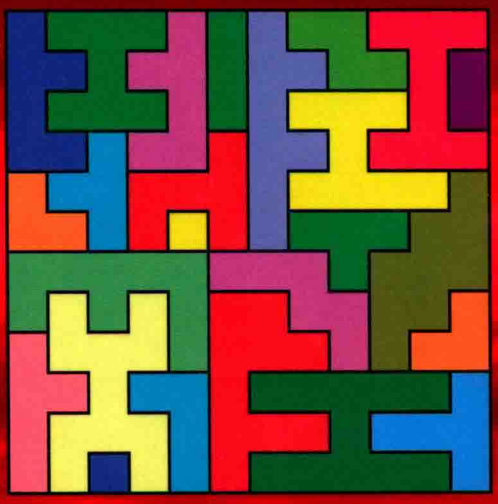
CHINESE SCIENCE TODAY

当代中国科学



Coevolutionary Computation and Multiagent Systems

Licheng Jiao, Jing Liu & Weicai Zhong

WIT PRESS

科学出版社
Science Press

Coevolutionary Computation and Multiagent Systems

Licheng Jiao
Jing Liu
Weicai Zhong
Xidian University, China

WITPRESS Southampton, Boston

Licheng Jiao
Jing Liu
Weicai Zhong

Xidian University, China

Published by

WIT Press
Ashurst Lodge, Ashurst, Southampton, SO40 7AA, UK
Tel: 44 (0) 238 029 3223; Fax: 44 (0) 238 029 2853
E-Mail: witpress@witpress.com
http://www.witpress.com

For USA, Canada and Mexico

WIT Press
25 Bridge Street, Billerica, MA 01821, USA
Tel: 978 667 5841; Fax: 978 667 7582
E-Mail: infousa@witpress.com
http://www.witpress.com

British Library Cataloguing-in-Publication Data

A Catalogue record for this book is available
from the British Library

ISBN: 978-1-84564-638-7
eISBN: 978-184564-639-4

Library of Congress Catalog Card Number: 2011936268

The original Chinese language work has been published by SCIENCE PRESS, Beijing.

Printed by Lightning Source, UK.

Coevolutionary Computational and Multiagent Systems

Preface

One of the remarkable features of modern science and technology is the mutual penetration and promotion between life science and engineering science. The rapid development in the field of computational intelligence, which includes neural networks, fuzzy logic, natural computation, evolutionary computation, and so on, also reflects such a feature. The research in the field of computational intelligence and recognition aims to achieve human intelligence by intelligently handling nonlinear real-world information, including image, video, voice, touch, etc. Due to its importance in information technology, computational intelligence has attracted increasing attention, and is also listed as one of the five key directions by the National Natural Science Foundation Committee of China.

According to fossil records, life has preceded the singled celled organism, and undergone a roadmap of evolution from a low level to a high level, and from the simple to the complex. Human beings, superior living organisms with thought and intelligence, are a spectacular success of evolution. Human beings can not only adapt to their environment, but also improve their adaptability via learning, imitation, and creation. Since the last century, researchers have extended the areas they study to include nature and human beings. Researchers have studied the evolutionary process of human beings itself, and extracted it as an optimization process. The classic example being evolutionary algorithms (EAs).

EAs are a kind of stochastic optimization approach, inspired by theories of biological evolution. Traditionally, EAs have been categorized into four subfields: Genetic Algorithms (GAs), Evolution Strategies (ESs), Evolutionary Programming (EP), and Genetic Programming (GP). Nowadays, the boundaries between these subfields are more fluid and the methods are often grouped together using the term Evolutionary Algorithms. This class of method does not require derivatives of the functions defining the problem and it is relatively robust and flexible for solving nonlinear optimization problems, due to the stochastic search operators involved in the algorithmic definition.

