicit control of another entity.*

This view of autonomy is endogenous. 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. However, only direct perturbation is prohibited; indirect influence is allowed and encouraged. The essence is that each entity is able to make decisions for itself, subject to the limitations of the available information. As we have noted from the previous examples, although all entities can interact directly or indirectly with their neighbors and/or environment, there are no any 'commands' imposed to the entities.

Definition 17 *Synthetic autonomy is an abstracted equivalent of the desired observable attribute of an entity in a natural complex system and is the fundamental building block of an AOC system.*

Any computational system having synthetic autonomy exhibits its emergent or emergent purposeful behavior. Similar to the messages that are passed between objects in an object-oriented program, the direct or indirect interactions are the 'glue' that helps to put the building blocks together to form a coherent system.

b) Autonomy at System Level:

Definition 18 *Emergent autonomy is the system-wide, externally observable, self-induced attribute of an AOC system that is built using entities having synthetic autonomy as the basic building block.*

It is not difficult to see that a computational system can be derived from many levels of abstraction. If a human society is to be modeled as a computational system, abstraction can possibly occur at the level of population, individual, biological system, cell, molecule, and atom. Note that autonomy according to Definition 16 is present at all these levels. Moreover, the emergent autonomy obtained at, say, the cell level, is the foundation for the emergent autonomy at the biological system level.

This multi-level view of autonomy encompasses Brooks' subsumption architecture [101] in that complex behaviors can be built up from multiple levels of simpler, and relatively more primitive, behavior.

Based on the above definitions, we can summarize AOC as follows:

Definition 19 *Autonomy Oriented Computing (AOC) is the study of bottom-up approaches to construct autonomous computational systems using entities (or computational units) that exhibit synthetic autonomy individually and the resultant system exhibits emergent autonomy through the self-organization of the constituents.*

E. Self-Organization in Autonomy Oriented Computing

Why AOC systems can successfully emulate human collective behaviors (e.g., the surfing behaviors of users in the AOC-based regularity characterization) and solve some hard computational problems (e.g., the AOC-based feature search and extraction, the AOC-based global optimization)? The key point is the self-organization of entities in AOC systems. All complex collective behaviors in AOC systems originate from the self-organization of their autonomous entities.

In order to achieve their respective goals, individual entities autonomously make decisions on selecting and performing their simple local behaviors. While selecting or performing, they need to consider not only their own state information, but also that of their neighbors and/or their environment. To do so, they will either directly interact with each other, or indirectly interact via their environment, to exchange their information. By performing behaviors, entities change their states towards their respective goals. Because autonomous entities take into account of others while behaving, from a global view, entities are aggregated together to achieve the global goal of AOC systems. In other words, the final collective behaviors are aggregation results of individual entities' behaviors. Figure 13 shows a schematic diagram of the process of self-organization. In general, we can define the process of self-organization as follows.

Definition 20 *The process of self-organization of entities $\{\mathbf{e}_i\}$ in an AOC system is a sequence(s) of state transitions $\big\{\{S_t^{e_i}\}|t = 0, \cdots, T\big\}$, which is subject to the following two constraints*[1]*:*

1) Locally, for each entity $\mathbf{e}_i$,

$$Pr\Big(\mathcal{F}(S_{t+1}^{e_i}) - \mathcal{F}(S_t^{e_i}) \succ 0\Big) \geq 0, \tag{28}$$

[1]The number of entities, i.e., $|\{\mathbf{e}_i\}|$, during the process of self-organization may vary over time.

