Toward Nature Inspired Computing[1]

By **Jiming Liu** and **K.C. Tsui**

In this article, we take a look at an emerging computing paradigm called *Nature Inspired Computing* (NIC). We examine the impacts of NIC in two aspects. First, NIC enables us to explain the underlying mechanism of a real-world complex system by formulating computing models and testing hypotheses through controlled experimentation. The end product of such computing experiments is a deep understanding or a new discovery of the real working mechanism of the modeled system. Second, NIC enables us to embody autonomous (e.g., life-like) behavior in solving computing problems. With detailed knowledge of the underlying mechanism, abstracted autonomous behavior can be used as a model for a general-purpose problem-solving strategy or method.

Formulation and Characteristics of NIC

Generally speaking, the *objectives* of NIC are twofold: (1) characterizing and understanding complex phenomena or systems behavior, and (2) designing and developing computing solutions to hard problems. Neither of these objectives can be achieved without formulating a model of the factors underlying a complex system. The *modeling* process can be started with a theoretical analysis from a macroscopic or microscopic view of the system. Alternatively, a blackbox or whitebox approach may be adopted. Blackbox approaches such as Markov models or artificial neural networks normally do not tell us much about the working mechanism. On the other hand, whitebox approaches such as agents with bounded rationality are more useful for explaining behavior [10].

NIC Formulation. The essence of NIC formulation lies in the conception of a computing system that is operated by population(s) of *autonomous entities*. The rest of the system is referred to as the environment. An autonomous entity consists of a detector (or a set of detectors), an effector (or again, a set of effectors), and a repository of local behavior rules (see Figure 1) [5][8].

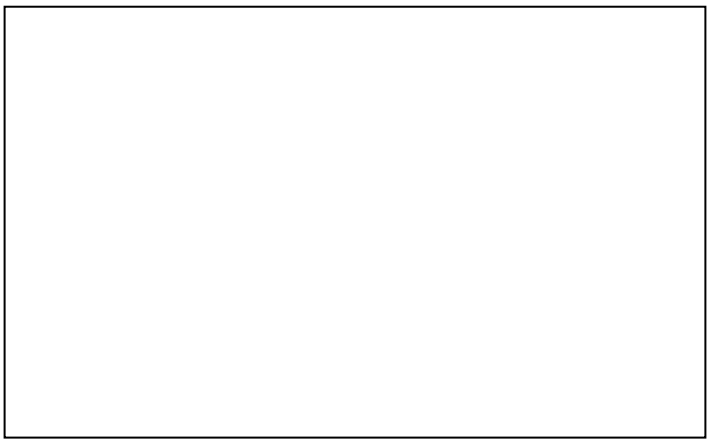

Figure 1. Modeling an autonomous entity in *Nature Inspired Computing* (NIC).

A detector receives information related to its neighbors and the environment. For example, in the simulation of a flock of birds, this information includes the speed and direction in which they are heading and the distance between the entities in question. Details of the content and format of this information need to be defined according to the system to be modeled or the problem to be solved. The notion of neighbors may be defined in terms of position (e.g., the bird in front, to the left, and to the right), distance (e.g., a radial distance of two grids), or both (e.g., the birds up to two grids in front).

Environmental information conveys the status of a certain feature that is of interest to an autonomous entity. The environment can also help carry sharable local knowledge. The effector of an autonomous entity refers collectively to the device for expressing actions. These actions can be either changes to an internal state, an external display of certain behavior, or changes to the environment in which the entity inhabits. An important role of the effector, as part of the local behavior model, is to facilitate implicit information sharing among autonomous entities. Central to an autonomous entity are the *behavior rules* that govern how it should act or react to the information collected by the detector from the environment and its neighbors. These rules decide into what state this entity should change and also what local knowledge should be released via the effector to the environment. An example of sharable local knowledge is the role of pheromones in an ant colony. This information is untargeted and the communication via the environment is undirected; any ant can pick up the information and react according to its own behavior model.