Although simplistic from a biologist's viewpoint, EAs are sufficiently complex to provide robust and powerful adaptive search mechanisms. Today, Evolutionary Computation (EC), the computation model based on EAs, is a thriving field, and EAs have been successfully applied to a broad variety of problems in an extremely diverse array of fields, such as acoustics,

aerospace engineering, astronomy and astrophysics, chemistry, electrical engineering, financial markets, game playing, geophysics, materials engineering, mathematics and algorithmics, molecular biology, pattern recognition and data mining, robotics, routing and scheduling.

Although EAs have many advantages over traditional optimization approaches and have been successfully applied to many fields, they still have weaknesses. Their main disadvantages are the ability to be trapped in local optima and have a high computational cost, thus traditional EAs' ability in solving large-scale problems is weak. It is worth stepping back and exploring how to best learn from nature and how to incorporate our existing knowledge of artificial intelligence into EC.

As a new promising branch of EC, coevolutionary computation has attracted increasing attention recently. The further development of this branch will require further efforts from various researchers. This book studies the background and foundation of coEC in depth, and introduces organizational coevolutionary algorithms and multiagent evolutionary algorithms. We introduce the dynamics of coevolutionary systems, prove the convergence of algorithms, and analyze the computational complexity of algorithms. This book focuses on both the theoretical foundation and practical applications, and not only provides new coevolutionary algorithms, but also provides new ideas and methods for further developing computational intelligence.

The whole book is divided into 10 chapters, which include the introduction on EC, coEC, complex adaptive systems, and multiagent systems (Chapter 1), organizational coevolutionary algorithms and their applications on large-scale classification, satisfiability problems, numerical optimization, and VLSI Floorplan problems (Chapters 2, 3, 4, 5, 6), multiagent evolutionary algorithms and their applications to high-dimensional numerical optimization, combinatorial optimization problems, and constraint satisfaction problems (Chapters 7, 8, 9, 10).

The work in this book was supported by the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation of China (Nos 61103119, 61072106, 61001202, 61003199 and 60970067), the Program for Cheung Kong Scholars and Innovative Research Team in University of China (No. IRT1170), the Fund for Foreign Scholars in University Research and Teaching Programs of China ("111" Project, No. B07048), and the European Union Seventh Framework Programme (No. 247619). We would like to thank those people who always support our work. They are Prof. Zheng Bo (Academician of the Chinese Academy of Sciences, Xidian University, China), Prof. Guoliang Chen (Academician of the Chinese Academy of Sciences, University of Science and Technology of China), Prof. Xin Yao (University of Birmingham, UK), Prof. Qingfu Zhang (University of Essex, UK), etc. We would also like to thank the editors at Science Press, China and WIT Press, UK for their hard work.

Licheng Jiao, Jing Liu, Weicai Zhong, April, 2012
Xidian University, Xi'an, China

Table of Contents

Chapter 1

Introduction

Evolutionary computation (EC) and multiagent systems (MAS), as new branches in the field of artificial intelligence, have attracted increasing attention. This book introduces the authors' recent work in these two fields. Therefore, this first chapter is devoted to the related background in the fields of EC and MAS.

1.1 Evolutionary computation

The origin of EC can be traced back to the late 1950s, and the influencing works include Bremermann [1], Friedberg [2], [3], Box [4], and others. Although EC remained relatively unknown to the broader scientific community for almost three decades, which was largely due to the lack of available powerful computer platforms at that time and some methodological shortcomings of those early approaches [5], it has started to receive significant attention since the 1980s, which benefited from the fundamental work of Holland [6], Rechenberg [7], Schwefel [8], and Fogel [9] during the 1970s.