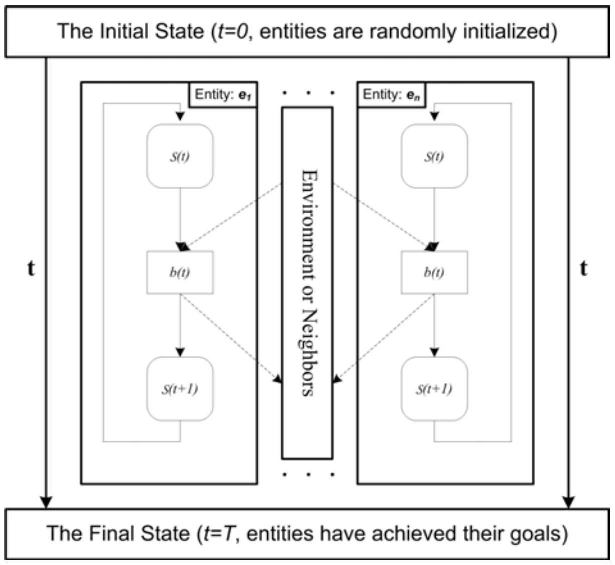

The Initial State (t=0, entities are randomly initialized)
Entity: e1
Entity: en
S(t)
b(t)
S(t+1)
Environment or Neighbors
t
t
The Final State (t=T, entities have achieved their goals)

Fig. 13. The process of self-organization in an AOC system.

where $\mathcal{F}(S_t^{e_i})$ returns the evaluation value of entity $\mathbf{e}_i$'s state at time t; $Pr(\cdot)$ returns a probability; $\mathcal{F}(S_{t+1}^{e_i}) - \mathcal{F}(S_t^{e_i}) \succ 0$, i.e., $\mathcal{F}(S_{t+1}^{e_i}) \succ \mathcal{F}(S_t^{e_i})$, denotes that the new state of entity $\mathbf{e}_i$ at time $t+1$ is better than the one at time t. Here, the specific meaning of 'better' depends on the evaluation function $\mathcal{F}$.

2) *Globally, for the whole system,*

$$Pr\Big(\Delta\mathbf{\Phi}_t \succ 0\Big) \geq 0, \tag{29}$$

where $\Delta\mathbf{\Phi}_t = \mathbf{\Phi}_t - \mathbf{\Phi}_{t-1}$ denotes the change in the value vectors of system objective function $\mathbf{\Phi}$ at time $t+1$ and t, respectively.

Given the above local and global constraints, the AOC system finally reaches a state where

$$\mathbf{\Phi}_T = Opt(\mathbf{\Phi}_t). \tag{30}$$

where $Opt(\cdot)$ returns the optimal value.

In the above definition, the local constraint (Equation 28) indicates that autonomous entities probabilistically evolve to better states at each step, although for some entities this point may not hold. The global constraint (Equation 29) denotes that the whole AOC system also probabilistically evolves to a better 'state' in terms of its system objective function $\mathbf{\Phi}$ at each step. We should point out that due to the nonlinear interactions in the AOC system, the process of self-organization exhibit nonlinear characteristics in the changes of $\Delta\mathbf{\Phi}_t$ as well as the evaluation of a system objective function over time. Equation 30 denotes that the whole AOC system finally reaches a global optimal state.

Let us take the AOC-based feature search and extraction as an example to demonstrate the process of self-organization. In this example, an entity tries to find a pixel that belongs to a certain homogeneous region. At such a pixel, its evaluation value will be better than those at other pixels. On one hand, when an entity finds a desired pixel, it will reproduce some offspring within its local environment, where the offspring will most probably find other desired pixels. This mechanism acts as positive feedback, through which the AOC system accumulates and amplifies the successful behavioral results of entities. On the other hand, if an entity cannot find a desired pixel after predefined steps, i.e., its lifespan, it will be deactivated. Through this negative feedback mechanism, the AOC system eliminates those entities with poor performance. From a global point of view, we can note that at the beginning steps, few entities can successfully find the desired homogeneous region. As the search progresses, more entities which are either reproduced or diffusing are able to locate pixels of a homogeneous region that has been found. This nonlinear process will continue until it achieves a state where all active entities stay in a certain homogeneous region. Hence, the whole AOC system is globally optimized.

Based on the previous definitions of an AOC system, entities, environments etc., we can formulate AOC shown in Section III as follows.