In order to adapt itself to a problem without being explicitly told in advance, an autonomous entity needs to modify its behavior rules over time. This is the *learning* capability of the individual. It is worth noting that *randomness* plays a part in the decision-making process of an autonomous entity, despite the presence of a rule set. This is to allow an autonomous entity to explore uncharted territories even in the presence of mounting evidence that it should exploit a certain path. On the other hand, randomness also helps the entity resolve conflict in the presence of equal support for suggestions to act in different manners, and avoid getting stuck in local optima.

The environment acts as the domain in which autonomous entities roam. This is a static view of the environment. The environment of an NIC system can also act as the "noticeboard" where the autonomous entities can post and read local information. In this dynamic view, the environment is changing all the time. For example, in the N-queen constraint satisfaction problem [7], the environment

can tell a particular queen on a chessboard how many constraints are violated in her neighborhood after a move is made. This, in effect, translates a global goal into a local goal for individual entities. The environment also keeps the central clock that helps synchronize the actions of all autonomous entities, if necessary.

Characteristics of NIC. Before we describe examples of NIC in either characterizing complex behavior or solving computing problems, let us first highlight the important ideas as well as *common characteristics* of NIC (i.e., A*utonomous,* D*istributed,* E*mergent,* A*daptive,* and S*elf-organized*; or *ADEAS* for short) [8].

Autonomous. In NIC, entities are individuals with bounded rationality that will act independently. There is no central controller for directing and coordinating individual entities. Formal computing models and techniques are often used to describe how the entities acquire and improve their reactive behavior based on their local and/or shared utilities, and how the behavior and utilities of the entities become goal-directed.

Distributed. Autonomous entities with localized decision-making capabilities are distributed in a heterogeneous computing environment, and locally interact among themselves to exchange their state information or to affect the states of others. In their distributed problem solving (e.g., scheduling and optimization), they continuously measure, update, and share information with other entities following certain predefined protocols.

Emergent. Distributed autonomous entities collectively exhibit complex (purposeful) behavior that is not present nor predefined in the behavior of the autonomous entities within a system. One of the interesting issues in studying the emergent behavior that leads to some desired computing solutions (e.g., optimal resource allocation) is how to mathematically model and measure the interrelationships between the local goals of the entities and the desired global goal(s) of the NIC system in a specific application.

Adaptive. Entities often change their behavior in response to changes in the environment in which they are situated. In doing so, they utilize behavioral adaptation mechanisms to continuously evaluate and fine-tune their behavioral attributes with reference to their goals as well as on-going feedback (e.g., intermediate rewards). Evolutionary approaches may be used to reproduce high-performing entities and eliminate the poor ones.

Self-organized. In NIC, autonomous entities and their environment are the basic elements. Local interactions between entities and their environment are the force that drives an NIC system to evolve towards certain desired states. Self-organization is the essential process of its working mechanism. Through the local interactions, an NIC system self-aggregates and amplifies the outcome of entity behavior. In other words, NIC-based systems utilize autonomous entities that self-organize to achieve the goals of systems modeling and problem solving.

NIC in Characterizing Complex Behavior

A complex system can be studied in many different ways. The obvious way to model a complex system is to look at it from the outside, observing various types of systems behavior and trying to summarize

them using models. Assumptions about unknown mechanisms have to be made in order to get the process started. Given some observable behavior of the desired system, NIC designers can verify the model by comparing the behavior of the model with the desired features. This process will be repeated several times before a good, though probably not perfect, prototype is found. Apart from obtaining a working model of the desired system, an important by-product of the process is the discovery of the mechanisms that were unknown when the design process first started.

Immune Systems Behavior. The human immune system is a typical example of a highly sensitive, adaptive, and self-regulated complex system involving numerous interactions among a vast number of cells of different types. Despite there being many clinical case studies and empirical findings on this subject [1], the working mechanism underlying the complex process of HIV invasion, erosion, and eventual crash of the immune system, such as how the local interactions at the level of HIV, T cells, and B cells affect this process, remains to be fully understood (i.e., characterized and predicted).

