面向自治计算介绍

An Introduction to *Autonomy-Oriented Computing*（AOC）

刘际明[i]，黄来磊[ii]，夏尚[ii]

香港浸会大学计算机科学系

Summary

The notion of Autonomy-Oriented Computing (AOC) first appeared in Liu's book entitled *Autonomous Agents and Multi-Agent Systems: Explorations on Learning, Self-Organization and Adaptive Computation*[1]. Yet, the effective use of AOC in a variety of domains had been demonstrated and published beyond this book, covering constraint satisfaction problem solving[2][3][4][5][6], mathematical programming[7], optimization[8][9], image processing[10][11][12][13], and data mining[14][15]. Since 2000, research projects have been launched to study the AOC approaches to characterizing (i.e., modeling and explaining) observed or desired regularities in real-world complex systems, e.g., self-organized Web regularities[16] and HIV infection dynamics[17], as a white-box alternative to the traditional top-down or statistical modeling. Recently, new studies have been carried out to solve complex problems in the real world, e.g., energy distribution problems, epidemic interventions, problems in public health systems, and social and cultural dynamics.

Generally speaking, AOC makes use of autonomous entities in solving computational problems as well as in modeling complex systems. In so doing, it relies on a novel computational system, i.e., an open non-equilibrium system, that enables modeled entities to locally interact with each other as well as with their environment, actively carrying out information exchanges and utility updates based on certain nature or real-world inspired behavioral rules. The interactions of the entities will dynamically form both positive and negative feedback mechanisms. As a result, certain desirable behavior of the entities and/or effects will be non-linearly aggregated/amplified in certain spatial and temporal scales, as opposed to others. This process has been well utilized and demonstrated in the earlier self-organized computing work, e.g., collective problem solving with a group of autonomous robots[18][19] and behavioral self-organization[20][21]. A systematic description of the theoretical and practical issues in AOC was given in *Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling*[25], published by Springer in 2004. As compared to other paradigms, such as centralized computation and top-down systems modeling, AOC has been found to be effectively useful and promising in the following aspects: (1) capturing the essence of autonomy in natural and artificial systems; (2) solving computationally hard problems, e.g., large-scale computation, distributed constraint satisfaction, and decentralized optimization, that are dynamically evolving and highly complex in terms of interaction and dimensionality; (3) characterizing complex phenomena or emergent behavior in natural and artificial systems that involve a large number of self-organizing, interacting entities; (4) discovering laws and mechanisms underlying complex phenomena or emergent behaviors.

---

[i] 刘际明 (Jiming Liu)：香港浸会大学计算机科学系讲座教授、系主任，通讯作者。Email: jiming@comp.hkbu.edu.hk , http://www.comp.hkbu.edu.hk/~jiming/

[ii] 黄来磊，夏尚：香港浸会大学计算机科学系面向自治计算（AOC）研究组在读博士生。

概述

面向自治计算[iii]（Autonomy-Oriented Computing, AOC）首先由刘际明教授在《Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation》[1]一书中提出。15 年来，自治计算已广泛而有效地应用于多个研究领域，已大大的超越了此书所谈及的范畴，其涵盖了约束满足[2][3][4][5][6]、数学规划[7]、优化[8][9]、图像处理[10][11][12][13]以及数据挖掘[14][15]等。自治计算研究组从 2000 年开始运用自治计算来刻画（意指建模或理解）、分析和研究真实世界中观察到的、或期望的复杂系统特征和机制。如自治计算作为区别于传统自顶向下或统计分析的白盒分析方法成功刻画了网页的自组织特征[16]和HIV感染的动态过程[17]。近期研究着重于如何运用自治计算解决现实世界中的复杂系统问题,如能源分配问题、流行病传播控制问题、以及公共医疗系统问题等。

自治计算的主要应用包括利用自治计算体解决计算问题，和对复杂系统进行建模两个方面。自治计算的主要思想是：依存于一种新型的计算系统（如一个开放的非平衡系统），自治计算体遵循特定的源自自然或真实世界的行为规则，通过相互之间或与周围环境的局部交互，实现信息交换和效用更新。计算体的交互过程能够动态形成正反馈和负反馈机制，在这两种机制的作用下，某些期望的计算体行为或作用在一定的时空范围内将会被积聚或放大，而其他的行为或作用则被减弱或抑制。早期的自组织计算研究表明运用这一过程能够有效地解决众多复杂问题，如自治机器人协同下的问题求解[18][19]和基本运动行为的自组织[20][21]等。2004 年 Springer 出版的《Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling》[25]一书对自治计算的理论和应用进行了系统介绍。与其他计算范式（如集中式计算或自顶向下的系统建模）相比，自治计算的优势主要包括：（1）刻画自然系统与人工系统中自治行为的本质特征；（2）解决复杂性计算问题，尤其是具有交互性和多维度性的动态演化和高复杂度问题，如大规模计算、分布式约束满足、分布式优化等；（3）刻画自然和人工系统中的复杂现象和涌现行为；（4）发现复杂现象和涌现行为背后的原理机制。

1 引言

我们对世界的认识正经历着剧烈的变革，复杂系统与复杂性科学为我们进一步认识和理解世界提供了新的世界观，被称为二十一世纪的科学。复杂系统具有的不同于传统确定性系统的特点（如大规模、分布式、异构、动态以及开放等）已广泛存在于政治、经济、社会、自然等各个领域中。因此，如何描述复杂系统的行为特征和解决复杂系统计算问题迫在眉睫。如：针对近年来的世界金融危机，如何从系统的角度刻画全球经纪人的投资行为对金融市场的影响，如何描述危机发展、蔓延和减弱的过程，以及如何认定危机产生的根源。传统基于统计建模的分析方法已经不能很好的揭示其背后所蕴含的原理机制。再如：在城市交通系统这一动态开放的不确定系统中，驾驶员的自主性以及他们之间的信息交互使得系统的组成形式在不断的变化，传统的自顶向下计算方法已经无法有效解决开放、动态、时变系统的优化问题。正是由于复杂系统的大规模、分布式、异构、动态、开放等特征使得传统计算方法在如何描述复杂系统的行为特征和解决复杂系统计算问题时遇

[iii]面向自治计算 (Autonomy-Oriented Computing, AOC) 后文简称自治计算。

到了困难——如大规模可能导致计算效率低下；动态开放系统可能导致算法结构设计复杂；不确定性会引起算法稳定性和收敛性的恶化等。因此，我们需要一种全新的计算范型来解决现实世界中复杂系统的建模和具有复杂系统特征的计算问题求解[22][23]。

面向自治计算（Autonomy-Oriented Computing, AOC）[24][25][26]作为一种新颖的自然启发式计算范型（nature-inspired computing paradigm），为解决复杂系统计算问题提供了新的思路。利用自治计算解决问题，首先要理解目标系统的真实特征，然后采用自底向上[iv]的方法解释、重构系统的自治行为特征并揭示其背后的原理机制，最后通过运用系统规律机制，为问题的求解提供有效的计算手段。因此自治计算既是一种复杂系统建模的工具，又是一种解决复杂系统计算问题的手段。

自治计算与粒计算都是非传统的计算范型，具有类似的思想起源。自治计算从自然界中自组织和涌现的规律出发，从生物、社会等复杂系统的运行、构造中得到启发，设计计算系统，刻画复杂系统行为，解决复杂计算问题。而粒计算从人类认知规律、人类解决问题的过程中得到启发，基于多层次（multi-level）、多视角（multi-view）的结构化思想，利用粒度（granularity）的概念[27][28]设计计算算法来解决复杂问题。自治计算系统中因自治计算体的交互涌现出的宏观结构对应于粒计算系统中的层面（levels），不同层面上的自组织行为能够导致不同大小（层次）粒度（granules）的出现，从而使得整个系统出现多层次结构（multiple hierarchies）。自治计算侧重于关注自组织行为如何产生以及产生怎样的宏观结构，而粒计算则侧重于关注不同层次结构间的相互关系，以及利用这种关系解决计算问题。因此，自治计算与粒计算既有本质区别又有内在联系，二者之间的关系值得进一步的研究和探讨。

2 自治计算初识

自治计算是一种处理大规模、分布式、异构、动态、开放系统中的问题求解方法，主要用来分析、刻画和重构复杂系统的自治行为和特征，并为复杂问题求解提供有效的解决途径。自治计算除了满足收敛性、效率性和有效性等传统算法的要求外，还具有计算的拓展性、鲁棒性和自适应性等特征，这为复杂问题分析和求解提供了良好的算法基础。自治计算的系统结构如图 1 所示，主要包括 AOC 问题对象、AOC 计算方法、AOC 效能评价和 AOC 建模方法等四个部分。

2.1 自治计算方法概述

自治计算作为一种复杂系统建模和计算问题求解的方法，可用以解释、模拟并刻画复杂系统的机制、行为和特征。本节将介绍自治计算方法的基本构成和特性，内容主要包括：（1）自治计算的构成要素；（2）自治计算的特性。

[iv] 在自治计算中自底向上“bottom-up”的含义是相对的。针对不同问题的背景，自底向上可以指代系统不同层次的对象，在计算演化过程中其也是一个动态的概念，可以随着计算的要求而变化调整。自治计算中并不会预先定义或指定对象所处的系统层次属性。