Definition 21 *Autonomy Oriented Computing (AOC) is a process involving the following steps:*

Step 1. Initially, design an AOC system $\langle\{\mathbf{e}_1, \cdots, \mathbf{e}_N\}, \mathbf{E}, \mathbf{\Phi}\rangle$*, in particular,*

- *For the environment, design its internal states* $\mathcal{ES}$*;*
- *For each entity, design its internal states* $\mathcal{S}$*, evaluation function* $\mathcal{F}$*, goals* $\mathcal{G}$*, behavior set* $\mathcal{B}$*, and behavioral rule set* $\mathcal{R}$*;*
- *Design the system objective function* $\mathbf{\Phi}$*;*

Step 2. Accurately or approximately determine the desired 'value' $\mathbf{\Phi}^*$ *of the system objective function* $\mathbf{\Phi}$*;*

Step 3. Perform the designed AOC system, and then obtain the final 'value' $\mathbf{\Phi}^{'}$*;*

Step 4. Define an optimization function $\Psi = |\mathbf{\Phi}^{'} - \mathbf{\Phi}^*|$ *as the guideline of the autonomy oriented computing;*

Step 5. If Ψ *is not optimized, modify or fine-tune parameters of entities and/or the environment;*

Step 6. Repeat the above steps until Ψ *is optimized.*

F. AOC-Based Formulations of Three Examples

According to the above discussed framework of AOC systems, we can formulate the examples illustrated in Sections V, VI, and VII as follows.

1) Feature Search and Extraction: As we have seen in Section V, in this example, autonomous entities are deployed on a 2-D representation of an image to locate homogeneous regions. They diffuse from their initial positions so as to explore the whole image. If they have found a certain homogeneous region, they will breed some offspring on the spot, which will continue the exploration task. Finally, the regions satisfying the assessing criterion of entities are labeled.

- The image serves as the **environment** of autonomous entities. Specifically, it is characterized by two attributes, namely, the grey-scale intensity of pixels and the labels of pixels indicating whether or not they belong to certain homogeneous regions. The former is static while the latter is dynamic, which is changed as entities explore more and more pixels.

- The following is the formulation of an **entity** in this example:
  - The **state** of an entity is characterized by four attributes: position, age, activity, and lifespan, where lifespan is a predefined constant, other three are dynamically changed;
  - The **goal** of an entity is to explore an image and find a certain homogeneous region;
  - The **evaluation function** is to judge whether or not a pixel, where the entity stays currently, belongs to a certain homogeneous region based on the relative contrast, regional mean, and region standard deviation of the grey-level intensity;
  - Each entity has four **behaviors**, namely, breeding, pixel labeling, diffusion, and decay. In more detail, breeding behavior replicates entities several times; pixel labeling behavior changes the state of the environment, i.e., the labels of pixels; diffusion behavior changes the positions of entities; and decay behavior adds the ages of entities by one. These behaviors are separated into two groups, i.e., (1) breeding and pixel labeling, and (2) diffusion and decay;
  - The **behavioral rule** is to select a group of behaviors based on the evaluation function at each step.
- The **interactions** in this example are embodied in two aspects: (1) an entity senses the grey-scale intensity information in its neighboring region; (2) through the pixel labeling behavior, an entity changes the label of the pixel, where it resides currently. Note that there are no direct interactions among entities.

2) WWW Regularity Characterization: In this example (see Section V), information foraging entities surf in the artificial Web to find the contents which they are interested in. At each step, they will judge whether or not they have collected enough contents. If so, they will leave the Web. Otherwise, they will continue to surf until they have been satisfied or completely lost their interests to surf.

- The **environment** is the artificial Web space, where information foraging entities (i.e., users) inhabit. The environment records the content information of all websites/pages.
- An **entity** is an information foraging agent:
  - The **state** of an entity is characterized by its interests, position, motivation, and reward, all of which are dynamically changed over time.
  - The support function acts as the **evaluation function** of an entity. It is used to judge (1) whether or not an entity has found enough information, or (2) whether or not an entity is completely dissatisfied with the surfing based on its state information.
  - The **goal** of an entity is to find enough contents satisfying its interests from the Web. Technically, an entity will try to achieve two goals, i.e., (1) the maximum support value or (2) the minimum support value.
  - The **behavior** in this example is moving from one website/page to another. In doing so, the entity may employ different strategies to select the next website/page. According to the differences in their strategies, entities are classified into three types: *rational* users, *recurrent* users, and *random* users.

- That the entity senses the content information of a website/page embodies the **interactions** between entities and their environment in this example.

3) Global Optimization: In EDO, entities reside and further explore a landscape corresponding to the search space of a given problem so as to find positions where the problem takes its maximum values.