Conventional modeling and simulation technologies are useful only to a very limited extent due to the computational scale and costs involved. This endeavor presents a tremendous challenge to the field of computing, since (1) the task of computing is seamlessly carried out in a variety of physical embodiments, and (2) there is no single multi-purpose or dedicated machine that can manage to accomplish a job of this nature. In other words, the key to success for such applications lies in a large-scale deployment of computational entities or agents capable of autonomously making their localized decisions and achieving their collective goals.

In order to understand the dynamics in the immune system during an HIV attack, we can use a two-dimensional lattice to build our NIC model. The lattice is circular so that the edges wrap around each other. Each site can be inhabited by HIV as well as immune cells, and they behave in the following ways: (1) *Interactions*: T cells recognize HIV by its signature (protein structure); HIV infects and kills cells. (2) *Proliferation*: reactions stimulate lymphoid tissues to produce more T cells. T cells are naturally reproduced. (3) *Death*: Besides being killed, HIV and T cells die naturally. (4) *Diffuse*: HIV diffuse from densely populated sites to neighboring sites (see Figure 2).

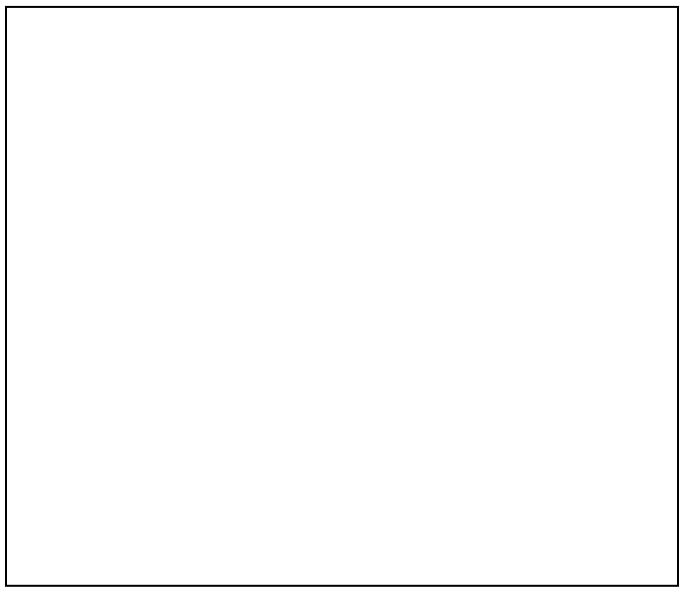

Figure 2. Modeling HIV, T cells, and O(ther) cells.

Figure 3 presents the temporal emergence of three-stage dynamics in HIV infection, which was generated from our NIC model described above [12]. The three stages are: (1) before B: the primary response, (2) B~C: the clinical latency, and (3) after D: the onset of AIDS. At A, the HIV population reaches a maximum point. Starting from C, the mechanism that decreases the natural ability of an organism to reproduce T cells is triggered. These NIC-generated results are consistent with empirically observed phenomena [1].