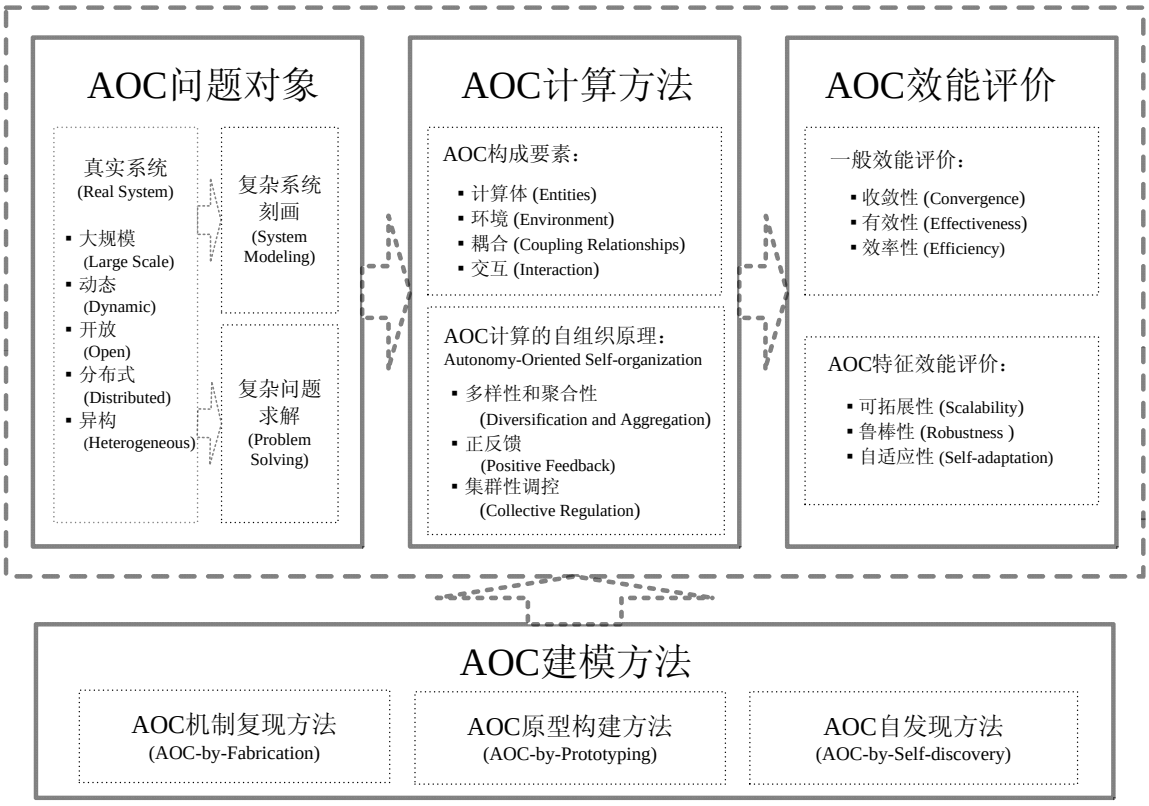


图 1. 自治计算系统结构

2.1.1 自治计算的构成要素

自治计算的构成要素包括：计算体（entities）、环境（environment）、耦合关系（coupling relationships）、交互（interaction）。自治计算的系统是由一组具有自治行为的计算体和其所处的环境组成。计算体可与其他计算体或环境交互信息，并可独立的调整或改变自身状态（state）与行为（behavior）。多个计算体在系统耦合关系的约束下，实现系统行为和结构的自组织。

2.1.2 自治计算的特性

自治计算是一个基于自治计算体的自底向上的计算范型（bottom-up computing paradigm）。在自治计算中，所有计算体的自治行为可产生系统层面上的涌现行为（emergence），即系统的整体行为模式将取决于计算体组织结构与行为模式的自组织（self-organization）过程。

自治计算的基本特性主要包括以下方面：

a. 自治性（Autonomy）

自治计算的自治性是指计算体的自治性，即计算体是一个不受其他计算体或外界控制，具备独立行为决定能力的个体。计算体通过与其他计算体或外界环境之间的交互，根据自身的状态和效用，在计算体行为规则的约束下，独立决定自己的后继状态

和行为。即系统不存在集中控制的机制（central control mechanism）来指挥或协调计算体或计算体之间的行为。

b. 自组织性（Self-organization）

自治计算中自组织性是指计算体与计算体、计算体与环境之间的交互行为。自组织性是系统结构和行为演化的推动力量。自治计算中自组织原理包括：正/负反馈机制、多样化和聚合性机制、集群性调控。通过自组织中的正/负反馈机制，系统可以强化或者减弱某种行为效果。多样化机制保证了系统行为的多样性，为系统通过选择实现“进化”提供了可能。聚合性机制有利于系统朝期望目标加速收敛。而集群性调控保证了系统的行为模式是向期望方向进行演化。

c. 涌现性（Emergence）

自治计算系统中会出现一些具有宏观特征的涌现行为（emergence）。涌现行为是指计算体之间的互相作用会在系统层面上产生某种无法预知的复杂模式（complex pattern）的结果。涌现并不是系统中计算体行为的简单叠加，而是系统中所有计算体行为的集群效应，是系统动态演化的结果。

涌现行为是计算体之间自组织的结果，是系统中所有计算体的集群行为（collective behavior）而非建模时计算体的预设行为。自治计算的一个重要应用是将计算中的涌现行为作为问题求解的计算手段，为问题求解提供高效的求解方案。如何通过调整计算体自治行为而促使系统出现我们所需要的涌现行为，也是自治计算中的一个重要内容。

d. 异步性（Asynchronization）

在自治计算中，计算体存在于一个异步计算环境中。计算体获取外界信息、实施自治行为的过程并不存在系统的集中同步机制。在分布式问题计算中（如调度和优化问题），计算体的行为会遵循预设的协议进行异步交互。

e. 自适应性（Self-adaptation）

自适应性是指自治计算系统可以根据期望目标的变化而产生新的行为模式和组织结构。这种自适应性体现在计算体本身所具有的系统期望目标的适应性调整机制。当系统期望目标发生变化时，在系统集群性调控的约束作用下，计算体可以自适应性的调整自身的行为和状态以期趋近新的系统期望目标。基于自治计算体适应性调整机制的集群效应，自治计算系统也能演化出新的行为模式和组织结构，以适应系统期望目标的变化。

2.2 自治计算的建模方法

在不同的复杂系统中，根据要解决的不同问题目标，AOC 有以下三种基本建模方法：

a. AOC 机制复现方法（AOC-by-Fabrication）

旨在通过复现或者利用真实世界中的自组织集群行为设计一种通用的问题解决方案。该方法是首先需要对系统的工作机制进行一定程度的认知，然后在模型设计的过程中进行简化、抽象和利用这些机制。这一思想也广泛的应用于其他领域的研究，如 L

系统（L-system）模拟复现了自然行为；蚁群计算[29]中模拟蚂蚁的觅食行为[30]；自私基因算法（selfish gene algorithm）[31][32]和文化算法（culture algorithm）[33]分别描述了自私行为和人类社会中的文化现象。与上述方法类似，AOC 利用自组织集群行为解决了图像分割问题[10][12]。

b. AOC 原型构建方法（AOC-by-Prototyping）

旨在研究理解复杂系统背后的工作机制。自治计算系统建立一组自治计算体集群，并模拟它们的自治行为和交互行为，从而建模复杂系统。通过一个不断尝试的过程，最终使系统变得足够的生动形象。自治计算原型构建方法是基于系统可见的特征，来分析测定系统背后的行为机制。这一思想在众多研究领域广泛应用，如社会现象研究[34]、交通流量和交通堵塞研究[35]、群体控制和群体行为[36][37]研究等。网络行为特征的刻画[16]就是 AOC 原型构建方法的一个应用例子。

c. AOC 自发现方法（AOC-by-Self-discovery）

旨在自发的发现或揭示问题的复杂系统问题的解决方案。在原型构建方法中，尝试过程被自治行为自发地取代。从另一个角度来说，目标系统期望的涌现行为与当前涌现行为之间的差异通过负反馈机制来影响计算体的自治行为，从而促使系统出现有助于问题求解的涌现行为。一些体现了自适应能力的进化算法就是此种方法的成功运用。自治计算自发现构建方法强调计算可以根据环境或者问题自适应式的调整，也就是说计算本身是自治演化的。自适应的思想在很多研究领域得到了广泛的应用，如自适应进化计算[38][39]就提出了系统中的参数（如变异率（mutation rate）、变异因子（mutation operator）、交配率（crossover rate）以及种群数量（population size））应该是自调整的。进化性扩散优化计算[8][9]也是自发现构建方法的应用例子。

综上所述，在实际问题求解时根据目标问题的复杂度和不同的机制理解程度，可以选择不同的自治计算方法：当目标问题相对简单、目标系统的运行机制已经被完全被理解，采取 AOC 机制复现方法（AOC-by-Fabrication），即形式化目标问题和实体运行机制，并在实际应用中验证模型的准确性；当具有系统的部分知识且有实际系统做参照时， 采取 AOC 原型构建方法（AOC-by-Prototyping），即根据现有知识构建初始模型，在运行过程中通过与实际系统不断比较，进行修正最终得到较为符合实际运行结果的模型；若对目标问题或系统有相对充分的了解并能使修正过程自动化时，采取 AOC 自发现方法（AOC-by-Self-discovery）。这三种方法并不是完全孤立的，AOC 机制复现方法是另外两种方法的基础；AOC 原型构建方法是 AOC 机制复现方法的不断重复；AOC 自发现方法是 AOC 机制复现方法和 AOC 原型构建方法的自动化实现。

2.3 自治计算的应用与效能评价

本节通过自治计算在图像分割问题[10][11][12][13]，约束满足问题[2][3][4][5][6]、以及环境建模问题[18]上的应用，使读者初步了解自治计算在问题求解方面的应用以及自治计算的效能评价。

a. 图像分割问题（Image Segmentation Problem）

图像分割问题[40][41]是指在一副图像中标注出具有相同属性的区域，如灰度值。Liu 等[10][12]提出了基于自治计算模型的搜索方案，较好的解决了图像区域分割的问题。该

方案在图像动态变化情况下亦能取得较好的搜索结果。在该问题中，计算体作为区域边界的搜索载体，图像像素点作为自治计算的环境。计算体不断的检查其周围的像素环境，并根据其所处环境边界特征的评估，实现计算体行为的自治选择。如果计算体当前所处的位置满足图像边界触发条件，计算体将此位置标志为边界，并激发计算体复制机制，产生新的搜索计算体。

b. 约束满足问题（Constraint Satisfaction Problem, CSP）

N 皇后问题是一个典型的约束满足问题。Han 等[2]利用自治计算模型为一般的约束满足问题求解提供了一个以较小计算代价而获得近似解的求解方案。在 N 皇后问题中，自治计算模型中每个计算体代表一个系统变量（皇后），二维棋盘代表系统环境。于是计算体（皇后）在环境（棋盘）中的位置就代表了约束满足问题的一个解方案。系统中计算体采用“适者生存”与“丛林法则”的行为规则，不断的调整其在环境中的位置，通过系统自组织行为，逐渐逼近系统的全局最优解。

c. 环境建模问题（World Modeling）

Liu 和 Wu[18]提出了一种基于自治机器人群的环境创建问题解决方案。将一组运动机器人放置在系统环境中，以较低的代价完成环境创建任务。通过机器人计算体之间的信息交互，系统的自组织行为使机器人协同完成环境创建任务，通过自适应机制使计算体不断动态调整它们之间的协同行为，从而使环境创建任务得以高效地完成。