1.1.1 Structure of evolutionary algorithms

Evolutionary algorithms (EAs) [5], [10]–[13] mimic the process of natural evolution, the driving process for the emergence of complex, and well-adapted organic structures. EAs maintain a population of individuals to produce approximately optimal solutions to the problem. Each individual in the population is evaluated, receiving a measure of its fitness in the environment. At each generation, they involve a competitive selection that weeds out poor individuals, thus exploiting the available fitness information. The individuals with high fitness are perturbed by using crossover and mutation operators, providing general heuristics for exploration. The non-deterministic nature of reproduction leads to a permanent production of novel genetic information and therefore to the creation of differing offspring. This neo-Darwinian model of organic evolution is reflected by the structure of the following general EA [5].

Algorithm 1.1: General structure of EAs

```
t ← 0;
Initialize P(t);
```

```
Evaluate P(t);
While not terminate do
  P'(t) ← Variation [P(t)];
  Evaluate [P'(t)];
  P(t+1) ← Select[P'(t) ∪ Q];
  t ← t+1;
End.
```

In this algorithm, $P(t)$ denotes a population of n individuals at generation t. Q is a special set of individuals that might be considered for selection, e.g., $Q = P(t)$ (but $Q = \emptyset$ is possible as well). An offstring population $P'(t)$ of size m is generated by means of variation operators such as recombination and/or mutation from the population $P(t)$. The offstring individuals are then evaluated by calculating the objective function values for teaching of the solutions represented by individuals in $P'(t)$, and a selection based on the fitness values is performed to drive the process toward better solutions. It should be noted that $m = 1$ is possible, thus including the so-called steady-state selection schemes [14], [15] if used in combination with $Q = P(t)$. Furthermore, by choosing $1 \leq m \leq n$ an arbitrary value of the generation gap [16] is adjustable, such that the transition between strictly generational and steady-state variants of the algorithm is also taken into account by the formulation offered here. It should also be noted that $m > n$, i.e., a reproduction surplus, is the normal case in nature.

Although simplistic from a biologist's viewpoint, EAs are sufficiently complex to provide robust and powerful adaptive search mechanisms. Today, EAs have been successfully applied to a broad variety of problems in an extremely diverse array of fields, such as acoustics, aerospace engineering, astronomy and astrophysics, chemistry, electrical engineering, financial markets, game playing, geophysics, materials engineering, mathematics and algorithmics, molecular biology, pattern recognition and data mining, robotics, routing, and scheduling.

1.1.2 Branches of evolutionary algorithms

The majority of current implementations of EAs descend from three strongly related but independently developed approaches: genetic algorithms (GAs), evolutionary programming, and evolution strategies, and one based on GAs: genetic programming.

GAs, introduced by Holland [6], [17], [18], and subsequently studied by De Jong [19]–[22], Goldberg [23]–[27], and others, have been originally proposed as a general model of adaptive processes, but by far the largest application of the techniques in the domain of optimization [21], [22].

Evolutionary programming, introduced by Fogel [9], [28] and extended by Burgin [29], [30], Atmar [31], Fogel [32]–[34], and others, was originally offered as an attempt to create artificial intelligence. The approach was to evolve finite-state machines (FSM) to predict events on the basis of former observations. An FSM is an abstract machine which transforms a sequence of input symbols into a sequence of output symbols. The transformation depends on a finite set of states and a finite set of state transition rules. The performance of an FSM with respect to its environment might then be measured on the basis of the machine's

prediction capability, i.e., by comparing each output symbol with the next input symbol and measuring the worth of a prediction by some payoff function.

Evolution strategies, as developed by Rechenberg [35], [36] and Schwefel [37], [38], and extended by Herdy [39], Kursawe [40], Ostermeier [41], [42], Rudolph [43], Schwefel [44], and others, were initially designed with the goal of solving difficult discrete and continuous, mainly experimental [45], parameter optimization problems.

Genetic programming applies evolutionary search to the space of tree structures which may be interpreted as computer programs in a language suitable to modification by mutation and recombination. The dominant approach to genetic programming uses (a subset of) LISP programs (S expressions) as genotype space [46], [47], but other programming languages including machine code are also used (see, e.g., [48–50]).