- The **environment** in EDO is the search space specified by the given global optimization problem. The environment records an possible solution of the given problem for each position. An entity stays at a certain position means it assigns the corresponding possible solution to the problem.
- An entity can be formulated as follows:
  - The **state** of an entity in this example records the position (i.e., object vector), age, lifespan, probability matrix, activity, etc. information.
  - The **goal** of each entity is to explore the environment and find a position, where the corresponding possible solution is the exact solution that optimizes the given problem.
  - The **evaluation function** is the fitness function, to evaluate whether or not the position, where the entity stays currently, is a good one.
  - An entity possesses the following **behaviors**: diffuse (i.e., rational move and random walk), reproduce, decay (i.e., age), and rejuvenate.
- In this example, two kinds of **interactions** are embodied: the probability matrix sharing between entities and their offspring is the direct interactions among entities, while sensing the possible solution recorded in each position is the interactions between entities and their environment.

IX. SUMMARY

In this article, we have identified and described the common formulations and characteristics of *Autonomy Oriented Computing* (AOC). The usage of AOC is vast as judged from the examples described above. AOC has three general approaches with different goals. AOC-by-fabrication is similar to construction with a blueprint where a known phenomenon is abstracted and replicated as a computational problem solving technique. AOC-by-prototyping presents a trial-and-error approach to finding explanations to some observations via an autonomy oriented system. Human involvement is required 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.

As can be noted, a crucial step in applying the AOC approaches lies in identifying a computational model of synthetic autonomy. The identification of such a model will rely on a good understanding of the problem at hand as well as an appropriate analogy from its natural counterpart. If the working mechanism behind the natural system is known, we can build a direct mapping, where certain special states of the system correspond to the solutions of the problem. On the other hand, if the working mechanism is not completely clear, we need to gradually modify the analogy through a manual (in the case of AOC-by-prototyping) or an automated (in the case of AOC-by-self-discovery) trial-and-error process of fine-tuning the mapping, until the working mechanism is uncovered or the problem is solved. The examples presented in this article have shown that the step of finding an appropriate analogy is realizable and the resulting AOC approaches are useful.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their constructive comments, which have helped to improve the paper. We would also like to acknowledge the experimentation assistance by Yichuan Cao, Huijian Zhou, and Shiwu Zhang in some of the research projects mentioned in the article. The present project has been in part supported by HKBU FRG and HKRGC grants.

REFERENCES

[1] S. Kauffman, *At Home in the Universe: the Search for Laws of Complexity*. Oxford University Press, 1996.

[2] S. Rihani, *Complex Systems Theory and Development Practice: Understanding Non-linear Realities*. Zed Books, 2002.

[3] Y. Louzoun, S. Solomon, H. Atlan, and I. R. Cohen, "The emergence of spatial complexity in the immune system," 2000, Los Alamos Physics Archive arXiv:cond-mat/0008133 (http://xxx.lanl.gov/html/cond-mat/0008133).

[4] B. Gurdip and Z. Chen, "Stock valuation in dynamic economies," 2001, university of Maryland. [Online]. Available: http://citeseer.ist.psu.edu/article/bakshi01stock.html

[5] L. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, 1966.

[6] H.-P. Schwefel, *Numerical Optimization of Computer Models*. John Wiley & Sons, 1981.

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

[8] 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*, vol. 26, no. 1, pp. 1–13, 1996.

[9] K. Vajravelu, Ed., *Differential Equations and Nonlinear Mechanics*. Kluwer Academic Publishers, 2001.

[10] A. M. Blokhin, *Differential Equations & Mathematical Modelling*. Nova Science Publishers, June 2002.

[11] J. H. Vandermeer and D. E. Goldberg, *Population Ecology: First Principles*. Princeton University Press, March 2003.

[12] J. Casti, *Would-Be Worlds: How Simulation is Changing the Frontiers of Science*. John Wiley & Son, 1997.

[13] C. G. Langton, "Artificial life," in *Artificial Life*, ser. Volume VI of SFI Studies in the Sciences of Complexity, C. G. Langton, Ed. Addison-Wesley, 1989, pp. 1–44.

[14] M. Resnick, *Turtles, Termites and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1994.

[15] J. Doran and N. Gilbert, "Simulating societies: An introduction," in *Simulating Societies: The Computer Simulation of Social Phenomena*, N. Gilbert and J. Doran, Eds. UCL Press, 1994, ch. 1, pp. 1–18.