在上述问题求解中，自治计算都是通过构建自底向上的系统模型，如在 N 皇后问题中，计算体代表系统变量（皇后）；在图像分割问题中计算体是具有边界搜索能力的载体；在环境建模中计算体是环境探测机器人。通过利用计算体的自治行为从而最终获得系统的全局解决方案。有以上应用例子可知自治计算是一种新颖的自底向上的面向计算体自治行为的计算模型，为解决现实生活中大规模、分布式、异构、动态、开放系统的问题求解提供了有效的计算手段。自治计算的效能可分析如下：

a. 拓展性（Scalability）

拓展性要求算法在处理大规模系统的时候不会因为系统规模的增大而使算法复杂度增加。也就是说计算复杂度与系统规模之间是互相独立的。在 N 皇后问题中，自治计算可以有效的解决 N=7000 的计算问题，然而占用的系统计算资源却十分有限。在图形分割问题中，自治计算体复制机制可以保证边界搜索迅速有效的找到图形分割区域，算法效率不会受到图形大小的较大影响。

b. 鲁棒性（Robustness）

鲁棒性要求算法在处理动态系统的时候对于系统波动和外来扰动具有一定的冗余特性，即算法的计算效果不会因为波动和扰动而发生明显的变化。在图像分割模型中，搜索计算体的复制与重激活机制使边界识别过程可以不受到扰动的影响。在环境创建问题中，扰动可以影响局部的机器人计算体的行为，但是通过机器人之间的交互作用，扰动的影响会逐渐弱化，而不会影响系统全局的计算效果。

c. 自适应性（Self-adaptation）

自适应性要求算法在处理开放系统的时候可以根据期望目标的变化而调整自身组织形式与行为模式，从而使算法能够在不同的系统目标下都能有效的解决计算问题。在环境创建问题中，机器人计算体通过自适应机制不断的调整计算体之间的合作行为，从而高效迅速的完成环境创建任务。

3 自治计算系统设计

在对自治计算有了一个初步的认识之后，本章具体介绍自治计算系统的设计方法。首先，在 3.1 节我们将介绍自治计算系统的基本组成要素，包括计算体与环境，自治性，耦合关系，交互。这些要素的核心是局部自主性建模（local autonomy modeling），这是所有自治计算模型的共同结构。其次，在 3.2 节我们着重介绍自治计算系统设计的基本原理——自组织计算性设计（self-organized computability design）。这一基本原理强调了自治计算利用复杂系统中自组织和涌现行为（见 2.1 节）来设计计算复杂系统，是自治计算系统设计的关键所在。最后，我们需要对设计完成的自治计算系统进行多方面评估。我们将在 3.3 节介绍自治计算系统的效能评价机制，包括一般效能评价与特征效能评价。

3.1 自治计算系统的组成要素

本节介绍组成自治计算系统的基本组成要素以及它们之间如何相互作用。自治计算系统模型由一组自治计算体（autonomous entities）及其所依赖的环境（environment）组成。计算体在耦合关系（coupling relationships）的制约下探知环境的状态，并基于一定的行为准则（behavioral rules）独立选择对外的行为表达，从而通过计算体的群体行为影响或改变系统状态。计算体与环境之间的信息交换与行为表达，在自治计算的系统中形成一个信息反馈回路。通过这个反馈回路，计算体和计算体之间、计算体和环境之间就产生了直接或间接的交互（direct and indirect interaction）行为。

3.1.1 计算体与环境（Entity and Environment）

计算体与环境是自治计算系统构成的两个基本模块（building blocks）。Box 1 给出了这两个基本模块在复杂系统刻画和复杂问题求解中的具体含义。

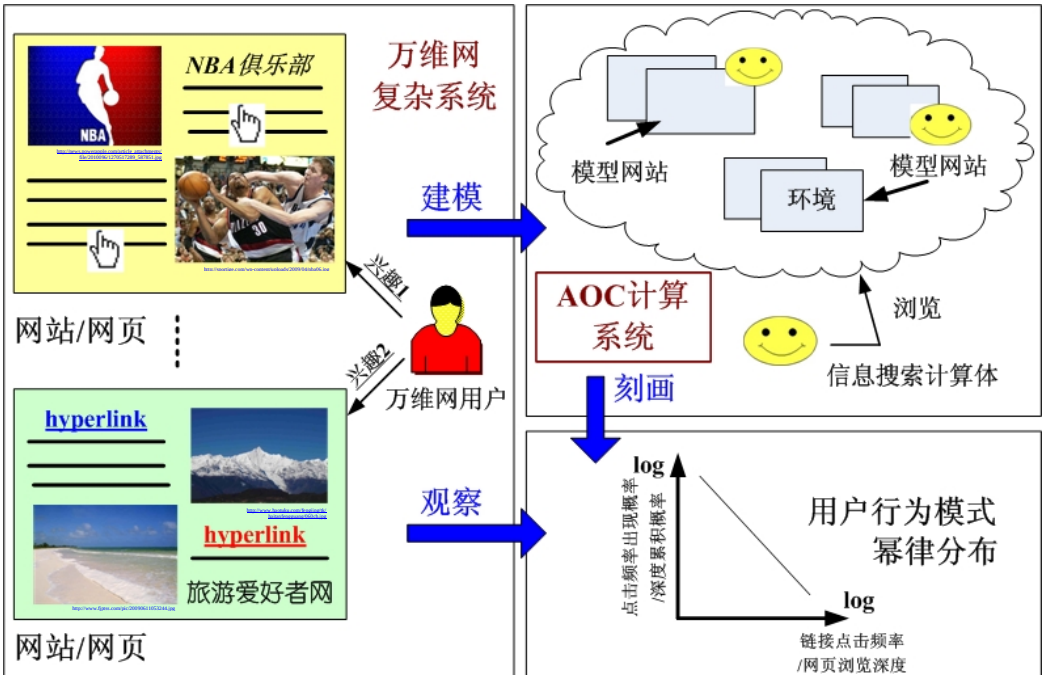
(1) 计算体 （Entity）

现实世界中的复杂系统，如生态系统、免疫系统、人脑、股市等，都由相应的具有自主行为的计算体所构成，比如生物物种、免疫细胞、神经元、交易员。自治计算系统也包含一组类似的自治计算体（autonomous entities）。自治计算体的核心在于自治性（autonomy），即计算体自身的状态或行为的改变是由计算体自身的原子行为（primitive behaviors）与行为规则（behavioral rules）独立决定的，而非受到外界因素控制。同时，在自治计算系统中，计算体（entities）可以受外界环境和周边其他计算体的影响。一方面，计算体通过获取外界信息，改变自己的内部状态和对外行为。另一方面，计算体的对外行为又可以影响其所处的环境状态和其周边的其他计算体。例如，在飞鸟迁徙中，每只飞鸟都是一个自治计算体。在飞行中，它们不断的从周围环境中获取鸟群的飞行速度与方向，并根据与周围其他飞鸟的相对位置，不断的调整自己飞行速度和方向，这个调整行为是飞鸟独立判断的结果，而非受到来自外部的指令或信号。

在自治计算系统中，作为系统构成的基本单元，计算体具有更广泛的涵义。

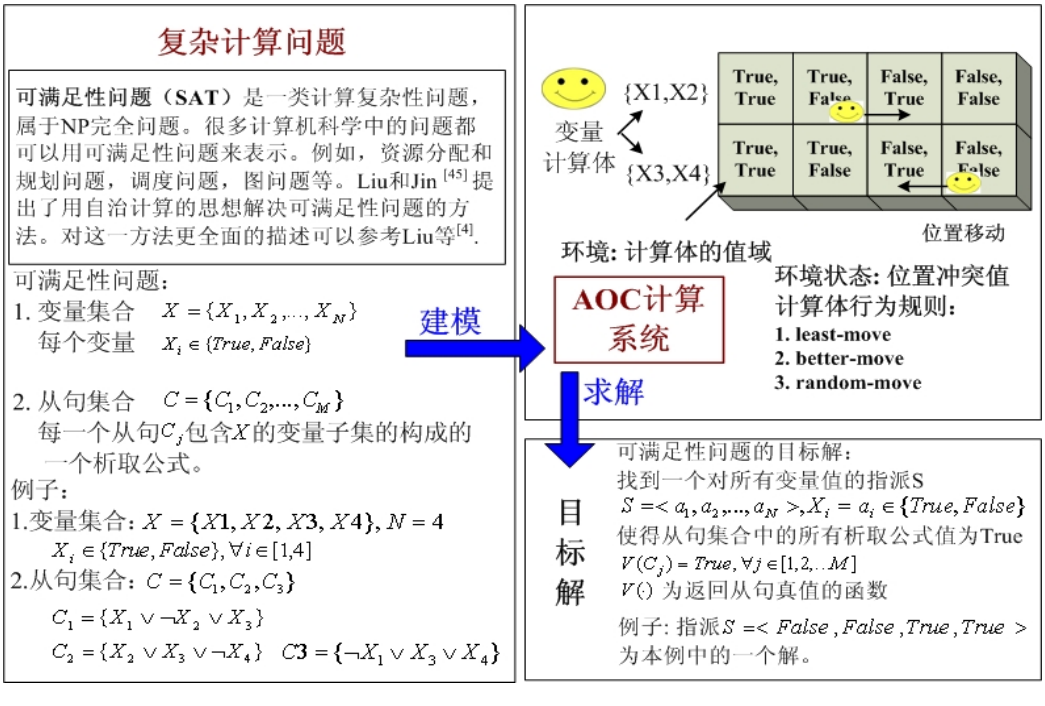
Box 1：计算体与环境

例 1：万维网用户行为刻画（复杂系统刻画例示）



(a)

例 2：可满足性问题求解（复杂问题求解例示）



(b)

a. 计算体的多层次性

在自治计算系统中，计算体可以定义在系统的不同的层次上。如计算体可以是系统的基本构成单元，也可以是由若干单元组成的子系统。不同层次的计算体可以互相包含，如 A 类型的计算体可由 B 类型的计算体所构成。计算体之间交互可以出现在不同层次之间。

例子：Hu 和 Liu 在[42][43]中使用自治计算的建模思想刻画 NBA 球队间的竞赛。在 NBA 竞赛模型中，球员可以定义为计算体，球队也是计算体，球队计算体是由球员计算体构成的。计算体之间的交互可以发生在球员之间，球队之间或者球员与球队之间。

