Presentation Slides for the AOC Book

Liu, Jin, & Tsui, *Autonomy Oriented Computing (AOC):* From Problem Solving to Complex Systems Modeling, Springer, 2005

Jiming Liu

Hong Kong Baptist University

May 10, 2005

Outline I

Outline II

Outline III

Outline IV

Potential Applications: A New Era of Discovery and Opportunity

- **Business:** To predict market share of a new product through large-scale simulations of consumer behavior;
- **Life and material sciences:** To develop amorphous computational particles (e.g., bio-robot agents to kill cancer cells or smart paints to fill cracks);
- **Environmental sciences:** To deploy wireless, mobile sensor networks to monitor wild vegetation (e.g., tracking);
- **Robotics:** To coordinate exploratory robots for collective tasks.

Common Characteristic: A New Computing Paradigm

- The task of computing is seamlessly carried out in a variety of physical embodiments;
- There is no single multi-purpose or dedicated machine;
- The key to success in such applications lies in a large-scale deployment of computational agents – capable of autonomously making their localized decisions and achieving their collective goals.

- A methodology for solving computationally hard problems (e.g., conventional constraint or optimization problems) or computing problems that involve large-scale, distributed, locally-interacting entities.
  - To develop models of computational autonomy;
  - To solve 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.
- A generic framework for handling modeling and characterization of complex systems, such as ecological, social, economical, mathematical, physical;
  - To characterize complex phenomena or emergent behavior in natural and artificial systems that involve a large number of self-organizing, interacting entities;
  - To discover laws and mechanisms underlying complex phenomena or emergent behavior.

- Our first systematic study on AOC originated in 1996.
  - J. Liu, Y. Y. Tang, and Y. Cao. An Evolutionary Autonomous Agents Approach to Image Feature Extraction. *IEEE Trans. on Evolutionary Computation*, 1(2):141-158, 1997.
  - J. Liu, H. Zhou, and Y. Y. Tang. Evolutionary Cellular Automata for Emergent Image Features. In Shun-ichi Amari et al., editors, *Progress in Neural Information Processing*, Springer, pages 458-463, 1996.
- The notion of AOC first appeared in 2001.
  - J. Liu. *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation*, World Scientific Publishing, 2001.
- The First International Workshop on AOC was organized and held in Montreal in 2001.

- J. Liu, K. C. Tsui, and J. Wu. Introducing Autonomy Oriented Computation (AOC). In *Proceedings of the First International Workshop on Autonomy Oriented Computation (AOC 2001)*, Montreal, May 29, 2001, pages 1-11.

- Earlier projects at the AOC Lab included:
  - **Constraint satisfaction problem solving**
    - Ideas of cellular automaton-like computational entities in solving constraint satisfaction problems (CSP):
    - J. Liu, J. Han, and Y. Y. Tang. Multi-Agent Oriented Constraint Satisfaction *Artificial Intelligence*, 136(1):101-144, 2002.
    - J. Han, J. Liu, and Q. Cai. From ALife Agents to a Kingdom of N Queens. In J. Liu and N. Zhong, editors, *Intelligent Agent Technology: Systems, Methodologies, and Tools*, World Scientific Publishing, pages 110-120, 1999.
    - Formal notions of computational complexity for AOC in distributed problem solving:

- X. Jin and J. Liu. Agent Networks: Topological and Clustering Characterization. In N. Zhong and J. Liu, editors, *Intelligent Technologies for Information Analysis*, Springer, pages 285-304, 2004.

- **Mathematical programming**
  - J. Liu and J. Yin. Multi-Agent Integer Programming. In *Lecture Notes in Computer Science*, Vol. 1983, Springer, pages 301-307, 2000.

- **Optimization**
  - Solving benchmark functional optimization problems:
  - K. C. Tsui and J. Liu. Evolutionary Multi-Agent Diffusion Approach to Optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific Publishing, 16(6):715-733, 2002.
  - K. C. Tsui and J. Liu. Evolutionary Diffusion Optimization, Part I: Description of the Algorithm. Also Part II: Performance Assessment. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, May 12-17, 2002.

- **Image processing**

- J. Liu and Y. Zhao. On Adaptive Agentlets for Distributed Divide-and-Conquer: A Dynamical Systems Approach. *IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 32(2):214-227, 2002.
- J. Liu and Y. Y. Tang. Adaptive Segmentation with Distributed Behavior Based Agents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):544-551, 1999.

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

- These AOC projects differ from traditional AI and agent studies in that here we pay special attention to the role of **self-organization**
  - It is a powerful methodology as demonstrated in nature and well suited to the problems that involve large-scale, distributed, locally interacting, and sometimes rational entities;
  - Also demonstrated in the earlier work on
    - Collective problem solving with a group of autonomous robots:
    - J. Liu and J. Wu. *Multi-Agent Robotic Systems*, CRC Press, 2001.

- J. Liu and J. Wu. Evolutionary Group Robots for Collective World Modeling. In *Proceedings of the Third International Conference on Autonomous Agents (AGENTS'99)*, Seattle, WA, May 1-5, 1999.
- Behavioral self-organization:
- J. Liu, H. Qin, Y. Y. Tang, and Y. Wu. Adaptation and Learning in Animated Creatures. In *Proceedings of the First International Conference on Autonomous Agents (AGENTS'97)*, Marina del Rey, California, Feb. 5-8, 1997.
- J. Liu and H. Qin. Behavioral Self-Organization in Synthetic Agents. *Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, 5(4):397-428, 2002.