[16] M. Mitchell, J. P. Crutchfield, and P. T. Hraber, "Dynamics, computation, and the "edge of chaos": A re-examination," in *Complexity: Metaphors, Models, and Reality*, G. A. Cowan, D. Pines, and D. Meltzer, Eds. Addison-Wesley, 1994, pp. 497–513.

[17] J. P. Crutchfield and M. Mitchell, "The evolution of emergent computation," in *Proceedings of the National Academy of Sciences*, vol. 92, no. 23, 1995, pp. 10 742–10 746.

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

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

[20] E. H. Durfee, "Distributed problem solving and planning," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. MIT Press, 1999, pp. 121–164.

[21] T. W. Sandholm, "Distributed rational decision making," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. MIT Press, 1999, pp. 201–258.

[22] G. Wagner, Y. Lesperance, and E. Yu, Eds., *Agent-Oriented Information Systems 2000*. iCue Publishing, June 2000.

[23] G. Wagner, K. Karlapalem, Y. Lesperance, and E. Yu, Eds., *Agent-Oriented Information Systems 2001*. iCue Publishing, May/June 2001.

[24] H. S. Nwana, D. T. Ndumu, and L. C. Lee, “ZEUS: An advanced tool-kit for engineering distributed multi-agent systems,” in *Proceedings of the Third International Conference on Practical Application of Intelligent Agents and Multi-Agents (PAAM’98)*, March 1998, pp. 377–391.

[25] T. Hogg and B. A. Huberman, “Better than the best: The power of cooperation,” in *1992 Lectures in Complex Systems*, ser. SFI Studies in the Sciences of Complexity, L. Nadel and D. Stein, Eds. Addison-Wesley, 1993, vol. V, pp. 165–184.

[26] B. A. Huberman, Ed., *The Ecology of Computation*. North-Holland, 1988.

[27] M. Yokoo, *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems*, ser. Springer Series on Agent Technology. Springer, 2001.

[28] 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*, vol. 10, no. 5, pp. 673–685, 1998.

[29] M. Yokoo and K. Hirayama, “Algorithms for distributed constraint satisfaction: A review,” *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 2, pp. 185–207, June 2000.

[30] C. G. Langton, “Preface,” in *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Addison-Wesley, 1992, pp. xiii–xviii.

[31] K. Sims, “Artificial evolution for computer graphics,” *Computer Graphics*, vol. 25, no. 4, pp. 319–328, 1991.

[32] P. Prusinkiewicz, M. Hammel, and R. Mech, “Visual models of morphogenesis: A guided tour,” May 1997. [Online]. Available: http://www.cpsc.ucalgary.ca/Research/bmv/vmm-deluxe/TitlePage.html

[33] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. Springer, 1990.

[34] T. S. Ray, “An approach to the synthesis of life,” in *Artificial Life II*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Addison-Wesley, 1992, pp. 371–408.

[35] J. R. Koza, *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[36] 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 *Behavioural Mechanisms of Food Selection*, ser. NATO-ASI Series, R. N. Hughes, Ed. Springer, 1990, vol. G 20.

[37] F. Corno, M. S. Reorda, and G. Squillero, “The selfish gene algorithm: A new evolutionary optimization strategy,” in *Proceedings of the Thirteenth Annual ACM Symposium on Applied Computing (SAC’98)*, February 1998, pp. 349–355.

[38] R. G. Reynolds, “An introduction to cultural algorithms,” in *Proceedings of the Third Annual Conference on Evolutionary Programming*, A. V. Sebald and L. J. Fogel, Eds. World Scientific, 1994, pp. 131–139.

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

[40] J. Liu, *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation*. World Scientific, 2001.

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

[42] I. Pitas, *Digital Image Processing Algorithms*. Prentice Hall, 1993.

[43] J. Liu, Y. Y. Tang, and Y. C. Cao, “An evolutionary autonomous agents approach to image feature extraction,” *IEEE Transaction on Evolutionary Computation*, vol. 1, no. 2, pp. 141–158, July 1997.

[44] J. Liu and Y. Y. Tang, “Adaptive image segmentation with distributed behavior-based agents,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 544–551, June 1999.