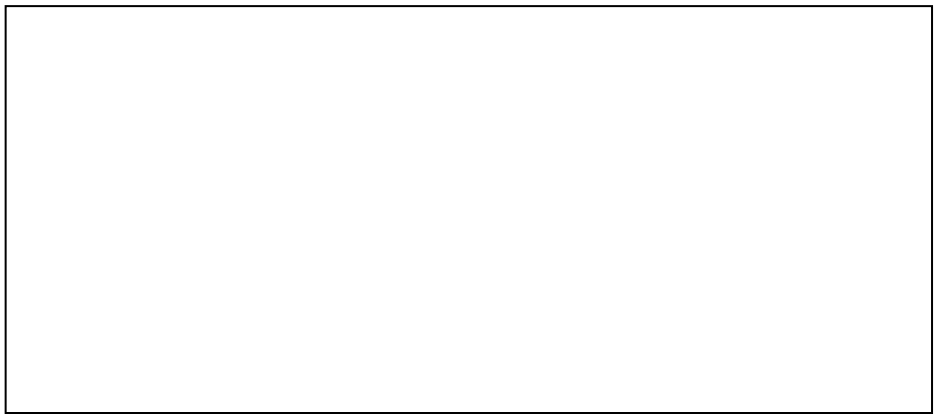

Figure 3. Simulation results on an HIV population at difference phases of AIDS.

Furthermore, our experiments also revealed [12] that AIDS cannot break out if HIV only destroys T cells without weakening the T cell reproduction mechanism. The emergence in "shape space" indicates that it is because of HIV's fast mutation that the immune system cannot eradicate HIV as easily as it does to other invaders. These discoveries can help researchers understand HIV-immune interaction dynamics more comprehensively.

In summary, the NIC approach to systems modeling starts from a microscopic view of the immune system. The elements of the model are the basic units of the immune system, namely, HIV and the immune cells. The model aims at capturing the essence of the immune system, but simplification is inevitable. Note that the autonomous entities in the model that belong to the same types normally have a similar set of behavior rules. The only difference among them is the parameters of the rules, which may be adapted in the lifetime of the entities. Probabilistic selection of certain behavior is also common in the entities. It must be emphasized that the environment of the model can also be considered as a unique entity in the model with its own behavior rules.

Understanding Self-Organized Web Regularities. Researchers have identified several interesting, self-organized regularities related to the World Wide Web, ranging from the growth and evolution of the Web to the usage patterns of Web surfing. Many regularities are best represented by characteristic distributions following either a Zipf-like law or a power law. Random-walk models [4] have been used to simulate some statistical regularities empirically observed on 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 interrelationships between user behavior and the contents or structure of the Web. The issues of user interest and motivation to navigate the Web are among the most important factors that directly determine the navigation behavior of users.

In an NIC approach to regularity characterization, Liu et al. [10] have gone one step further by proposing a computing model of Web surfing that takes into account the characteristics of users, such as interest profiles, motivations, and navigation strategies. In their model, users are viewed as *information-foraging entities* inhabiting the Web space, which is a collection of websites connected by hyperlinks. When an entity finds certain websites that have contents related to its topic(s) of interest, it will become more ready to search websites at the next level; that is, it becomes more motivated to surf deeper. On the other hand, when an entity does not find any interesting information after a certain amount of foraging, or when it has found enough contents to satisfy its interest, it will stop foraging and leave the Web space.

Liu et al. have conducted experiments in which users are categorized into three different groups: *recurrent* users who are familiar with the Web structure, *rational* users who are new to the website but know clearly what they are looking for, and *random* users who have no strong intention to retrieve any information but are just "wandering" around. The results of these experiments, which use both synthetic and empirical data from NASA, show that the foraging entity model can readily generate power-law distributions in surfing step and link-click-frequency, which are similar to those of the real world, and hence they offer a whitebox explanation of self-organized Web regularities.

NIC in Solving Computing Problems

The key factors contributing to the success of the above systems modeling examples are the distinctive characteristics of the elements they have modeled. As Marvin Minsky of MIT suggested in his *Society of Minds* theory, "To explain the mind, we have to show how minds are built from mindless stuff, from parts that are much smaller and simpler than anything we'd consider smart." So, if a person wants to formulate a problem-solving strategy based on some observation from nature, how and where should he/she start?

To formulate an NIC problem-solving system, one needs a working system in the natural or physical world, after which models can be extracted. As with complex systems modeling, the abstracted behavior of the working system becomes the property of the elements to be modeled.