1.1.3 Evolutionary computation and complex adaptive systems

Although EAs have many advantages over traditional optimization approaches and have been successfully applied to many fields; they still have weaknesses. Their main disadvantages are that they are easy to be trapped in local optima and have a high computational cost, thus traditional EAs' ability in solving large-scale problems is weak.

The world is replete with complex systems. They range from natural systems, such as the biosphere and climate, to human systems, such as communications, transport, and global financial markets. Understanding and managing all these systems is one of the most pressing problems of our time. Holland [51], [52] is the first to systematically and rigorously describe and define adaptive process (adaptation) from biology to investigate the complex phenomena generated by complex natural and artificial systems, more precisely, complex adaptive systems (CASs), which are a collective designation for nonlinear systems defined by the interaction of large number of adaptive agents. In fact, GAs are one realization of CASs. Thus, the theoretical framework for adaptation presented in [51] can be a theoretical footing of GAs, and even EAs.

In CAS, the basic units are adaptive agents with goals and learning capabilities and the entire system is built on the interactions among such agents, environment, etc. Holland's formal framework of adaptation contains seven central components, which are listed in Table 1.1. In addition, an adaptive system can be expressed as a four-tuple $(\Lambda, \Omega, I, \tau)$. Since the adaptive plan τ initially has incomplete information about which structure is the most suitable one, to reduce this uncertainty, the plan τ must test the performance of different structures in the environment. When the plan τ tries a structure $\Lambda(t) \in \Lambda$ at time t, the particular environment $E \in \varepsilon$ confronting the adaptive system signals a response $I(t) \in I$. The performance or payoff $\mu_E(\Lambda(t))$, given by the function μ_E, is generally an important part of the information $I(t)$. When a plan receives only information about payoff, it is called as payoff-only plan. Once $I(t)$ and $\Lambda(t)$ are given, the plan τ determines operator $\omega_t \in \Omega$, and hence $\Lambda(t+1)$ by drawing a random sample from $\Lambda(t)$ according to the distribution determined by ω_t.

EAs are actually one kind of the adaptive plans. However, compared with the above original framework of adaptation, we think the traditional EAs are weak in the following three aspects.

Table 1.1: Seven central components in Holland's formal framework of adaptation

E	The environment of the system undergoing adaptation.
μ	The measure of the performance of different structures in the environment.
Λ	The set of attainable structures and the domain of action of the adaptive plan.
Ω	The set of operators for modifying structures with $\omega \in \Omega$ being a function mapping Λ into some set of probability distributions over Λ.
I	The set of possible inputs to the system from environment.
τ	The adaptive plan on the basis of the input I and structure Λ at time t to determine which operator is to be applied at time t.
χ	The criterion for comparing the efficiency of different plans.

(1) The adaptiveness of the plan τ generating and testing different structures in different environments is lost to a large extent. In fact, the operator ω_t is determined on the basis of the information $I(t)$ and $\Lambda(t)$ in Holland's original framework; however, Holland simply used a fixed operator sequence to generate new structures in the next trial in late reproductive plan. Moreover, nearly all following researchers in GAs, even in EAs, follow this pattern or use equivalent ways. Despite the fact that some researchers have proposed some "adaptive" methods to dynamically tune the parameters of operators during the evolutionary process, there is still a considerable distance to the "adaptiveness" of the adaptive plan given in the original framework of adaptation.

(2) A number of operators have been proposed, but little work has been done to select an appropriate set of them according to a specific task. Previous work has shown that no single operator can perform well and uniformly outperform other operators over all search and learning tasks. This has been confirmed by the "no free lunch theorems" [53]. Thus, it is necessary to select an appropriate set of operators (including the ranges of corresponding parameters) based on the characteristics of the problems for EAs during the evolutionary process. Accordingly, it is also necessary to choose proper representations of solutions among a variety of representations available.