- AOC approaches to characterizing observed or desired regularities in real-world complex systems, as a white-box alternative to the traditional top-down or statistical modeling:
  - Self-organized Web regularities:
  - J. Liu, S. Zhang, and J. Yang. Characterizing Web Usage Regularities with Information Foraging Agents. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):566-584, 2004.
  - J. Liu and S. Zhang. Unveiling the Origins of Internet Use Patterns. In *Proceedings of INET 2001, The Internet Global Summit*, Stockholmsmssan, Stockholm, Sweden, June 5-8, 2001.
  - HIV infection dynamics:
  - S. Zhang and J. Liu. A Massively Multi-Agent System for Discovering HIV-Immune Interaction Dynamics. In Proceedings of the First International Workshop on Massively Multi-Agent Systems (MMAS'04), Kyoto, Japan, Dec. 10-11, 2004.
- AOC applications to the Internet
  - The Wisdom Web:

- J. Liu. Web Intelligence (WI): What Makes Wisdom Web? In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, Aug. 9-15, 2003, pages 1596-1601, Morgan Kaufmann Publishers.
- J. Liu. *Web Intelligence (WI): Some Research Challenges*, Invited Talk, the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Aug. 9-15, 2003, Acapulco, Mexico.
- Web Intelligence (WI) capabilities (e.g., autonomous service planning; distributed resource discovery and optimization):
- Y. Wang and J. Liu. Macroscopic Model of Agent Based Load Balancing on Grids. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, Melbourne, Australia, July 14-18, 2003.
- K. C. Tsui, J. Liu, and M. J. Kaiser. Self-Organized Load Balancing in Proxy Servers. *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, 20(1):31-50, 2003.

AOC is related to a broad range of disciplines, such as:

- Artificial life (ALife) – simulation of life in a computer setting. It falls short of its use as a computational approach to problem solving. AOC does not necessarily need to exactly reproduce lifelike behavior, as natural phenomena are usually abstracted and simplified.
- Agent-based simulation (ABS) – finding explanations to observed phenomena. There is no computational problem to be solved in ABS.
- Self-organized criticality (SOC) [Jensen, 1998] – combining concepts of self-organization and critical behavior to explain complex phenomena (e.g., avalanches, sand pile, rice pile, droplet formation, earthquakes, and evolution.)
- Studies on multi-agent systems for distributed decision making [Durfee, 1999, Sandholm, 1999] – handling computational tasks by delegating responsibilities to groups of agents.

- Ecology of computation
  [Hogg and Huberman, 1993, Huberman, 1988] – individual problem solvers tackling a problem with different methods.
- Distributed constraint satisfaction problem (distributed CSP) [Yokoo et al., 2001, Yokoo et al., 1998, Yokoo and Hirayama, 2000] – to solve CSPs in a distributed manner by employing direct communications to coordinate the assignments to their respective variables.
  - Generally speaking, direct communication in a large-scale multi-agent system is time-consuming. Hence, in an AOC system, autonomous entities utilize indirect communication through their environment.
  - Furthermore, AOC is inspired by complex systems where numerous entities self-organize themselves through nonlinear interactions and aggregations, and gradually emerge certain complex behaviors.

- Swarm intelligence
  [Bonabeau et al., 1999, Bonabeau et al., 2000] – a social insect metaphor in problem solving. Unlike AOC, this study does not address the issues of discovering problem solvers or explaining complex systems behavior.

Other Related Disciplines I

- Agent-Based Problem Solving;
- Amorphous Computing;
- Artificial Intelligence;
- Autonomous Agents and Multi-Agent Systems;
- Complex Adaptive Systems;
- Computational Biology;
- Computational Finance and Economics;
- Data Fusion and Exploration;
- Emergent Computation;
- Image Processing and Computer Vision;
- Intelligent Systems;
- Modeling and Simulation;

Other Related Disciplines II

- Nature Inspired Computation;
- Operations Research;
- Optimization;
- Programming Paradigms;
- Robotics and Automation;
- Self-Organization.

Two Emphases in this Presentation

1. A general overview of the AOC modeling methodology;
2. Some representative case studies on how to implement the general approaches to AOC.

Complex Multi-Entity Systems

- Some researchers want to understand the working mechanism of a complex system concerned.
  - Immunologists, for example, want to know the way in which the human immune system reacts to antigens [Louzoun et al., 2000].
  - Economists want to know the factors contributing to the ups and downs in share prices.
- Others studying complex systems behavior want to simulate the observed complex behavior and formulate problem solving strategies for hard computational problems, such as global optimization.
- Computer scientists and mathematicians have formulated various algorithms based on natural evolution to solve their problems at hand. In general, one wants to be able to explain, predict, reconstruct, and deploy a complex system.

How to Build a Complex Systems Model? I

- **Top-down** approaches start from the high-level characterization of a system and use various tools, such as ordinary differential equations.
  These approaches generally treat every part of a complex system homogeneously and tend to model average cases well, where the behavioral difference of the individuals is minimal and can be ignored [Casti, 1997].
- **Bottom-up** approaches are characterized as:
  - **Autonomous:** System entities are rational individuals that act independently. In other words, a central controller for directing and coordinating individual entities is absent.
  - **Emergent:** They exhibit complex behavior that is not present or predefined in the behavior of the autonomous entities.

How to Build a Complex Systems Model? II

- **Adaptive:** They often change their behavior in response to changes in the environment in which they are situated.
- **Self-organized:** They are able to organize themselves to achieve the above.

Complex Systems Modeling Using a Bottom-up Approach

- It centers around the external behavior and internal behavior of individual entities.
- Trickiest part: How to define the relationship between these two types of behavior?
- AOC adds a new dimension to the modeling process, i.e., modeling and deploying autonomy.

- **Autonomy** – an attribute of entities in a complex system;
- **Autonomous entity** – the building block of an AOC system.

Types of Behavior

Entities in a complex system can perform certain primitive behavior as well as three types of complex behavior: emergent behavior, purposeful behavior, and emergent purposeful behavior.