例子：机器人运动刻画中（见 Liu 和 Qin[21][44]）的基础动作如走、跳等可以视为导致机器人行动的计算体，而组成动作的基础动作元素可以视为另一层次的计算体。不同层次计算体内部和之间都产生相互影响。

b. 计算体的类型多样性

自治计算系统中，计算体可以分为为概念型和功能型两大类。例如，Box 1 图（a） 中的万维网信息搜索计算体对应于现实中网页浏览用户（功能性的实体）。在可满足性问题（satisfiability problem, SAT）求解中，一组命题变量的集合定义为计算体（见Liu等[45]）属于概念计算体（见Box 1 图（b））[v]。

c. 计算体的功能多样性

一方面，计算体可以是功能单一的个体，例如，基本“刺激-反应”（stimulus-response）规则下决定采取何种行为的计算体，如迁徙鸟群中的飞鸟。另一方面，计算体也可以是具有复杂认知功能的“理智体”，如市场交易中的交易员。

d. 计算体的异构性

在自治计算系统中，计算体既可以是同质（homogenous）的，如具有相同原子行为的计算体；也可以是异构（heterogeneous）的，如在 Zhang 和 Liu[17]提出的解释 HIV 病毒侵蚀人体免疫系统过程的模型中，有 HIV 病毒、T 细胞、自然细胞等多种异构的计算体类型。

e. 计算体的形成动态性

从形成角度看，计算体的结构（structure）和行为（behavior）既可以预先定义，在系统演化过程中保持静态不变（基础型计算体）；也可以由涌现产生，随着系统的演化而动态改变（涌现型计算体），如 Facebook 中自发形成的兴趣小组、机器人运动刻画中（见 Liu 和 Qin[21][44]）的行为基模（behavioral motif）等。

综上所述，在自治计算系统中计算体的建模具有极大的灵活性和一般性。其具体含义与我们要解决的问题背景相关，是建模过程中对于系统抽象的结果。

---

[v]由于 AOC 重点关注自治体（entities）的计算本质，因此我们称其为“计算体”，使其与现实中的实体进行区别。

(2) 环境 （Environment）

环境（environment）是自治计算系统的另外一个重要组成部分。系统中除计算体以外的部分都可视为环境。Liu 等[25]指出，环境在自治计算系统中有三个主要作用。

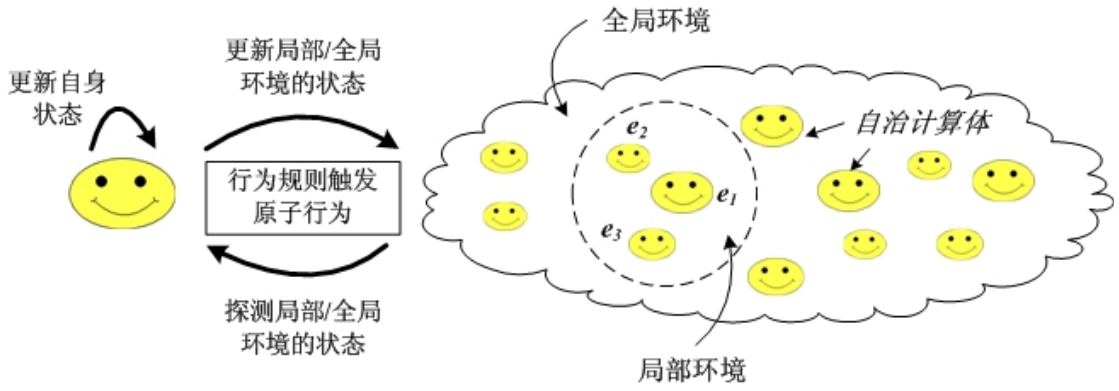
a. 以静态角度而言，环境是自治计算体的存在空间和活动区域。

b. 从动态角度来看，环境可以作为计算体之间信息交互的渠道和媒介。一方面，环境状态可以被计算体所感知；另一方面，计算体的自治行为也能影响系统状态。因此，在自治计算系统中，环境状态是动态变化的。例如：在 Liu 等[10][12]提出的基于自治计算的图像边界搜索识别问题中，计算体可以获知其所处图像位置中局部像素的灰度值，如果当前像素属于一个均匀区域，计算体会在此像素环境中设置标记。此时，像素环境就是计算体之间进行间接交互的媒介。

c. 环境的第三个作用就是可以对计算体的信息同步提供参照空间。例如，在同一个时刻，环境可以负责更新所有计算体所处的位置。

从与计算体进行信息交换的角度出发，环境包含两层含义：（1）计算体的局部环境（local environment），比如计算体可感知范围内的邻居计算体；（2）系统的全局环境（global environment），指整个自治计算系统的环境状态[vi]。环境与自治计算体间的关系参见Box 2。

Box 2：自治计算系统中环境与自治计算体的关系



一方面，自治计算体从局部或全局环境中获取信息（比如，其他计算体的状态，环境中资源的信息等），更新自己的状态。另一方面，计算体的行为又可以影响其所处的环境以及其他计算体。具体体现在， 计算体根据自身的自治性，如定义的行为规则，触发特定的原子行为，更新局部和/或全局环境的状态。

[vi]自治计算中局部和全局环境是一组相对概念，取决于问题建模的需要。以视角而言，某个计算体的局部环境可以是其他计算体的全局环境；以层次而言，某个层面的环境（如球队士气）是一类计算体（球员）的全局环境（见 Hu 和 Liu[42][43]），也可以是另一类计算体（赛季）的局部环境。

(3) 计算体自治行为（Autonomy）

自治计算系统中，计算体的自治行为是指计算体从环境中或从其他计算体获取信息，基于自身的行为规则（behavioral rules）独立决定其下一时刻的状态（state）或行为，从而影响或改变环境以及其他计算体的状态。Liu [24] 指出计算体的自治行为可以概括为以下步骤：

a. 获取计算体当前的状态（state）和效用（utility），以及外部环境的状态，包括局部环境（local environment）和/或全局环境（global environment）。

b. 设计计算体的评估函数（evaluation function）与效用函数（utility function）。

c. 基于计算体的评估值（evaluation）与效用（utility），在计算体行为规则的约束下（behavioral rules），决定下一时刻的内部状态（internal state）和对外原子行为（primitive behaviors）。

d. 影响外部环境或其他计算体。这种影响包括如下三种可能：（1）状态变换；（2）效益更新；（3）对外行为或交互。

计算体的自治行为使计算体和环境之间形成一个信息反馈回路。通过这个反馈回路，计算体之间、计算体与环境之间形成直接和间接的交互。在一定的模型设计下，这种交互会产生非线性效果，从而使系统产生目标涌现行为，趋向或达到全局目标。

3.1.2 耦合关系（Coupling Relationship）

耦合关系是自治计算系统的基本概念之一，它规定了自治计算系统中计算体之间，计算体与环境之间的约束关系。我们可以从以下两个角度来理解耦合关系。

a. 从形式上看，耦合关系不仅体现在结构上，即结构性耦合（structural coupling）也可以体现在功能上，即功能性耦合（functional coupling）。前者如不同网页之间存在的超链接结构（限制浏览范围），后者如一支蓝球队中不同球员具有各自的功能性角色（前锋、中锋、后卫等相互配合）。Box 3 对这两种耦合关系作进一步讨论。

b. 从行为上看，耦合关系既可以是预先定义的，也可以在行为过程中动态改变。以社会网络为例，如果将人作为计算体，人与人之间的社会关系作为耦合关系，那么亲子关系一旦形成之后就固定不变，比如好友关系则可以动态改变。

3.1.3 交互（Interaction）

交互是自治计算产生自组织现象的必然条件。系统的涌现行为即是源自于计算体之间基于结构性、功能性耦合关系的局部交互（local interaction）。自治计算系统包含两种类型的交互：即直接交互（direct interaction）和间接交互（indirect interaction）。

a. 直接交互（Direct Interaction）

直接交互是计算体通过直接与其他计算体交互获取信息的行为。例如，日常生活中的语言交谈，书信来往都属于直接交互。

b. 间接交互（Indirect Interaction）

间接交互是计算体通过环境，实现间接获取系统信息的行为。例如，通过发表论文、书籍这一媒介，可以实现所处不同地域研究人员的知识共享。

3.1.4 总结概括

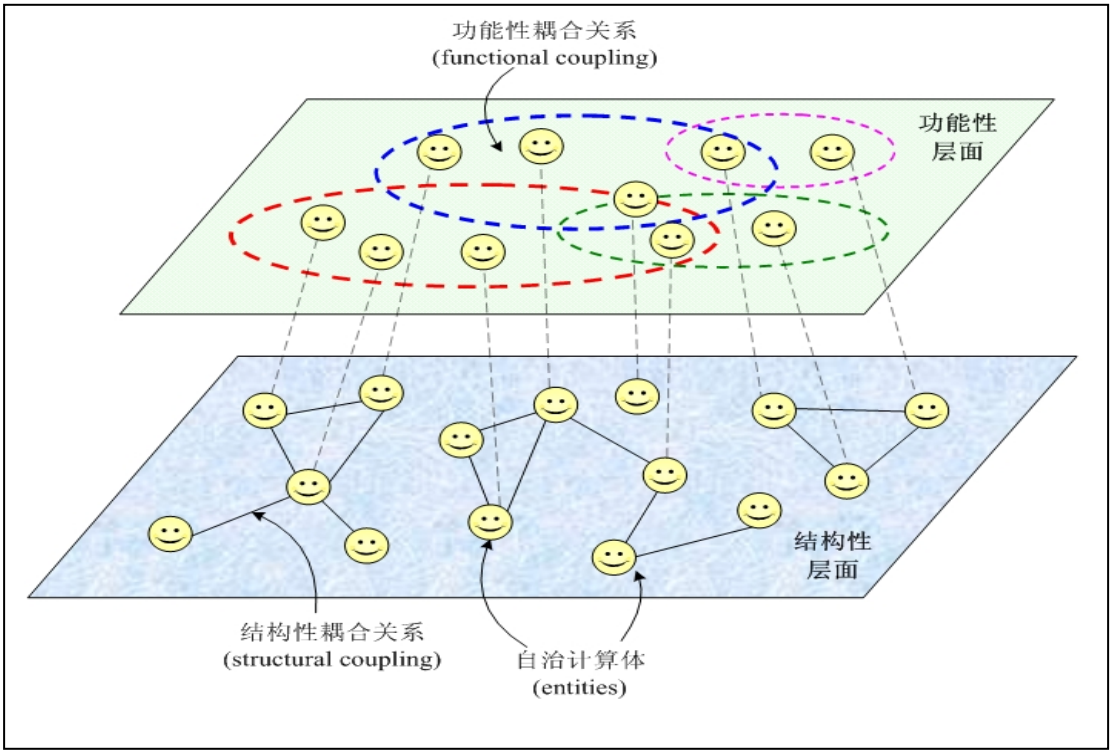
本节主要介绍了自治计算系统的基本组成要素，即：计算体与环境、耦合关系、交互， 以及这些要素所包含的具体涵义。这些基本要素共同构成自治计算模型的基本框架。为在这一框架上实现复杂系统建模或解决复杂计算问题，下节将介绍系统设计的基本原理。