(3) Available EAs have not accumulated and made use of their experiences in multiple applications. Users would expect to choose an appropriate EA according to the problems that they are going to perform since they often lack not only the expertise necessary to select a suitable algorithm, but also the availability of many models to proceed on a trial-and-error basis. An inappropriate selection of algorithm will result in slow convergence, or even produce a suboptimal solution due to local optima in the complex problem. In addition, users want to profit from repetitive use of previous experiences over similar tasks and problems, and do not need to start from scratch on new tasks.

To summarize, traditional EAs can be viewed as one realization of CAS but ignore individuals' learning capabilities. On the other hand, some researchers had also pointed out that Holland's original vision of CAS was more like an agent-based system than a typically centralized EAs used today [54].

1.2 Multiagent systems

The study of MAS began in the field of distributed artificial intelligence about 30 years ago and focuses on systems in which many intelligent agents interact with each other. Nowadays, these systems are not simply a research topic but are also becoming an important subject of academic teaching and industrial and commercial application.

MAS researchers develop communication languages, interaction protocols, and agent architectures that facilitate the development of MASs. For example, an MAS researcher can tell you how to program each ant in a colony in order to get them all to bring food to the nest in the most efficient manner, or how to set up rules so that a group of selfish agents will work together to accomplish a given task. MAS researchers draw on ideas from many disciplines outside of artificial intelligence, including biology, sociology, economics, organization and management science, complex systems, and philosophy.

1.2.1 Agents

The agents are considered to be autonomous entities, such as software programs or robots. Their interactions can be either cooperative or selfish. That is, the agents can share a common goal (e.g., an ant colony), or they can pursue their own interests (as in the free market economy). In fact, the notion of agents should be taken in a broad sense, encompassing a wide spectrum of computational entities that can sense their local task conditions and accordingly make decision on how to react to the sensed conditions by performing certain behaviors in the task environments.

Wooldridge and Jennings define the agents as follows [55].

Definition 1.1: *An* **agent** *is a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its design objectives.*

This definition does not say anything about what type of environment an agent occupies. Thus, agents can occupy many different types of environment. Figure 1.1 gives an abstract, top-level view of an agent [56]. In this diagram, we can see the action output generated by the agent in order to affect its environment. In most domains of reasonable complexity, an agent will not have complete control over its environment. It will have at best partial control, in that it can influence it. From the point of view of the agent, this means that the same action performed twice in apparently identical circumstances might appear to have entirely different effects, and in particular, it may fail to have the desired effect. Thus, agents in all but the most trivial of environments must be prepared for the possibility of failure. We can sum this situation up formally by saying that the environments are non-deterministic.

Normally, an agent will have a repertoire of actions available to it. This set of possible actions represents the agent's effectoric capability: its ability of modify its environments. Note that not all actions can be performed in all situations. Actions therefore have preconditions associated with them, which define the possible situations in which they can be applied.

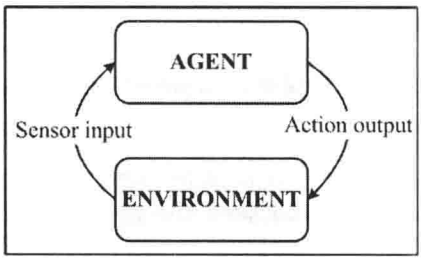


Figure 1.1: An agent in its environment. The agent takes sensory input from the environment, and produces as output actions that affect it. The interaction is usually an ongoing, non-terminating one.

The key problem faced by an agent is deciding which of its actions it should perform in order to best satisfy its design objectives. Agent architectures are really software architectures for decision-making systems that are embedded in an environment. The complexity of the decision-making process can be affected by a number of different environmental properties.