Constraint Problem Solving. Based on the general principles of "survival of the fittest" – whereby poor performers will be eliminated – and the "law of the jungle" – whereby weak performers will be eaten by stronger ones – Liu et al. [6][7] have devised NIC systems to solve some well-known constraint satisfaction problems. One of them is the N-queen problem, in which N queens are required to be placed on an N × N chessboard, such that no two queens are placed in the same row, column, or diagonal. Based on the rules of the problem, an NIC model is formulated in the following manner. Each queen is modeled as an autonomous entity in the system and multiple queens are assigned to each row of the chessboard (a grid environment). This is to allow for competition between the queens in the same row so that the queen with the best strategy survives. The system calculates the number of violated constraints for each position on the grid. This represents the environmental information that all queens have access to in making decisions about where to move, the possible movements being restricted to positions in the same row. There are three possible movement strategies: the *random-move* strategy, which involves randomly selecting a new position for a queen; the *least-move* strategy, which involves selecting the position with the least number of violations; and the *coop-move* strategy, which

promotes cooperation between the queens by eliminating certain positions in which one may potentially create conflicts with other queens. These strategies are selected probabilistically.

An initial amount of *energy* is given to each queen. A queen will "die" if its energy falls below a predefined threshold. Energy can change in two ways. When a queen moves to a new position that violates the set constraint with *m* queens, it loses *m* units of energy. This will also cause those queens that attack this new position to lose *1* unit of energy. The intention is to encourage the queens to find a position with the smallest number of violations. The law of the jungle is implemented by having two or more queens to occupy the same grid position and fight over it. The queen with the highest energy will win and "eat" the loser(s) by absorbing all its (their) energy. The above model is able to efficiently solve the N-queen problem with moderate amount of computation.

Optimization. In the commonly used version of a genetic algorithm [3], a member of the family of evolutionary algorithms, the process of sexual evolution is simplified to selection, recombination, and mutation, without the explicit identification of male and female, for example, in the gene pool. John Holland of the University of Michigan, in his quest for a model to understand evolution, has developed a genetic algorithm for optimization. The basic unit in this artificial evolution is a candidate solution to the optimization problem, commonly termed a chromosome. A genetic algorithm has a pool of them. Interactions between candidate solutions are achieved via artificial reproduction where operations mimicking natural evolution allow the candidate solutions to produce offspring that carry part of either parent (crossover) with occasional variation (mutation). While reproduction can be viewed as the cooperative side of all the chromosomes, competition among chromosomes for a position in the next generation realizes the principle of survival of the fittest.

Evolutionary autonomous agents [9] and evolution strategies [11], 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 this simplification and modification, evolutionary algorithms capture the essence of natural evolution and are proven multi-objective optimization techniques. Another NIC algorithm that has been applied in similar domains is the Ant System [2], which mimics the food foraging behavior of ants.

Development of Autonomy Oriented Computing (AOC)

As a concrete manifestation of the NIC paradigm, *Autonomy Oriented Computing* (AOC) has become a new field of computer science that systematically explores the metaphors and models of *autonomy* as offered by nature (e.g., *physical, biological, and social entities* of varying complexity), as well as their role in addressing our practical computing needs. It studies emergent autonomy as the core behavior of a computing system and draws on such principles as multi-entity formulation, local interaction, nonlinear aggregation, adaptation, and self-organization [5][8]. Three general approaches can readily be applied in developing an AOC system: AOC-by-fabrication, AOC-by-prototyping, and AOC-by-self-discovery. These three approaches have been found to be promising in several application areas [7][9][10][12]. Recent work on AOC has opened up new ways of understanding and developing NIC theories and methodologies, and has provided working examples that demonstrate the power and features of the NIC paradigm in: (1) characterizing emergent behavior in natural and artificial systems that involve a large number of self-organizing, locally-interacting entities, and in (2) solving large-scale computation, distributed constraint satisfaction, and decentralized optimization problems [8].