Box 3：自治计算系统中的耦合关系与交互

例 1. 结构性耦合与计算体-计算体交互

在信任关系（trust relationship）网络形成的刻画过程中（见 Liu 和 Qiu[58]），计算体（比如人与人）之间的耦合关系表现为与邻居不同程度的信任关系。这对应于社会网络中的结构性耦合。与现实中的理性人一样，计算体倾向于选择与自己信任关系强的其他计算体进行交互，如发送、回应服务请求等。交互的结果会进一步更新计算体之间的信任（耦合）关系。比如计算体 A 帮助 B 成功的完成一项服务，则 B 对 A 的信任（耦合）关系得到增强；反之，如果完成任务失败，则他们之间的信任关系将削弱。

例 2. 结构性耦合，功能性耦合与计算体的关系



上图显示结构性耦合与功能性耦合的关系。计算体之间具有结构性耦合（例如：好友关系，用结构性层面中的实线表示）并不代表他们具有功能性耦合（例如：属于相同的工作团队，用功能性层面中不同大小的椭圆表示）；而不具备结构性耦合的计算体之间可以有功能性耦合关系（例如：临时形成的战略合作关系，图中体现在属于同一个椭圆的计算体之间可能没有实线相连，即不具有结构性耦合关系）。

3.2 自治计算系统基本设计原理

正如 2.1 小节的介绍，自治计算方法基于复杂系统中自组织和涌现的基本原理刻画系统、分析问题。本节将对这些基本原理在具体模型设计中的体现—自组织计算性（self-organized computability），做更详细的介绍。

3.2.1 自组织计算性的必要条件

为了实现对系统宏观涌现行为建模，以及对现实复杂问题求解，我们需要借助自组织过程所伴随的非线性（non-linearity）特征。换句话说，线性系统肯定不会产生涌现行为，因为线性系统的组成成分不会有新奇（Radical Novelty）[46]的产生。类似的，线性算法通常无法为计算复杂性问题（比如 NP-hard 问题）提供令人满意的解决方案。从获取非线性特征的角度出发，我们提出自组织计算性的必要条件。

Liu[24] 指出: “在一个自治计算系统中，自组织计算的必要条件（necessary condition）在于多样性（diversity）以及可计算结构（computable configurations）的涌现性。这两点可以由如下两个过程来实现：（1） 小范围和大范围的随机化探索（short and long-range stochastic exploration）过程；（2） 基于正反馈 （positive feedback） 的加速聚合（aggregation）过程。”

下面我们进一步详细介绍自组织计算的必要条件以及与此相关的几个关键要素。

a. 多样性和聚合性假设（Diversification and Aggregation Postulate）

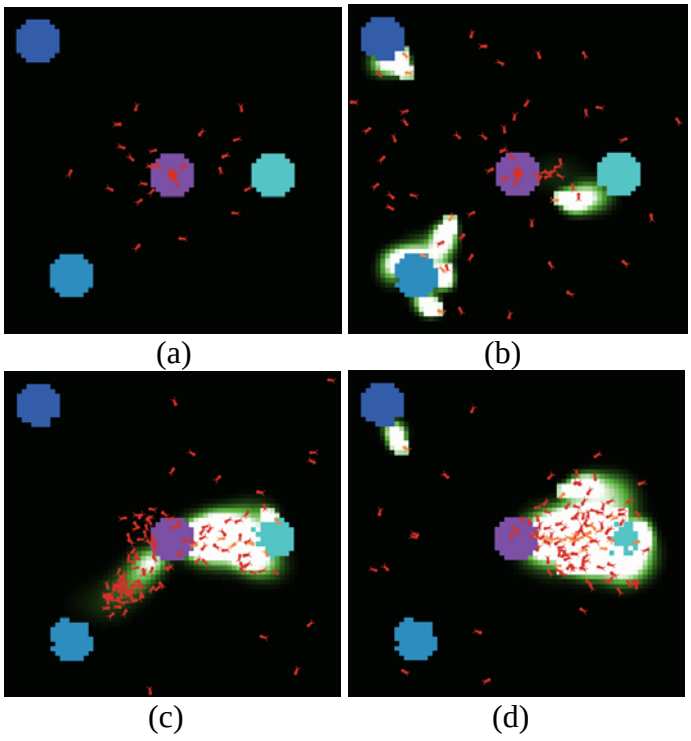
多样性和聚合性作为自组织计算必要条件的两个方面存在一定的互补平衡关系。Liu[24]指出，一方面，小范围和大范围的随机化探索为自治计算系统的计算过程创造足够的多样性结构（diverse configurations）。另一方面，正反馈机制对引导系统到达一个临界状态（critical mass）或者阈值（threshold level）起到关键性作用。从这个角度出发，随机性与聚合性的适当平衡对系统达到临界状态就显得十分关键。此时的系统处于稳定性的界限（boundaries of stability），或者混沌的边缘（edge of chaos）[47][48]。在这一临界状态，自治计算系统呈现出复杂涌现行为，体现在系统的多样性、信息流动、计算效率、以及/或者资源的节约都被最大化。Box 4 体现了多样性与聚合性对系统演化的作用。

b. 交互 vs. 多样性

Liu [24]指出，在自治计算系统中，局部交互与涌现行为呈现出一定的互逆法则（inverse rule）。正如自治计算在约束可满足性问题的工作[6]中所揭示的，变量计算体（Box 1 图（b））之间过多的交互会减缓甚至破坏整个系统找到解的过程。因此，为了保障自治计算系统的多样性和鲁棒性，我们应当合理设计计算体之间发生交互的条件及频率。

3.2.2 自组织计算性的充分条件

体现自组织特征的复杂系统具有系统演化多稳态性（multi-stability）的共通点，即由于结构的出现源自初始化时的随机扰动（fluctuation），而任意初始的随机扰动都可能在随后的演化过程中被放大（amplification），导致系统的最终状态受制于初始状态的随机扰动。在自治计算建模过程中，自组织的这一特性需要避免，因为我们在创建模型或求解问题时有明确的刻画目标或者求解目标。如果不采取某种机制对随机搜索和基于正反馈的聚合过程加以调节，很难保证自治计算系统会收敛到我们所期望的目标。针对这一目的，我们提出自治计算系统自组织计算性的几个充分条件 。



Box 4：随机化探索和基于正反馈的聚合性机制

图 (a) – (d) 展示了蚁群集体觅食的过程。图（a）中表现了蚂蚁（小点）的从蚁巢（中心圆）出发进行随机化探索。图（b）表明某些蚂蚁找到了分布在环境中的三个食物源（3 个周边圆），并留下信息素（pheromone）（光斑区域）。图(c)和(d)显示由于信息素的作用，距离蚁巢最近的食物源聚合绝大部分的蚂蚁。图(d)展示由于正反馈的聚合作用，离蚁巢最近的食物源被迅速消耗完毕。但是大范围的随机搜索确保了新的食物源能够继续被蚂蚁群体发现和利用。（图形由 NetLogo 软件绘制[49]）

Liu[24]指出：“在一个自治计算系统中，自组织计算性的充分条件（sufficient condition）在于计算体的设计应当在局部自主性建模过程中隐含特定的全局影响（global influence）或倾向（bias），并且从长期的角度来看这种影响和偏好对自治计算系统趋向于全局目标保持一致。需要注意的是，这一条件中的影响和偏好并不强加于计算体而只对其行为产生影响。”

a. 集群性调控（Collective Regulation）

为了获得系统期望的目标或者全局解，计算体的局部自主性（local autonomy）需要在其理性化（deliberative）或反应式（reactive）的行为交互中，通过集群性调控得到控制。在具体的模型设计过程中，这种集群性调控可以通过适当的外部环境评估函数 F 和内部效用评估函数 u，或者局部行为规则集 R 来实现（参见 Box 5）。此时，计算体具有的某些特征在更大的时间范围和空间规模上，将与系统期望的全局目标解保持一致，或者偏向于达到这一目标。

b. 正反馈（Positive Feedback）

上文的集群性调控，通过在微观层面上规范计算体的行为，使宏观层面上涌现出的模式能够与自治计算系统的全局目标保持一致。Liu [24]指出，在涌现计算的过程中，正反馈的主要作用在于触发并加速系统的全局涌现过程，从而使得集群性调控对计算体局部自主性的影响，比如收敛到全局最优解，变得更加有效果。在自组织计算性充分条件的辅助下，正反馈机制将偏好于某些系统期望的特征，并且进一步对其进行放大。与此同时，那些与系统期望相背离的状态将被迅速的排除。体现这一过程的例子参见 Box 6。

Box 5：集群性调控（行为规则体现）

本例的全局目标在于刻画鸟群飞翔模式，是 AOC 机制复现方法的具体示例，由 Reynolds[50] 提出。

计算体：飞鸟
环境： 局部领域内的邻居计算体

行为规则：

- 分离（separation）：如果计算体与最近邻居计算体距离小于阈值，则调整距离远离邻居。
- 保持方向（alignment）：如果计算体的方向与周围邻居的平均方向不一致，则向平均方向调整。
- 聚拢（cohesion）：如果计算体与周围邻居距离太远，则向邻居靠拢。

图片来源: http://www.treehugger.com/ducks-flying-v-formation-migrating-photo1.jpg

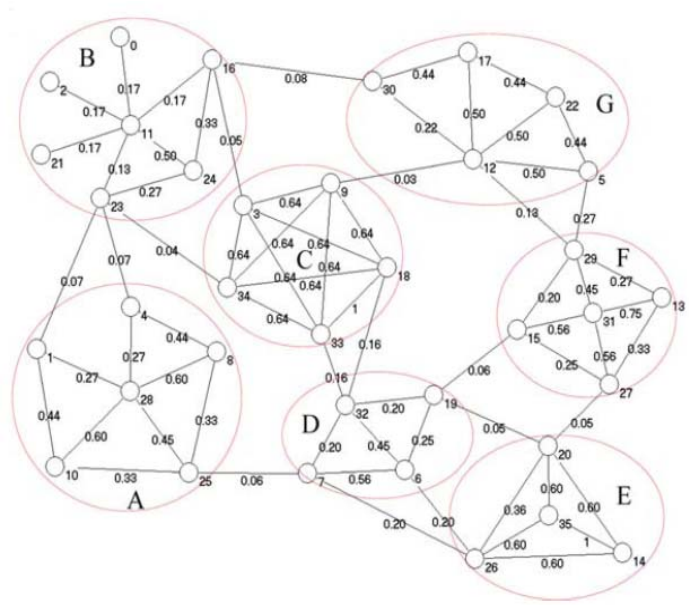
图片来源: http://www.ocracokeonline.com/ocracoke/Ocracoke%20ferry/slides/seagull%20flock.JPG