An intelligent agent is one that is capable of taking a flexible autonomous action in order to meet its design objectives, where flexibility means three things [55]:

- *Reactivity*: Intelligent agents are able to perceive their environment, and respond in a timely fashion to the changes that occur in it, in order to satisfy their design objectives.
- *Proactiveness*: Intelligent agents are able to exhibit goal-directed behavior by taking the initiative, in order to satisfy their design objectives.
- *Social ability*: Intelligent agents are capable of interacting with other agents (and possibly humans), in order to satisfy their design objectives.

1.2.2 MAS

Obviously, an MAS is a system composed of a set of agents. However, compared with a single agent, the characteristics of MASs are that (1) each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint; (2) there is no system global control; (3) data are decentralized; and (4) computation is asynchronous.

With the characteristics of being autonomous, adaptive, robust, and easy to implement, agent-based approaches have found many potential applications in dealing with tasks that are less-structured or ill-defined. In such tasks, complete mathematical or computational solutions may be either unavailable or too expensive to use. Regardless of their domains of application, agents often have one thing in common, namely, they locally interact with their task environments, computational or physical. Responding to different local constraints received from their task environments, the agents can select and exhibit different behavioral patterns. In the case of optimization or search, the behavioral patterns of the agents may be reflected in their decisions on in what direction and how much localized search is necessary. The behavioral patterns of the agents may be predefined and activated whenever certain

conditions are satisfied during agent–environment interaction, or dynamically acquired based on some embedded learning mechanisms.

Autonomous agents are often developed to solve specific tasks in a distributed fashion, involving multiple agents of different capabilities. These agents will coordinate, cooperate, and sometimes compete among themselves in order to most effectively accomplish a given task. Singh [57] proposed a theoretical framework for modeling the intensions, know-how, and communications of agents in an MAS. Barbuceanu and Fox [58] demonstrated how communicative actions, conversations, and a decision theory can be integrated in order to achieve multiagent coordination.

The main application of MAS at the moment can be listed as follows:

- *Problem solving*: As an alternative to centralized problem solving, either because problems are themselves distributed, or because the distribution of problem solving between different agents reveals itself to be more efficient way to organize the problem solving – it can be flexible and allow failures in the system – or because, in some cases, it is the only way to solve the problem.
- *Multiagent simulation*: Simulation is widely used to enhance knowledge in biology or in social science and MAS gives us the possibility to make artificial universes that are small laboratories for the testing of theories about local behaviors. Examples include *Simdelta* (Cambier and Bousquet) and *SimPop* (Bura).
- *Construction of synthetic worlds*: These artificial universes can be used to describe specific interaction mechanisms and analyze their impact at a global level in the system. The entities that are represented are usually called animats, since they are mainly inspired by animal behaviors (hunting, searching, or gathering habits). The aim of this research is to have societies of agents that are very flexible and can adapt even in cases of individual failure. (For example, when robots are sent on an expedition, they are required to be very independent of the instructions they could receive).
- *Collective robotics*: Defining the robots as an MAS where each subsystem has a specific goal and deals with that goal only. Once all the small tasks are accomplished the big task is also accomplished. MAS approaches can also be used in the coordination of different mobile robots in a common space.
- *Kenetic program design*: MAS can also be seen as a very efficient modular way to program.

References

[1] H. J. Bremermann, 'Optimization through evolution and recombination', in *Self-Organizing Systems*, M. C. Yovitsetal (ed.), Spartan, Washington, DC (1962).
[2] R. M. Friedberg, 'A learning machine: Part I', *IBMJ*, 2(1), 2–13, January (1958).
[3] R. M. Friedberg, B. Dunham, and J. H. North, 'A learning machine: Part II', *IBMJ*, 3(7), 282–287, July (1959).
[4] G. E. P. Box, 'Evolutionary operation: a method for increasing industrial productivity', *Appl. Statistics*, VI (2), 81–101 (1957).
[5] T. Bäck, U. Hammel, and H.-P. Schwefel, 'Evolutionary computation: comments on the history and current state', *IEEE Trans. Evol. Comput.* 1(1), 3–17 (1997).