- **Primitive behavior** – the behavior governed by a set of predefined rules.
  These rules dictate how the states of the entity are updated.
  They are triggered by some internal or external stimuli.
- **Emergent behavior** – the behavior not inherent in the primitive behavior of an entity.
  It is achieved through nonlinear interactions among individual entities.

Emergent behavior may not be the same as collective behavior as it may not involve sharing of power or division of labor among individual entities.

- **Purposeful behavior** – the behavior that leads to certain desired states (i.e., goals) of entities;
- **Emergent purposeful behavior** – the emergent behavior that directs entities towards certain goals.
  - If the entities of a complex system are able to adapt, the primitive behavior of entities is bound to be different over time. As a result, different types of complex behavior may be emerged.
  - Emergent behavior may not arise only through interactions among individual entities. It can also arise through interactions among groups of entities.

Ant Colony as an Example I

- Food foraging is an individual task as well as a group task [Goss et al., 1990].
- Purposeful behavior – *wandering around* of ants.
- Emergent behavior – their convergence on a certain food source.
  - Ants start off with some kind of random walk in the absence of any information about a food source.
  - While wandering, ants lay some quantities of pheromone along their paths.
- Emergent purposeful behavior – Once a food source is found, more ants will gradually follow the path between the food source and the nest, and consequently more pheromone will be laid along this path.
  - More pheromone will in turn recruit more ants. This process acts as a positive feedback loop, until the food source is exhausted and the pheromone evaporates.

Autonomy Defined

- **Entity autonomy:** Condition/quality of being self-governed, self-determined, and self-directed – to make decisions for themselves, subject to information availability and self-imposed constraints.
  Example: The primitive behavior of an entity is free from the explicit control of other entities (except indirect influence).
- **Synthetic autonomy:** An abstracted equivalent of the autonomy of an entity in a natural complex system.
  AOC aims at building computational systems where entities are equipped with synthetic autonomy – An AOC system exhibits emergent (purposeful) behavior.
- **Emergent autonomy:** An observable, self-induced condition or quality of an AOC system is composed of entities with synthetic autonomy.

Levels of Abstraction

If a human society is to be modeled as a computational system, abstraction can possibly occur at several levels: population, individual, biological system, cell, molecule, and atom.

- Entity autonomy, synthetic autonomy, and emergent autonomy are present at all these levels.
- The autonomy obtained at a lower level, say, the cell level, is the foundation of the autonomy at a higher level, say, the biological system level.
- This multi-level view of autonomy encompasses Brooks' subsumption architecture [Brooks, 1991].

Computational System Autonomy

Autonomy in a computational system, built from computational entities with synthetic autonomy, refers to conditions/qualities of having self-governed, self-determined, and self-directed computational entities that exhibit emergent autonomy.

Three AOC Approaches

AOC systems are classified based on:

- How much human involvement is necessary?
- How sophisticated a model of computational autonomy is?

- AOC-by-fabrication
- AOC-by-prototyping
- AOC-by-self-discovery

AOC-by-Fabrication

- It aims at replicating and utilizing certain *self-organized collective behavior* from the real world to form a *general purpose problem solver*.
- The working mechanism is more or less known and may be simplified during the modeling process.
- Examples: Nature inspired techniques, such as the genetic algorithm (GA), entity-based image feature extraction, artificial creature animation, and ant colony optimization.
- Research in artificial life is related to this AOC approach up to the behavior replication stage.

AOC-by-Prototyping

- It attempts to understand the working mechanism underlying a complex system by modeling and simulating autonomous entities.
- A manual trial-and-error process is employed to achieve an artificial system as vivid as possible.
- Examples: Internet ecology, traffic jams, Web regularities based on self-adaptive information foraging entities.

AOC-by-Self-Discovery

- It automatically fine-tunes the parameters of autonomous behaviors in solving and modeling certain problems.
- The difference measure between the desired emergent behavior and the current emergent behavior of the system in question becomes part of the feedback that affects the primitive behavior of an entity.
- Examples: Autonomous entities to adaptively solve a large-scale, distributed optimization problem in real time (e.g., evolutionary algorithms that exhibit self-adaptive capabilities).

Key Features of AOC

- AOC focuses on modeling and developing systems with autonomous entities, in an attempt to solve hard computational problems and to characterize complex systems behavior.
- The basic element is an *autonomous* entity.
- Entities locally determine their behavior by themselves, and no global control mechanism exists.

A Comparison of OOP, AOP, and AOC

A comparison among object oriented programming (OOP), agent oriented programming (AOP) [Shoham, 1993], and autonomy oriented computing (AOC).

	Object Oriented Programming (OOP)	*Agent Oriented Programming (AOP)*	*Autonomy Oriented Computing (AOC)*
Basic element	object	agent	autonomous entity and environment
Characterization of a basic element	member variables and member functions	beliefs, decisions, capabilities, and obligations	states, evaluation function, goals, primitive behavior, and behavioral rules
Interaction	inheritance and message passing among objects	messages among agents, including inform, request, offer, promise, decline, etc.	(1) interaction between entities and their environment and (2) direct or indirect interaction among entities
Computation	message passing and response methods	message passing and response methods	(1) aggregation of behavior and interaction and (2) self-organization in autonomous entities
Suitability	(1) systems modeling and (2) computation based on reusable codes	(1) developing distributed systems and (2) solving distributed problems [Kuhnel, 1997]	(1) solving hard computational problems and (2) characterizing complex systems behavior
Implementation of functionality	member functions	mental state transitions	primitive behaviors

Autonomy Oriented Modeling I

- An environment contains a homogeneous region with the same physical feature (i.e., a goal region).
- The feature of the goal can be evaluated based on some measurements (e.g., grey level intensity of an image).
- The task of autonomous entities is to search the feature locations of the goal region.
- Entities can recognize and distinguish feature locations, if encountered, and then decide and execute their reactive behavior.