Conclusion

In this article, we have presented an overview of the *Nature Inspired Computing* (NIC) paradigm, as well as its AOC manifestation, by describing the underlying principles and conceptual constructs and demonstrating them with examples of complex systems modeling and problem solving. The NIC paradigm differs from the imperative, logical, constraint, object-oriented, or component-based paradigms, not only in its fundamental concepts and constructs, as mentioned above, but also in the effectiveness and efficiency of computing that can be achieved through its special *ADEAS characteristics*. NIC approaches have been found to be most effective in dealing with problems of the following nature:

- Problems of high complexity (e.g., the systems to be characterized involve a large number of autonomous entities, or the computing problems to be solved involve large-scale, high-dimension, highly nonlinear interactions/relationships, and highly interrelated/constrained variables);

- Highly distributed and locally interacting problems (i.e., they are not centralized, nor ready/efficient for batch processing);

- The environment in which the problems are situated is dynamically updated or changes in real time;

- The goal of modeling and analysis is not to extract some superficial patterns/relationships (i.e., data transformation or association from one form to another), but to discover and understand the *deep patterns* — the underlying mechanisms/processes that produce the data in the first place (i.e., to provide an explanation of the cause/origin).

We will continue to see new NIC theories and methodologies developed, and to understand their wide-ranging impacts on modern computer science as well as on other disciplines such as sociology, economics, and natural sciences. Promising applications will be found in explaining gene regulatory networks (GRNs) and drug resistance mechanisms for anti-cancer drug design, predicting the social-economic sustainability of self-organizing on-line markets or communities, and real-time autonomous data processing in massive mobile sensor networks for eco-geological observations, to name just a few.

References

1. Coffin, J.M. HIV population dynamics in Vivo: implications for genetic variation, pathogenesis, and therapy. *Science*, 267 (1995), 483-489.
2. Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: optimization by a colony of cooperative agents. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, 26, 1 (1996), 1-13.
3. Holland, J.H. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.
4. Huberman, B.A., Pirolli, P.L.T., Pitkow, J.E., and Lukose, R.M. Strong regularities in World Wide Web surfing. *Science*, 280 (3 Apr 1997), 96-97.
5. Liu, J. *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation*. World Scientific, 2001.
6. Liu, J. and Han, J. ALIFE: a multi-agent computing paradigm for constraint satisfaction problems. *International Journal of Pattern Recognition and Artificial Intelligence*, 15, 3 (2001), 475-491.

7. Liu, J., Han, J., and Tang, Y.Y. Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136, 1 (2002), 101-144.
8. Liu, J., Jin, X., and Tsui, K.C. *Autonomy Oriented Computing (AOC): From Problem Solving to Complex Systems Modeling*. Springer, Dec 2004.
9. Liu, J., Tang, Y.Y., and Cao, Y. An evolutionary autonomous agents approach to image feature extraction. *IEEE Transactions on Evolutionary Computation*, 1, 2 (1997), 141-158.
10. Liu, J., Zhang, S., and Yang, J. Characterizing Web usage regularities with information foraging agents. *IEEE Transactions on Knowledge and Data Engineering*, 16, 5 (2004), 566-584.
11. Schwefel, H.-P. *Numerical Optimization of Computer Models*. Wiley, 1981.
12. Zhang, S. and Liu, J. A massively multi-agent system for discovering HIV-immune interaction dynamics. In T. Ishida, L. Gasser, and H. Nakashima (Eds.), *Massively Multi-Agent Systems I: First International Workshop, MMAS 2004, Kyoto, Japan, December 10-11, 2004, Revised Selected and Invited Papers*. LNCS Vol. 3446, Springer, 2005, 161-173.

Acknowledgement
The authors would like to acknowledge the funding from the Research Grants Council of Hong Kong (RGC/HKBU2121/03E and RGC/HKBU2040/02E) and the Natural Sciences and Engineering Research Council of Canada for supporting some of the mentioned work as carried out at Prof. Liu's AOC/AAMAS research lab.