集群性调控：在这三条局部行为规则的调整下，计算体的行为仅受到周围邻居的飞行方向以及与他们之间距离的影响。即邻居的行为对计算体起到集群性调控（collective regulation）的作用。使得自治计算系统在宏观层面涌现出协调一致的飞翔模式。

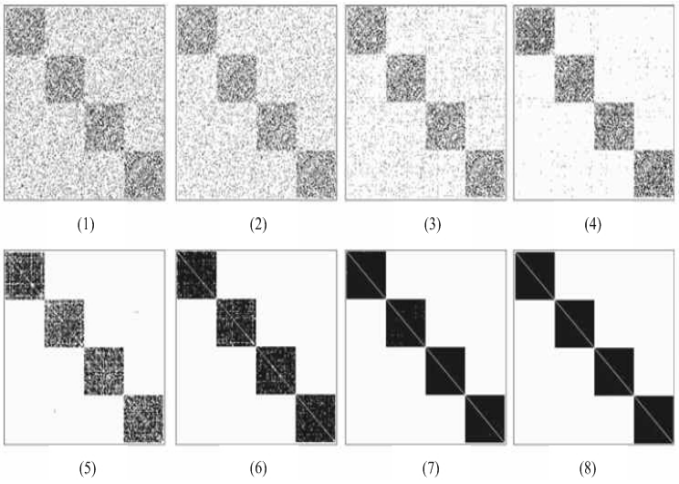
以计算的角度而言，自然界中飞鸟的集群飞翔不仅是宏观有序的涌现模式，也是优化计算的结果。例如，在多种鸟类飞翔中都观察到的“V 字型”模式，可以起到拓宽视野，节约飞行能量消耗的作用[51]。

3.2.3 总结概括

这一小节介绍了实现自组织计算的必要条件和充分条件。计算性的必要条件包括随机化探索和正反馈，两者对涌现行为的发生起到至关重要的作用。而自组织计算性的充分条件，包括在局部自主性建模过程中添加特定的影响和偏好，能够帮助自治计算系统达到“目标行为”。把这两者结合在一起就能够促使自治计算系统实现“目标性涌现行为”。这正是自治计算方法的关键目标所在。



(a)



(b)

Box 6：正反馈的收敛性规范作用

动态社会关系网挖掘中的正反馈机制 (来源 Yang 和 Liu [15] 图 4 和 7)。当今社会，随着移动设备如手机，PDA 等的普及，移动服务越来越丰富。本例中自治计算算法的目标在于挖掘出移动社会关系网中的兴趣相投的隐形社区（hidden community）。

图 (a) 显示了一个包含社区结构的网络。每个节点代表一个移动服务用户，每条边代表两个用户之间存在好友关系（如 Facebook 好友）。边上的数值代表两个用户的兴趣相似度。

图（b) 显示了自治计算算法在图（a）网络上进行社区挖掘的动态演化。每个子图显示了图（a）网络对应的邻接矩阵在某一时刻的值。子图中的颜色代表相邻两个节点相似度的大小（相似度越大颜色越深）。

如果两个节点属于同一个隐形社区，那么他们之间的耦合关系值就会在迭代计算中得到放大加强（颜色变深），相反，若非属于同一个社区，耦合关系值会逐渐弱化减小（颜色变浅）。这体现了正反馈的收敛性规范作用。

3.3 自治计算系统的评价

在设计完成自治计算系统和相关算法之后，我们需要对其效能进行全面评价。效能评价指标主要包含两类：一类是自治计算的一般效能评价，包括：（1） 收敛性（convergence）；（2）有效性（effectiveness）；（3） 效率性（efficiency）。这是我们衡量系统设计优劣的一般指标。另一类是自治计算的特征效能评价，包括：（1） 可拓展性（scalability）；（2） 鲁棒性（robustness）；（3）自适应性（self-adaptation）。这是自治计算系统设计中重点关注的指标。

3.3.1 自治计算的一般效能评价

一般效能特征描述了自治计算系统具有的趋近或动态稳定在系统期望目标附近的计算行为特性，包括自治计算的收敛性、有效性和效率性。

自治计算收敛性是指系统求解的动态过程最终能够饱和并稳定在系统解附近。自治计算的求解过程是一个非线性过程，自组织涌现行为（如分叉（bifurcation））为系统求解计算提供了解方案。自治计算的收敛性要求复杂系统的非线性求解过程在正/负反馈机制的作用下具有趋向饱和稳定在某一系统解附近的特性。

自治计算有效性是指系统动态求解过程最终能够收敛于系统期望目标附近。自治计算是建立在计算体的自治行为之上，而计算体的自治行为通过系统的正/负反馈机制以及系统的集群性调控作用，自组织并最终产生稳定收敛的系统解。自治计算的有效性要求计算体能够不断调整其自治行为特征，以缩小系统解与系统期望目标之间的差异，最终使系统收敛于系统期望目标。

自治计算的效率性是指自治计算的计算成本，其包含两个方面：（1）系统的计算成本，如计算时间；（2）自治体的计算成本，如自治体的效用更新和交互成本。自治计算过程是基于计算体自治行为的异步过程，通过不断的调整计算体的自治行为和自组织形式，促使计算过程出现有利于系统求解的涌现行为，从而降低系统计算成本，提高效率。

3.3.2 自治计算的特征效能评价

自治计算的特征效能评价是自治计算系统设计中的重点关注目标，也是自治计算能够解决具有大规模、分布式、异构、动态以及开放等特征的系统计算问题的原因之所在。包括自治计算的可拓展性、鲁棒性和自适应性。

(1) 可拓展性（Scalability）

a.可拓展性的衡量

在自治计算中，可拓展性包含两层含义：（1）以传统计算复杂性而言，可拓展性指自治计算算法是否能够在不同的系统规模下保持较好的计算效率。例如，在计算机病毒免疫的应用中，要求在动态变化的网络中识别出一系列重要节点（以节点度数高低进行衡量）作为免疫对象。传统算法（如目标免疫策略[53]，D 步免疫策略[54]等）无法在网络规模变大的情况下保证算法策略的计算效率。而由 Liu 等提出的自治计算免疫算法[59]，其识别重要节点的执行效率会随着系统规模（网络中节点数目）的增大而提高。（2）针对问题本身的复杂性或刻画目标系统的特征而言，可拓展性指自治计算算法是否能够有效的解决（刻画）不同复杂性的问题（系统）。例如，Liu[60]根据组合判定问题中发现的相变现象将问题解空间分为弱约束区域（under-constrained region）和过约束区域（over-constrained region）,自治计算算法能否在过约束区域有效地获取近似解或最优解是衡量系统可拓展性的一个标志。

b.实现可拓展性的基本设计原则

自治计算系统的可拓展性源于自组织性设计中的正反馈原理。系统要实现自我增强的聚合效应，一个关键的要素是正反馈循环中可利用的计算资源和计算能力。当系统的计算资源和计算能力随着系统的规模增长时，由于正反馈导致的计算中的非线性效果就能够被进一步的扩大，从而使系统的效能显著提高。例如，上文提到的自治计算免疫算法 [59]，正反馈由多计算体和环境之间的激励机制（stigmergy）[55]形成。而引发这一非线性特征的关键要素是计算体在环境中留下引导其他计算体搜索的信息素（局部环境中的重要节点标志）。当系统规模增大时，这一信息素能够在整个网络中更快的传播，使得更多的计算体能够利用正反馈带来的非线性提升自己的效能，进而非线性的提高系统效能，从而实现了可扩展性。

(2) 鲁棒性（Robustness）

a. 鲁棒性的衡量

鲁棒性主要衡量在外界扰动出现后，计算系统是否能够保持自身的计算效能。这也包括两个方面的含义：一是从复杂问题求解的角度，需要衡量自治计算算法在外界输入产生扰动时依然保证计算效能。例如，动态社会网络挖掘[15]中，不论底层的网络结构如何变化，自治计算系统总是能够迅速有效的识别出隐藏在网络中的社区结构。二是在复杂系统建模中，鲁棒性体现在系统宏观模式在外部输入条件的变化下是否具有稳定性。举例来说，在万维网用户行为模式刻画（Box 1 图（a））中，计算体浏览行为产生的宏观结果应当不受输入条件（如网页拓扑结构，计算体的数量等）的影响。

b.实现鲁棒性的基本设计原则

在自治计算系统设计中需要注意两个关键要素。一是自治计算计算复杂系统由多计算实体组成，这使得系统在功能上具有一定的冗余特性，从而能够应对外界环境的扰动。如在蚂蚁（ant colony）寻找食物的过程中，部分蚂蚁被其他生物捕食并不会影响整个蚁群的搜索效率。二是正反馈的自我增强性（self-reinforcement）能够实现鲁棒性。如自催化集合（autocatalytic sets）中，催化环中的各个部分会相互催化从而增强各自的适应度。当出现外部的扰动（如某种物种数目减少）时，系统能够通过催化环的相互催化作用抵御这种扰动，从而实现系统整体的鲁棒性。

(3) 自适应性（Self-adaptation）

a. 自适应性的衡量

自适应性体现在，当建模刻画的系统和求解问题本身的结构发生变化时，自治系统能否自主的调整自身的行为规则和组织结构，以主动适应这些外部环境和/或系统目标的变化。此时，不同层次自治计算体自身行为和结构的动态演化，在宏观层面上使整个自治系统产生上述自动调节的适应性行为。

b. 实现自适应性的基本设计原则

自治计算系统和算法获取自适应性的关键在于以下两点：一是设计计算体的自治性时添加动态性机制。这使得计算体自身的行为规则和自治体之间的耦合关系能够动态演化以适应环境的改变。演化计算、学习算法的思想能够提供一定的借鉴和启发。二是计算体行为的动态改变需要与自治系统整体目标（如环境改变造成新的宏观模式）相协调。具体的，需要计算体的个体行为能在集群性调控、正反馈等自治计算基本原理的作用下与系统全局目标相适应。

4 自治计算与其他相关领域的区别与联系

4.1 自治计算范型 （Autonomy-Oriented Computing Paradigm）

在计算科学范型发展进程中，以下几种范型对计算发展有重要意义：命令型（imperative）、功能型（functional）、逻辑型（logical）、以及面向对象性范型（object-oriented）。命令型范型是系统包含程序运行状态（state）和命令（statement），通过命令语句来改变系统的状态。命令型范型是一个系统指令集，通过指令来改变系统的状态。功能型范型是将计算视为一个求解方程组的过程，相对于命令型的通过命令执行改变系统状态而言，功能型强调方程式的求解。以上两种计算范型都建立在系统的输入输出关系之上的。逻辑型范型是为输入输出建立起一般的关系描述，它描述了问题解决方案的一组特征集，而不是求解的步骤。面向对象范型是建立在命令型范例之上，它可以看作是一个扩展型的命令范例，将变量和操作封装在类中。基于面向对象范型的启发，Shoham 提出了一种面向 agent 的系统设计模型 （agent-oriented），其中 agent 的基本单元是由一组智能参数构成的，如信念、责任和决定。