Image Processing:

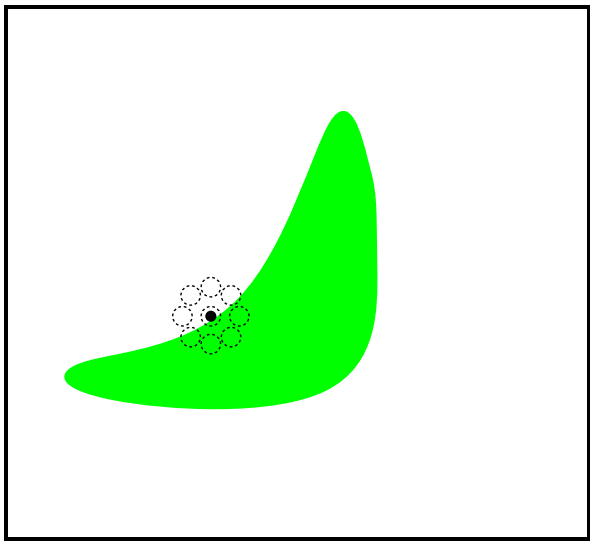
- Autonomous entities are deployed to the two-dimensional representation of an image.

Autonomy Oriented Modeling II

- Each entity is equipped with an ability to assess the homogeneity of a region within a predefined locality.
- Homogeneity is defined by the relative contrast, regional mean, and region standard deviation of the grey level intensity.
- When an autonomous entity locates a homogeneous region within the range of the pixel at which it presently resides, it breeds a certain number of offspring entities and delivers them to its local region in different directions.
- On the other hand, when a heterogeneous region is found, an entity will diffuse to another pixel in a certain direction within its local region.

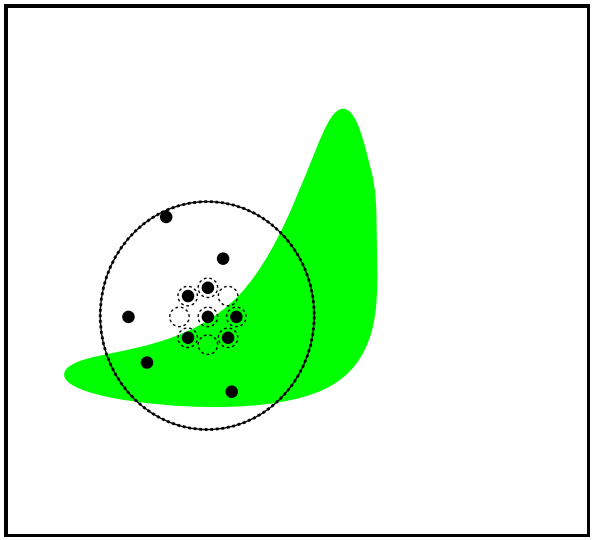
An Illustration of Autonomous Entities

As an entity, which is marked as a solid circle, moves to a new location, it senses its neighboring locations, marked by dotted circles in this example. Specifically, it counts the number of locations at which the grey level intensity is close to that of the entity's current location.



An Illustration of Autonomous Entities

When the count reaches a certain value, it is said that a triggering condition has been satisfied. This is in fact the case in our illustrative example, as the location of the entity is right next to the border of a shaded region. Thus, the entity will asexually self-reproduce some offspring entities within its local region.



An Illustration of Autonomous Entities

At the following steps, the offspring will diffuse to new locations. By doing so, some of them will encounter new border feature locations as well and thereafter self-reproduce more entities. On the other hand, the entities that cannot find any border feature locations after a given number of diffusion steps will be automatically turned off [Liu, 2001].

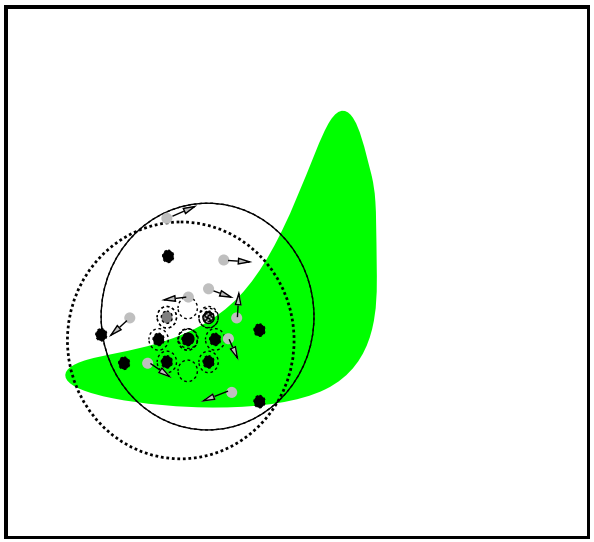
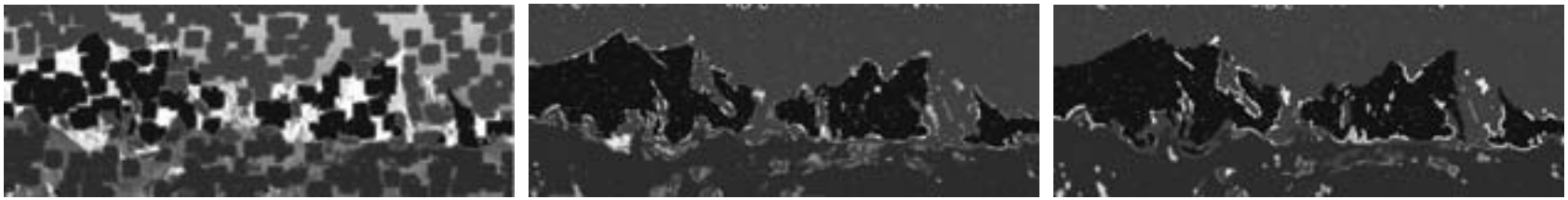


Image Segmentation Problem

Segmenting a landscape image that contains three complex-shaped homogeneous regions. 1,500 entities, evenly divided into three classes, are randomly distributed over the given image.