[45] R. Conte and N. Gilbert, “Introduction: Computer simulation for social theory,” in *Artificial Societies*, N. Gilbert and R. Conte, Eds. UCL Press, 1995, ch. 1, pp. 1–15.

[46] D. Helbing and B. A. Huberman, “Coherent movng states in highway traffic,” *Nature*, vol. 396, pp. 738–740, December 1998.

[47] S. Rasmussen and C. Barrett, “Elements of a theory of simulation,” Santa Fe Institute, Tech. Rep. 95-04-040, 1995.

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

[49] G. K. Still, “Crowd dynamics,” Ph.D. dissertation, Mathematics Department, Warwick University, August 2000.

[50] D. Helbing, I. Farkas, and T. Vicsek, “Simulating dynamic features of escape panic,” *Nature*, vol. 407, pp. 487–490, September 2000.

[51] G. K. Zipf, *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

[52] B. A. Huberman, P. L. Pirolli, J. E. Pitkow, and R. M. Lukose, “Strong regularities in World Wide Web surfing,” *Science*, vol. 280, pp. 96–97, April 3, 1997.

[53] R. M. Lukose and B. A. Huberman, “Surfing as an real option,” in *Proceedings of the International Conference on Information and Computation Economics*, October 1998, pp. 45–51.

[54] M. Levene and G. Loizou, “Computing the entropy of user navigation in the Web,” Department of Computer Science, University College London, Research Note RN/99/42, 1999.

[55] M. Levene, J. Borges, and G. Loizou, “Zipf’s law for Web surfers,” *Knowledge and Information Systems*, vol. 3, pp. 120–129, 2001.

[56] A. Thatcher, “Determining interests and motives in WWW navigation,” in *Proceedings of the Second International Cyberspace Conference on Ergonomics (CybErg1999)*, 1999.

[57] J. Liu, S. Zhang, and J. Yang, “Characterizing Web usage regularities with information foraging agents,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 5, pp. 566–584, 2004.

[58] J. Liu and S. Zhang, “Unveiling the origin of Web surfing regularities,” in *Proceedings of iNET 2001*, 2001.

[59] B. A. Huberman and L. A. Adamic, “Growth dynamics of the world-wide web,” *Nature*, vol. 410, p. 131, September 9, 1999.

[60] S.-H. Yeung, “An agent-based model for www characterization,” Master’s thesis, Department of Computer Science, Hong Kong Baptist University, Hong Kong, August 2001.

[61] D. Kaplan and L. Glass, *Understanding Nonlinear Dynamics*. Springer, 1995.

[62] A. J. Angeline, “Adaptive and self-adaptive evolutionary computations,” in *Computation Intelligence: A Dynamic Systems Perspective*, M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, and T. Fukuda, Eds. IEEE Press, 1995, pp. 152–163.

[63] T. Bäck, “Self-adaptation,” in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing and Oxford University Press, 1997, pp. C7.1:1–15.

[64] J. J. Grefenstette, “Optimization of control parameters for genetic algorithms,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[65] R. Hinterding, Z. Michalewicz, and A. E. Eiben, “Adaptation in evolutionary computation: A survey,” in *Proceedings of the Fourth IEEE Conference on Evolutionary Computation*, T. Bäck, Z. Michalewicz, and X. Yao, Eds. IEEE Press, 1997, pp. 65–69.

[66] T. Bäck, “The interaction of mutation rate, selection, and self adaptation within a genetic algorithm,” in *Parallel Problem Solving From Nature - PPSN II*. Elsevier, 1992, pp. 85–94.

[67] T. Bäck and M. Schutz, “Intelligent mutation rate control in canonical genetic algorithms,” in *Proceedings of the International Syposium on Methodologies for Intelligent Systems*, 1996, pp. 158–167.

[68] H. R. Gzickman and K. P. Sycara, “Self-adaptation of mutation rates and dynamic fitness,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eigth Innovative Applications of Artificial Intelligence Conference*, vol. 2. MIT Press, August 1996, p. 1389.

[69] M. Sebag and M. Schoenauer, “Mutation by imitation in bollean evolution strategies,” in *Parallel Problem Solving from Nature - PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Springer, September 1996, pp. 356–365.