与以往的范型不同，自治计算范型注重利用自治计算体来建模和演化系统，从而解决复杂计算问题和刻画复杂系统行为。在自治计算中基本单元是自治计算体，核心概念是计算体的自治行为，也就是计算体自发的基于自身局部信息来决定自己的行为，而不存在全局性的调控机制。

面向对象的范型 OOP（Object-Oriented Programming）、面向 agent 的范型 AOP（Agent-Oriented Programming）和面向自治的范型 AOC（Autonomy-Oriented Computing）的对比分析见表 1。

表 1. 三种计算范型的比较[25]

	面向对象范型(OOP)	**基于 Agent 范型(AOP)**	**面向自治范型(AOC)**
基本组成单元	对象	代理	计算体、环境
基本单元特征	成员函数、成员变量	信念、决策、能力、责任	状态、评估函数、目标、原子行为、行为规则
交互	继承，对象之间消息传递	消息传递，包括通知、请求、报价、允诺和拒绝	计算体与计算体、计算体与环境之间的交互, 计算体之间的直接交互与间接交互
计算	消息传递与消息响应	消息传递与消息响应	面向自治的自组织性
适用范围	系统建模与代码复用	分布式系统构建和分布式问题求解	复杂系统行为刻画和复杂系统计算问题求解
功能实现形式	成员函数设计	智能状态转移	计算体的行为

对于以上三种计算范型，OOP 和 AOP 计算需要先验知识作为计算依据，如知识表达、推理机制等人工智能技术，通过预先设定对象或 agent 的消息传递与消息响应行为来完成计算功能。而在 AOC 计算中，计算不需要预先的设定机制，而是通过计算体的自治

性自组织的完成计算任务。对于 OOP 和 AOP 而言，由于计算是建立在先验知识的基础上，所以在面对动态、开放、时变系统的计算问题时，不能做到有效的自适应、自调整。而在 AOC 计算中，自治计算体可以根据系统目标的变化，自适应调整自己的状态和行为，从而可以实现计算的动态自适应。OOP 计算是同步过程，系统中有中央控制机制来协调对象之间的消息传递和消息响应。而在 AOC 中，计算体的自治性决定了面向自治计算是异步过程。从计算机理而言，OOP 和 AOP 都是基于消息机制，计算的过程即是对象或者 agent 传递、处理、响应消息的过程。而在 AOC 计算中，计算体通过与其他计算体以及环境之间直接或者间接的交互来实现系统的自组织。在 AOC 系统中，自治计算体的局部行为会根据系统的负反馈作用进行优化调整，于是非线性的积聚效应在负反馈的调整下使系统产生某种期望的行为或状态。

总而言之，AOC 是一种类似于自然系统问题求解机制的自底向上方法，这也是自治计算能够适用于解决计算问题以及刻画复杂系统行为的优势所在。

4.2 自治计算与相关计算方法的比较

自治计算的立足点是计算体的自治行为，核心是自组织原理，目标是刻画复杂系统自治行为特征和解决复杂系统计算问题。通过与自治计算相关的研究领域和计算算法的对比，可以更进一步的了解和认识自治计算的内涵和外延。

a. 自主计算（Autonomic Computing，AC）

自主计算[56][57]是 IBM 公司于 2001 年提出的基于计算机系统的自治管理方法，其核心思想是信息系统能够自主地对自身进行管理，维持其可靠性。自主计算（AC）与自治计算（AOC）的相同点都在于强调系统中基本单元的自治性，即计算组件（computing components）或计算体（entities）都是通过局部的交互行为（与其他同类成员之间交互，或环境之间交互）来独立的决定自己的动作行为，而非受到中央控制机制的命令或指示。AC 计算与 AOC 计算的不同点以下包括 3 个方面：（1）从研究领域来看，AC 计算是针对计算机管理系统的复杂性而设计的自调节、自管理机制，而 AOC 计算是更加普遍意义上的复杂系统自治行为刻画和复杂计算问题的解决方法。（2）AC 计算的立足点是计算组件（computing components）的自管理、自调节，而 AOC 计算除了计算体的自治性（entity autonomy）以外还包含计算体之间结构的自组织性（structural self-organization）、行为的自组织性（behavioral self-organization）和系统的涌现行为（emergence）。（3）AC 计算是一个解决方案，是为了应对计算机管理系统中复杂度不断的增加。AOC 计算是设计复杂问题解决方案的方法论，AOC 通过对系统特征的抽象，构建一套完整的系统分析，建模以及解决机制。

b. 多 agent 系统（Multi-Agent Systems，MAS）

MAS 系统和 AOC 系统都是自底向上的系统分析方法，它们之间的区别在于：（1）MAS 中的 agent 基于 BDI 模型，即 agent 的思维状态包括信念（belief）、愿望（desire）和意图（intention）这三个属性。因此，MAS 中的 agent 对象在系统中的指代意义通常是是预先定义且固定不变的，只是根据自己的三个属性理性的改变自身状态和行为。而 AOC 中计算体主要通过个体行为、行为规则和局部目标进行描述，其指代对象是相对概念，随着计算过程的演化会不断的调整其指代对象在系统中的层次。（2）MAS 中的构成单元是智能性的 agent，如 agent 可能具有逻辑推断机制等。而在

AOC 计算中，计算体不一定具有智能性。即计算体可能仅拥有原子行为，不具有认知能力，如简单的“刺激-反应”计算体。（3）MAS 关注如何通过多 agent 之间的协作来实现问题求解。而 AOC 关注自治计算体如何通过自组织行为演化、产生系统的组织结构和宏观现象。其系统目标不单是问题求解，还包括系统现象、行为、模式的刻画。（4）MAS 不针对某种特定的计算系统，既适用于简单问题的求解，也可以用于复杂问题的建模。而 AOC 依存于一种新型的计算系统（如一个开放的非平衡系统），关注于复杂系统特征的复现与机制分析，以及解决复杂系统中的计算问题。

c. Agent 系统仿真（Agent-Based Simulation）

自治计算的任务之一是分析、刻画复杂系统，从这个角度来说，自治计算是基于计算体的系统仿真。AOC 通过计算体的原子行为，以及计算体与计算体、计算体与环境之间的交互，实现系统的自组织，从而模拟、复现、刻画复杂系统的行为特征和结构组织特性。自治计算的另一任务是利用系统的涌现为解决现实中存在的大规模、分布式、开放、动态系统问题提供解决方案。从这个角度看，自治计算系统是设计问题解决方案的方法论，而非简单的系统复杂行为模拟。

d. 统计力学（Statistical Mechanics）

统计力学从大量粒子的宏观运动行为来描述复杂系统特性。自治计算与统计力学的区别在于，统计力学描述的是宏观行为的统计特性，而 AOC 着眼于宏观行为背后的机制，如计算体的自治行为的机理，计算体之间的自组织机制，以及系统涌现产生的条件与机制。

e. 群体智能（Swarm Intelligence）

群体智能（swarm intelligence）是一类自然启发式计算（nature-inspired computing）。群体智能从蚂蚁觅食、建巢，鸟群捕食等自然过程得到启发，设计具有相似原理的优化算法，例如蚁群优化算法（ant colony optimization）、粒子群优化算法（particle swarm optimization）等。与自治计算类似，群体智能考察系统的分布式特点，考察系统基本单元局部行为，以及涌现的宏观模式和全局解。群体智能与自治计算的区别在于：（1）从模型假设和一般性来看，群体智能中计算体的行为规则是算法构建的已知条件。因此，群体智能可以看作自治计算中机制复现方法的具体实例。自治计算更关注开发自然启发式计算的一般方法（比如上文提到的自治计算的三个方法），而不是仅仅关注于某一类具体的算法。（2）从算法评价来看，群体智能中的算法（如蚁群优化算法）通常关注一般的算法效能（如有效性、效率性等），而自治计算算法在关注这些基本评价指标的同时，还关注算法的可拓展性、鲁棒性、自适应性等特征效能评价指标，以及这些特征效能产生的根本原因。

f. 复杂问题求解方法

自治计算在处理复杂计算问题（如 NP-hard 问题中的可满足性问题（satisfiability problem））时明确使用复杂系统的结构和思想，这与传统的算法类型（如系统性方法（systematic approach）、局部搜索方法（local search approach）、随机算法（random algorithm））相比具有显著不同。AOC 与系统化搜索的区别在于：AOC 从一个完整的解出发进行问题求解，在这个过程中，自治计算采用构造包含多自主性计算体的计算复杂系统来表示和解决约束满足性问题。AOC 与局部化搜索的区别在于：（1）适应度的思想并不应用在整个候选解上，而是应用于单个计算体，在此基础上我们引入了全局适应度函数和局部适应度函数的概念。（2）从实现角度来看，自治计算方法采取异

步并发（asynchronous concurrency）而不是局部搜索的顺序性（sequential）方式。（3）自治计算通过设计自主性计算体的局部自主性来实现自组织计算，明确体现和强调了自组织原理的使用。

5 结束语

当今世界，面对普遍存在的大规模、分布式、异构、以及动态开放系统，人们亟需理解其背后的运作机理，解决由此衍生出的各种复杂计算问题。作为一种全新的自然启发式计算范式，自治计算具有应对这两项挑战的独特优势，已在约束满足、组合优化、图像处理、数据挖掘等经典计算领域得到广泛的应用和验证。自治计算的优势具体体现在通过计算体自治性的设计和利用自组织计算的原理，使得自治系统快速收敛到系统的期望目标，或复杂问题的目标解。基于自治系统解决方案不仅具有收敛性、效率性和有效性等一般系统和算法的特征，还具有可拓展性、鲁棒性，自适应性等特征效能。因此，自治计算能够立足于计算的角度，研究自然界、人类社会中广泛存在的问题，如疾病控制、能源政策调整、社会舆情分析等。相信这一复杂计算的新领域将在自身迅速发展的同时为人们认识和改造世界做出更大的贡献。

致谢

我们衷心的感谢多年来 AOC 项目的研究人员（按姓氏拼音排序）：Yichuan Cao, William Cheung, Samuel P. M. Choi, Chao Gao, Jing Han, Bingcheng Hu，Xiaolong Jin, Markus Kaiser, Jun Le, Yunqi Lei, Hong Qin, Hongjun Qiu, Y. Y. Tang, Yi Tang, K. C. Tsui, Tingting Wang, Yuan-shi Wang, Jianbin Wu, Xiaofeng Xie, Bo Yang, Chunyan Yao, Yiyu Yao, Yiming Ye, Jian Yin, Shiwu Zhang, Yi Zhao, Ning Zhong, Huijian Zhou 等。同时诚挚的感谢香港浸会大学计算机科学系为我们提供的开放而舒适的研究环境。特别感谢陶丽和史本云同学在本文修订过程中的修改意见和写作建议。

参考文献

[1] J. Liu. Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation. World Scientific Publishing, River Edge, NJ, USA, 2001.