(a) $t = 0$ (b) $t = 1$ (c) $t = 2$



(d) $t = 5$ (e) $t = 10$ (f) $t = 50$

Computational Steps

In the AOC-based image segmentation, the computational steps required can be estimated by counting active entities over time (i.e., the entities whose ages do not exceed a given life span).
The total number of active entities (i.e., computational steps) involved in extracting a homogeneous region is less than the size of the given image, $526 \times 197 = 103,622$.

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

- Collective world modeling with a group of mobile robots in an unknown, less structured environment [Liu and Wu, 2001].
- The goal is to enable mobile robots to cooperatively perform a map building task with fewer sensory measurement steps, to construct a potential field map as efficiently as possible.

Self-Organization I

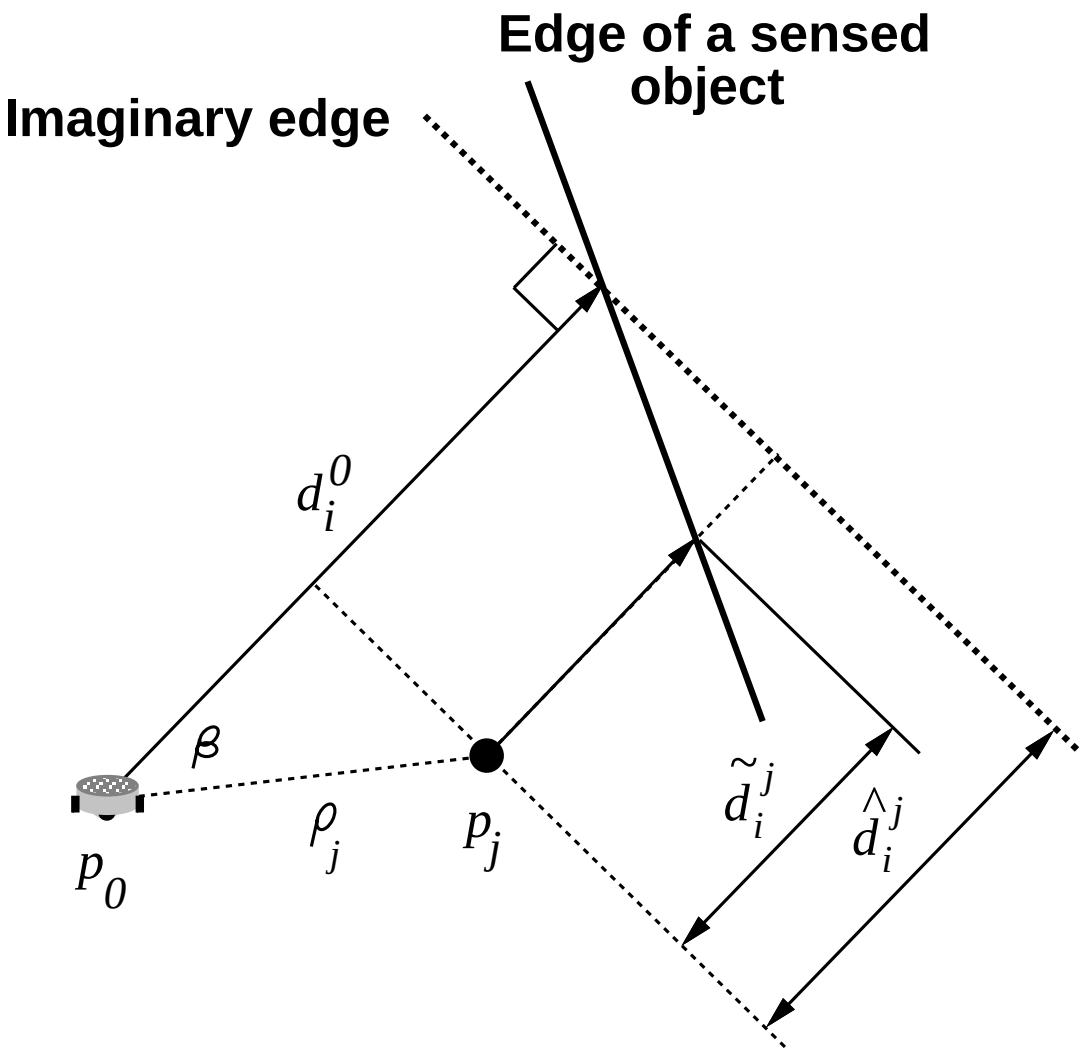
Suppose that a robot moves to position p_0.

- measure its distances to the surrounding obstacles of its environment in several directions (n).
- record measurements in a sensing vector, $\mathcal{S}_0 = [d_1^0, d_2^0, \cdots, d_i^0, \cdots, d_n^0]$, with respect to position p_0 where d_i^0= distance between position p_0 and an obstacle sensed in the ith direction.

Self-Organization II

- associate this information to its adjacent positions. The estimated proximity of any position p_j inside the neighboring region of p_0 to a sensed obstacle = $\hat{d}_i^j = d_i^0 - \rho_j \cdot cos\beta \ \ (i = 1, 2, \cdots, n)$, where $\beta = \alpha_0^{(i)} - \alpha_j$. $\alpha_0^{(i)}$ and α_j= polar angles of the sensing direction and of position p_j. $\hat{d}_i^j$= estimate for p_j based on the ith direction sensing value. d_i^0= current measurement taken from p_0 in the ith direction.
  Estimated proximity values for position p_j can be written as $\hat{\mathcal{S}}_j = \left[\hat{d}_1^j, \hat{d}_2^j, \cdots, \hat{d}_i^j, \cdots, \hat{d}_n^j\right]$.