[70] E. A. Williams and W. A. Crossley, “Empirically derived population size and mutation rate guidelines for a genetic algorithm with uniform crossover,” in *Proceedings of the Second Online World Conference on Soft Computing in Engineering Design and Manufacture (WSC2)*, June 1997.

[71] K.-H. Liang, X. Yao, and C. Newton, “Dynamic control of adaptive parameters in evolutionary programming,” in *Proceedings of Simulated Evolution and Learning (SEAL98): Second Asia Pacific Conference*. Springer, 1998, pp. 42–49.

[72] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed. Springer, 1994.

[73] A. K. Swain and A. S. Morris, “A novel hybrid evolutionary programming method for function optimization,” in *Proceedings of Congress on Evolutionary Computation (CEC2000)*, 2000, pp. 1369–1376.

[74] X. Yao, Y. Liu, and G. Lin, “Evoltutionary programming made faster,” *IEEE Transaction on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[75] E.-J. Ko and O. N. Garcia, “Adaptive control of crossover rate in genetic programming,” in *Proceedings of the Articial Neural Networks in Engineering (ANNIE’95)*. ASME Press, 1995.

[76] J. E. Smith and T. C. Fogarty, “Adaptive parameterised evolutionary systems: Self adaptive recombination and mutation in genetic algorithm,” in *Parallel Problem Solving from Nature - PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Springer, September 1996, pp. 441–450.

[77] P. J. Angeline, “Two self-adaptive crossover operators for genetic programming,” in *Advances in Genetic Programming*, P. J. Angeline and K. E. Kinnear, Jr. , Eds. MIT Press, 1996, pp. 89–110.

[78] M.-S. Ko, T.-W. Kang, and C.-S. Hwang, “Adaptive crossover operator based on locality and convergence,” in *Proceedings of the 1996 IEEE International Conference on Intelligence and Systems*. IEEE Computer Society Press, November 1996, pp. 18–22.

[79] B. Freisleben, “Metaevolutionary approaches,” in *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing and Oxford University Press, 1997, pp. C7.2:1–8.

[80] T. Bäck, “Parallel optimization of evolutionary algorithms,” in *Parellel Problem Solving from Nature - PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Springer, 1994, pp. 418–427.

[81] K. A. DeJong, “Analysis of the behavior of a class of genetic adaptive systems,” Ph.D. dissertation, Department of Computer and Communication Sciences, University of Michigan, 1975.

[82] B. Freisleben and M. Härtfelder, “In search of the best genetic algorithm for the traveling salesman problem,” in *Proceedings of the Ninth International Conference on Control Systems and Computer Science*, 1993, pp. 485–493.

[83] ——, “Optimization of genetic algorithms by genetic algorithms,” in *Artificial Neural Networks and Genetic Algorithms*, R. F. Albrecht, C. R. Reeves, and N. C. Steele, Eds. Springer, 1993, pp. 392–399.

[84] C.-J. Chung and R. G. Reynolds, “Knowledge-based self-adaptation in evolutionary search,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 1, pp. 19–33, 2000.

[85] F. Herrera and M. Lozano, “Adaptive genetic operators based on coevolution with fuzzy behaviors,” Department of Computer Science and Artificial Intelligence, University of Granada, Tech. Rep. DECSAI-98-01-05, March 1998.

[86] M. A. Lee and H. Takagi, “Dynamic control of genetic algorithms using fuzzy logic techniques,” in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed. Morgan Kaufmann, 1993, pp. 76–83.

[87] A. Tettamanzi, “Evolutionary algorithms and fuzzy logic: A two-way integration,” in *Proceedings of the Second Joint Conference on Information Sciences*, 1995, pp. 464–467.

[88] 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.

[89] S. Chiusano, F. Corno, P. Prinetto, and M. S. Reorda, “Cellular automata for sequential test pattern generation,” in *Proceedings of the Fifteenth IEEE VLSI Test Symposium (VTS97)*, April, 1997, pp. 60–65.

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

[91] A. Torn and A. Zilinskas, *Global Optimization*. Springer, 1989.

[92] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*. Springer, 1990.

[93] R. Horst and P. M. Pardalos, Eds., *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.