[2] J. Han, J. Liu, and Q. Cai. From ALife Agents to a Kingdom of N Queens. Intelligent Agent Technology: Systems, Methodologies, and Tools, World Scientific Publishing, pp.110–120, 1999.

[3] J. Liu, J. Han. A Multi-agent Computing Paradigm for Constraint Satisfaction Problems. International Journal of Pattern Recognition and Artificial Intelligence, 15(3), pp.475–91, 2001.

[4] J. Liu, J. Han, and Y. Y. Tang. Multi-agent Oriented Constraint Satisfaction. Artificial Intelligence, 136(1), pp.101–144, 2002.

[5] X. Jin, J. Liu. Agent Networks: Topological and Clustering Characterization. Intelligent Technologies for Information Analysis, Springer, pp.285–304, 2004.

[6] X. Jin, J. Liu. Agent Network Topology and Complexity. In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03), ACM, pp.1020–1021, 2003.

[7] J. Liu, J. Yin. Multi-Agent Integer Programming. In Lecture Notes in Computer Science, Springer, 1983, pp.301–307, 2000.

[8] K. C. Tsui, J. Liu. Evolutionary Diffusion Optimization, Part I: Description of the Algorithm. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Honolulu, Hawaii, May 12–17, 2002.

[9] K. C. Tsui, J. Liu. Evolutionary Diffusion Optimization, Part II: Performance Assessment. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), Honolulu, Hawaii, May 12–17, 2002.

[10] J. Liu, Y. Y. Tang, and Y. Cao. An Evolutionary Autonomous Agents Approach to Image Feature Extraction. IEEE Transactions on Evolutionary Computation, 1(2), pp.141–158, 1997.

[11] J. Liu, H. Zhou, and Y. Y. Tang. Evolutionary Cellular Automata for Emergent Image Features. In Progress in Neural Information Processing, Springer, pp.458–463, 1996.

[12] J. Liu, Y. Y. Tang. Adaptive Image Segmentation with Distributed Behavior Based Agents. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(6), pp.544–55l, 1999.

[13] J. Liu, Y. Zhao. On Adaptive Agentlets for Distributed Divide-and-Conquer: A Dynamical Systems Approach. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 32(2), pp.214–227, 2002.

[14] J. Liu. Autonomy-Oriented Computing (AOC): A New Paradigm in Data Mining and Modeling (Invited Talk), Workshop on Data Mining and Modeling, Hong Kong, June 27–28, 2002.

[15] B. Yang, J. Liu, and D. Liu. An Autonomy-Oriented Computing Approach to Community Mining in Distributed and Dynamic Networks. Autonomous Agents and Multi-Agent Systems, 20(2), pp.123–157, 2010.

[16] J. Liu, S. Zhang, and J. Yang. Characterizing Web Usage Regularities with Information Foraging Agents. IEEE Transactions on Knowledge and Data Engineering, 16(5), pp.566-584, 2004.

[17] S. Zhang, J. Liu. A Massively Multi-agent System for Discovering HIV-Immune Interaction Dynamics. Lecture Notes in Computer Science, Springer, Massively Multi-agent Systems, pp.161-173, 2005.

[18] J. Liu, J. Wu. Multi-Agent Robotic Systems. CRC Press, 2001.

[19] J. Liu. Self-organization, Evolution, and Learning, Invited Lectures by Leading Researchers, In Pacific Rim International Workshop on Multi-Agents (PRIMA'02) Summer School on Agents and Multi-Agent Systems, Tokyo, August 17, 2002.
[20] J. Liu, H. Qin, Y. Y. Tang, and Y. Wu. Adaptation and Learning in Animated Creatures. In Proceedings of the 1st International Conference on Autonomous Agents (AGENTS'97), pp.5-8, 1997.
[21] J. Liu, H. Qin. Behavioral Self-organization in Lifelike Synthetic Agents. Autonomous Agents and Multi-Agent Systems, 5(4), pp.397–428, 2002.
[22] R. A. Brooks. The Next 50 Years. Communications of the ACM, 51(1), pp.63–64, 2008.
[23] G. Agha. Computing in Pervasive Cyberspace. Communications of the ACM, 51(1), pp.68–70, 2008.
[24] J. Liu. Autonomy-Oriented Computing (AOC): The Nature and Implications of A Paradigm for Self-organized Computing (Keynote Talk), In Proceedings of the 4th International Conference on Natural Computation (ICNC'08) and the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'08), pp.3–11, 2008.
[25] J. Liu, X. Jin, and K. C. Tsui. Autonomy-Oriented Computing: From Problem Solving to Complex Systems Modeling (Multiagent Systems, Artificial Societies, and Simulated Organizations). Springer, Secaucus, NJ, USA, 2004.
[26] J. Liu, X. Jin, and K. C. Tsui. Autonomy-Oriented Computing (AOC): Formulating Computational Systems with Autonomous Components. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 35(6), pp.879–902, 2005.
[27] Y.Y. Yao. Granular Computing: Past, Present and Future, In the 2009 IEEE International Conference on Granular Computing (GrC'08), pp. 80-85, 2008.
[28] Y.Y. Yao. The Rise of Granular Computing, Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 20(3), pp.299-308, 2008.
[29] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperative Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1), pp. 1–13, 1996.
[30] 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. Behavioural Mechanisms of Food Selection, NATO-ASI Series, Springer, 20, pp.661-678, 1990.
[31] 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), pp. 349–355, 1998.
[32] R. Dawkins. The Selfish Gene. Oxford University Press, 1989.
[33] R. G. Reynolds. An Introduction to Cultural Algorithms. In Proceedings of the Third Annual Conference on Evolutionary Programming. World Scientific, pp.131–139, 1994.
[34] J. Doran, N. Gilbert. Simulating Societies: An Introduction, Simulating Societies: The Computer Simulation of Social Phenomena. UCL Press, pp.1–18, 1994.
[35] D. Helbing, B. A. Huberman. Coherent Moving States in Highway Traffic, Nature, 396, pp.738–740, 1998.
[36] G. K. Still. Crowd Dynamics. Ph.D. dissertation, Mathematics Department, Warwick University, 2000.
[37] D. Helbing, I. Farkas, and T. Vicsek. Simulating Dynamic Features of Escape Panic. Nature, 407, pp.487–490, 2000.
[38] A. J. Angeline. Adaptive and Self-adaptive Evolutionary Computations. Computation Intelligence: A Dynamic Systems Perspective. IEEE Press, pp.152–163, 1995.
[39] T. Back. Self-adaptation. Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press, pp.C7.1:1–15, 1997.
[40] T. Pavlidis. Algorithms for Graphics and Image Processing. Computer Science Press, 1992.
[41] I. Pitas. Digital Image Processing Algorithms. Prentice Hall, 1993.

[42] B. Hu, J. Liu, and X. Jin. Multi-agent RoboNBA Simulation: From Local Behaviours to Global Characteristics. Special Issue: Agent-Directed Simulation of the Simulation: Transactions of the Society for Modelling and Simulation International, 81(7), 2005.
[43] J. Liu, B. Hu. On Emergent Complex Behavior, Self-organized Criticality and Phase Transitions in Multi-agent Systems: Autonomy Oriented Computing (AOC) Perspectives. International Journal of Modelling, Identification and Control, 3(1), pp.3–16, 2008.
[44] J. Liu, H. Qin. Organizing Synthetic Agent Behaviors Based on a Motif Architecture. In Proceedings of the Third International Conference on Autonomous Agents (AGENTS'99), pp.340–341, 1999.
[45] J. Liu, X. Jin, and J. Han. Distributed Problem Solving without Communication - An Examination of Computationally Hard Satisfiability Problems. International Journal of Pattern Recognition and Artificial Intelligence, pp.1041-1064, 2002.
[46] T. D. Wolf, T. Holvoet. Emergence Versus Self-organisation: Different Concepts but Promising when Combined. Engineering Self-Organising Systems, Springer, pp.1-15, 2005.
[47] P. Bak. How Nature Works. Copernicus, 1996.
[48] C. G. Langton. Computation at the Edge of Chaos: Phase Transitions and Emergent Computation. Physica D: Nonlinear Phenomena, 42(1-3), pp. 12–37, 1990.
[49] U. Wilensky. NetLogo Ants Model. 1997. http://ccl.northwestern.edu/netlogo/models/Ants.
[50] C.W. Reynolds. Flocks, Herd, and Schools: A Distributed Behavioral Model, Computer Graphics, 21(4), pp.15–24, 1987.
[51] A. Nathan, V. C. Barbosa. V-like Formations in Flocks of Artificial Birds. Artificial Life, 14(2), pp.179-188, 2008.
[52] R. Reddy. To Dream the Possible Dream (Turing Award Lecture). Communications of the ACM, 39(5), pp.105–112, 1996.
[53] Z. Dezso, A.-L. Barabasi. Halting Viruses in Scale-free Networks. Physical Review E, 65(5), 2002.
[54] P. Echenique, J. Gomez-Gardenes, Y. Moreno, and A. Vazquez. Distanced Covering Problem in Scale-free Networks with Degree Correlation. Physical Review E, 71(3), 2005.
[55] E. Bonabeau. Editor's Introduction: Stigmergy, Artificial Life, 5(2), pp.95-96, 1999.
[56] J. Kephart, D. Chess. The Vision of Autonomic Computing. IEEE Computer, 36(1), pp. 41–50, 2003.
[57] P. Horn. Autonomic Computing: IBM's Perspective on the State of Information Technology. IBM Corp, 2001.
[58] H. Qiu, J. Liu, and N. Zhong. A Dynamic Trust Network for Autonomy-Oriented Partner Finding. In Internal Conference on Active Media Technology (AMT'09), pp.323–334, 2009.
[59] J. Liu, C. Gao, and N. Zhong. A Distributed Immunization Strategy Based on Autonomy-Oriented Computing. In 18th International Symposium on Methodologies for Intelligent Systems (ISMIS'09), pp.503–512, September 14-17, 2009.
[60] J. Liu, Y.W. Chen, and Y.Z. Lu. Self-Organized Combinatorial Optimization. Research Report of Department of Computer Science, Hong Kong Baptist University, May 8, 2008.