The Distance Association Scheme [Liu, 2001]



Potential Field Estimate I

- A confidence weight for each element of $\hat{\mathcal{S}}_j = w_j = e^{-\eta\rho_j^2}$, where ρ_j = distance between the robot and position p_j.
- Potential field estimate at position p_j:

$$\hat{U}_j^t = \sum_{i=1}^{n} e^{-\lambda \hat{d}_i^j}, \tag{1}$$

Potential Field Estimate II

- At time t, a set of potential field estimates, $\Omega_t^j = \{\hat{U}_j^{t_1}, \hat{U}_j^{t_2}, \cdots, \hat{U}_j^{t_i}, \cdots, \hat{U}_j^{t_k}\}$, can be derived by k robots with respect to position p_j,

$$\Omega_t^j \longleftarrow \Omega_{t-1}^j \bigcup \mathcal{Q}, \tag{2}$$

  where Ω_{t-1}^j denotes a set of potential field estimates for position p_j at time $t-1$, and $\mathcal{Q} = \hat{U}_j^{t_k}$, where subscript k indicates that the potential value is estimated based on the measurement of the kth robot.
- Ω_t^j is associated with a confidence weight set: $W_t^j = \{w_j^{t_1}, w_j^{t_2}, \cdots, w_j^{t_i}, \cdots, w_j^{t_k}\}$.

Acceptable Potential Field Value

$$U_j^t = \begin{cases} \hat{U}_j^{t_i}, & \exists i \in [1,k], w_j^{t_i} = 1, \\ \sum_{i=1}^{k} \hat{U}_j^{t_i} \cdot \bar{w}_j^{t_i}, & otherwise, \end{cases} \tag{3}$$

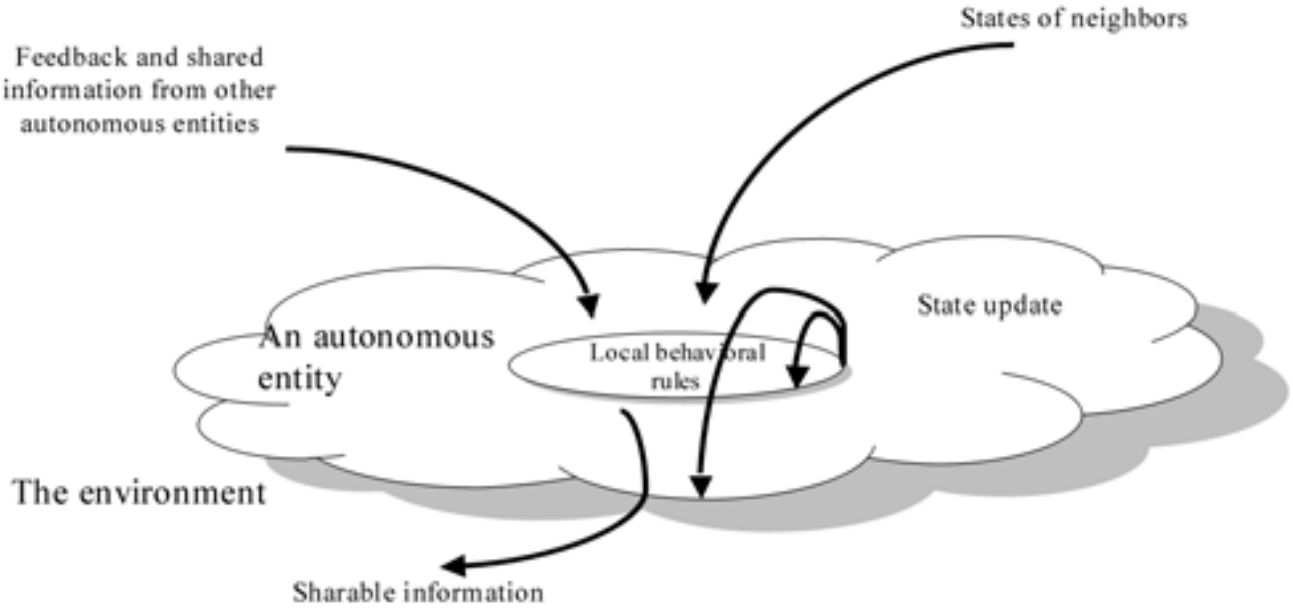
where $\bar{w}_j^{t_i}$ denotes a normalized weight component of W_t^j, i.e.,

$$\bar{w}_j^{t_i} = \frac{w_j^{t_i}}{\sum_{n=1}^{k} w_j^{t_n}}. \tag{4}$$

Adaptation

- The efficiency of the self-organization based collective world modeling can be optimized.
- It is possible to define an adaptation mechanism for distributed autonomous robots to dynamically generate/modify their group cooperative behavior based on some group performance criteria.
- The selected (i.e., high fitness) cooperative behavior is used to control autonomous robots in their interactions with the environment.