[94] J. Mockus, *Bayesian Approach to Global Optimization : Theory and Applications*. Kluwer Academic Publishers, 1989.

[95] K. C. Tsui and J. Liu, “Evolutionary multi-agent diffusion approach to optimization,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 16, no. 6, pp. 715–733, 2002.

[96] ——, “Evolutionary diffusion optimization, Part I: Description of the algorithm,” in *Proceedings of the Congress on Evolutionary Computation (CEC 2002)*, X. Yao, Ed., 2002, pp. 169–174.

[97] ——, “Evolutionary diffusion optimization, Part II: Performance assessment,” in *Proceedings of the Congress on Evolutionary Computation (CEC 2002)*, X. Yao, Ed., 2002, pp. 1284–1290.

[98] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[99] N. R. Jennings and M. Wooldridge, “Software agents,” *IEE Review*, vol. 42, no. 1, pp. 17–21, January 1996.

[100] P. Maes, “Modeling adaptive autonomous agents,” in *Artificial Life: An Overview*, C. G. Langton, Ed. MIT Press, 1995, pp. 135–162.

[101] R. A. Brooks, “Intelligence without representation,” *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

Jiming Liu is Professor and Head of Department of Computer Science at Hong Kong Baptist University (HKBU). He leads Centre for e-Transformation Research (CTR), a government-funded centre focusing on basic and applied research in the areas of Web Intelligence, Autonomy Oriented Computing, Data Mining, and Grid Computing. He directs the research activities at AAMAS/AOC Research Lab (i.e., Autonomous Agents and Multi-Agent Systems / Autonomy Oriented Computing) at HKBU. Prof. Liu holds a Bachelor of Science degree in Physics from East China Normal University in Shanghai, a Master of Arts (Educational Technology) from Concordia University in Montreal, and a Master of Engineering and a Doctor of Philosophy in Electrical and Computer Engineering from McGill University. Prof. Liu's past research involved investigating and developing: Autonomy Oriented Computing (AOC), Intelligent Agents and Multi-Agent Systems, and Web Intelligence (WI). Prof. Liu has published over 160 scientific articles in refereed international journals, books, and conferences. In addition, he has published 21 books. Among them, 7 are monograph books. Prof. Liu is the Editor-in-Chief of Web Intelligence and Agent Systems (IOS Press, The Netherlands), Annual Review of Intelligent Informatics (World Scientific Publishing), and The IEEE Computational Intelligence Bulletin (IEEE Computer Society TCCI). He is Associate Editor of Knowledge and Information Systems: An International Journal, Springer, and International Journal of Web Services Research, Idea Group. He has been Guest Editor of IEEE Computer, Special Issue on Web Intelligence and the special issues of several other international journals, such as Computational Intelligence, Knowledge-Based Systems, Journal of Intelligent Information Systems, and Cognitive Systems Research. Previously, he served as Guest Editors for Special Issue on Intelligent Agents in E-Commerce, Electronic Commerce Research Journal and Special Issue on Agent Technology, International Journal of Pattern Recognition and Artificial Intelligence. Prof. Liu is the co-founder of Web Intelligence Consortium (WIC), an international organization dedicated to promoting world-wide scientific research and industrial development in the era of Web and agent Intelligence. He has founded and served as Program Chair, Conference Chair, Workshop Chair, and General Chair for several international conferences and workshops, including The IEEE/WIC/ACM International Conference on Web Intelligence (WI) series and The IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT) series, and has served as the Senior Program Committee Member, Program Committee Member, and Steering/Planning Committee Member for many major international conferences.

Xiaolong Jin received the B.Sc. degree from the Department of Applied Mathematics, Beijing University of Aeronautics and Astronautics in July, 1998 and the M.Eng. from Academy of Mathematics and System Sciences, Chinese Academy of Sciences in July, 2001. He is currently pursuing the Ph.D. degree in computer science at Hong Kong Baptist University. His current research interests include autonomous agents & multi-agent systems, autonomy oriented computing, Grid computing, peer-to-peer computing, distributed problem solving, satisfiability problems, constraint satisfaction problems, etc.

Kwok Ching Tsui is an assistant Professor in the Department of Computer Science at Hong Kong Baptist University. He is actively involved in research on agent technology, nature-inspired computation, information engineering on the Internet and E-commerce.