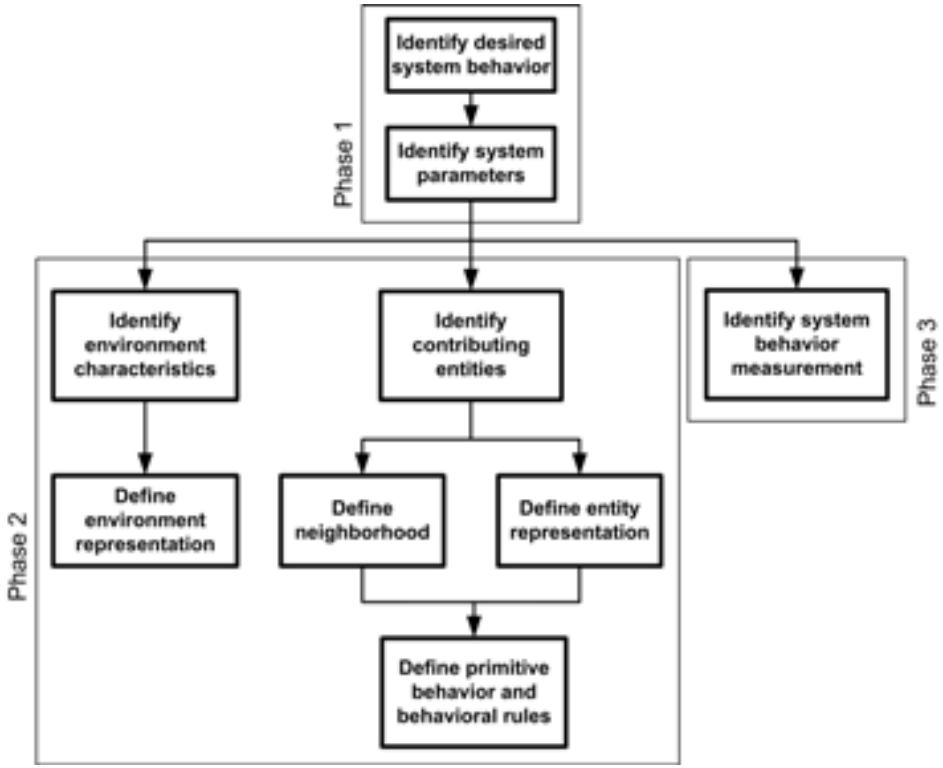
Functional Modules in an Autonomous Entity



- Autonomous entities need to modify their behavioral rules over time in order to adapt to a dynamically changing environment.
- This is the learning capability of autonomous entities.
- Randomness in the decision making process enables an entity to explore uncharted territories or to resolve conflicts.

- There are a collection of autonomous entities. The right level of abstraction has to be chosen so that autonomous entities can be identified.
  This does not exclude the *repeated* application of this technique to suit the need of the specific problem that can be best modeled by *multiple levels* of abstraction.
- There are some relationships between autonomous entities in the form of constraints, such as limitations on the position inhabitable by a queen in an n-queen problem.
- A performance measurement is available to assess the quality of any solution.
  It can be used in AOC as a guideline to direct the behavior of autonomous entities.

The Major Phases in Developing an AOC System



AOC can be viewed as *a methodology for engineering* a computing system to solve hard computational problems or model complex systems.

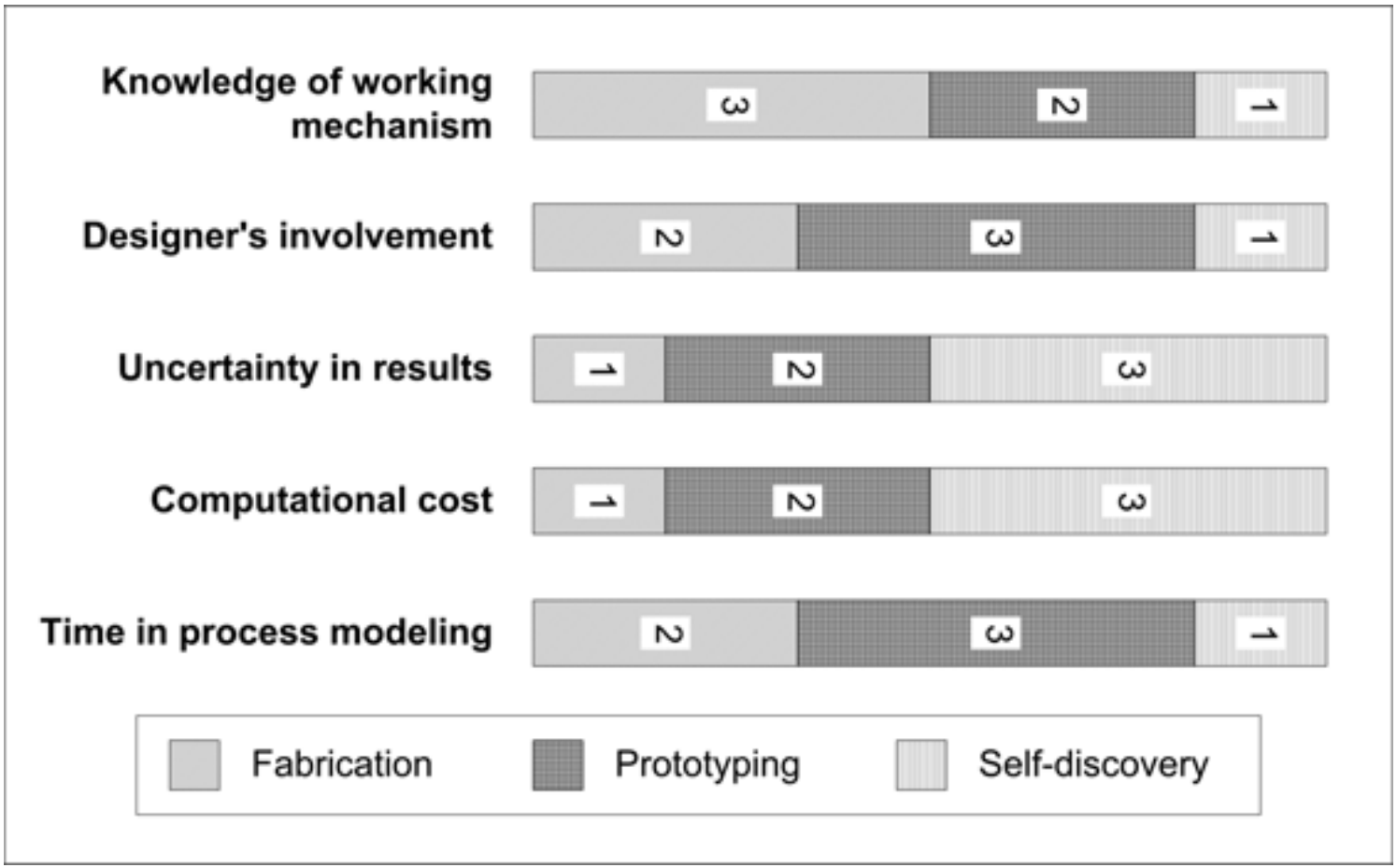
Varying Degrees of Human Involvement

From an engineering point of view, AOC systems differ in at least five aspects:

1. **Knowledge of working mechanism:** Analogies are drawn from the natural or physical world.
2. **Designer's involvement:** More knowledge means more involvement in the implementation.
3. **Uncertainty in results:** It varies according to the knowledge of the actual working mechanism.
4. **Computational cost:** Computational cost refers to the time and space complexity of an algorithm.
5. **Time in process modeling:** It hitches on the complexity of the problem to be solved or the system to be modeled as well as the degree of understanding.

Ranking AOC based on Engineering Requirements

(1 being the easiest and 3 the hardest)



Homogeneity Individuals may differ only in the parameters for characterizing their goals, behaviors, and behavioral rules, but not in their structures.
In the AOC-based image segmentation, although entities are classified into three different classes, they are homogeneous in that they differ only in the parameters for describing their goals.

Simplicity The behavior model of each autonomous entity is simple.

Locality The interactions among autonomous entities are strictly local although the notion of locality can be physical as well as logical.

Implicity Another form of interaction comes from the implicit knowledge sharing among autonomous entities via their common environment.

Uncertainty Behavior is not purely deterministic.

Amplification Desirable behavior is amplified while undesirable behavior is eliminated via mechanisms, such as birth and death – positive feedback.

Recursion Emergent autonomy results from the aggregation of primitive autonomy of entities through iterations.
A system exhibiting such emergent autonomy can serve as the basic element of a more complex system that may show its own emergent autonomy.

Scalability AOC envisions an environment able to scale up or down according to the ever changing needs of a dynamical complex system.

Openness New types of autonomous entity can be accommodated seamlessly into an AOC system.

- Generality measures the applicability of an algorithm to different problem domains.
- Robustness concerns the sensitivity of an algorithm in terms of its parameter settings.
- Completeness of an algorithm assesses its ability to search the whole solution space.
- Efficiency measures the effectiveness of an algorithm to find an optimal solution and how quickly such a solution is found (usually reflected in the computational cost of an algorithm).
- Computational cost refers to the requirements on computational cycles and memory space in the process of finding a solution.

Other Factors affecting AOC Performance

- Randomness is an important factor in any self-organizing system, such as those formulated according to the AOC principles.
- Emergence is a property of AOC that is not pre-programmable.
- The complexity of AOC can have a direct implication on whether or not a problem is solvable.
- The evolvability of AOC [Nehaniv, 2000] refers to its ability to evolve an optimal solution.

Autonomy Oriented Computing (AOC) System

An AOC system is a tuple $\langle \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \cdots, \mathbf{e}_N\}, \mathbf{E}, \blacksquare \rangle$, where $\{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \cdots, \mathbf{e}_N\}$ = group of autonomous entities; $\mathbf{E}$ = environment in which entities reside; $\blacksquare$ = system objective function, which is usually a nonlinear function of entity states.

Environment

- An environment plays its communication medium role through its state changes as caused by the primitive behavior.
- Environment **E** is characterized by a set $\mathcal{ES} = \{es_1, es_2, \cdots, es_i, \cdots, es_{N_{\mathcal{ES}}}\}$, where each $es_i \in D_{es_i}$ = static or dynamical attribute; $N_{\mathcal{ES}}$ = the number of attributes. At each moment, $\mathcal{ES}$ = current state of environment **E**.
- The state space of **E** = $D_{\mathcal{ES}} = D_{es_1} \times D_{es_2} \times \cdots \times D_{es_i} \times \cdots \times D_{es_{N_{\mathcal{ES}}}}$.

Autonomous Entities

- An autonomous entity **e** is a tuple $\langle \mathcal{S}, \mathcal{F}, \mathcal{G}, \mathcal{B}, \mathcal{R} \rangle$, where $\mathcal{S}$ = current state of entity **e**; $\mathcal{F}$ = evaluation function; $\mathcal{G}$ = goal set of entity **e**; $\mathcal{B}$ and $\mathcal{R}$ = primitive behaviors and behavioral rules.
- Based on the differences in $\mathcal{S}, \mathcal{F}, \mathcal{G}, \mathcal{B}$, and $\mathcal{R}$, entities in an AOC system can be categorized into *different classes*.

Neighbors

- The neighbors of entity **e** are a group of entities $\mathcal{L}^e = \{I_1^e, I_2^e, \cdots, I_i^e, \cdots, I_{N_\mathcal{L}}^e\}$,
  where $N_\mathcal{L}$ = the number of neighbors.
- The relationship (e.g., distance) between each neighbor I_i^e and entity **e** satisfies certain application-dependent constraint(s).
- In different AOC systems, the neighbors of an entity can be *fixed* or *dynamically changed*.

State

- State $\mathcal{S}$ of autonomous entity **e** is characterized by a set of static or dynamical attributes, i.e., $\mathcal{S} = \{s_1, s_2, \cdots, s_i, \cdots, s_{N_\mathcal{S}}\}$, where $s_i \in D_{s_i}$ and $N_\mathcal{S}$ denotes the number of attributes.
- $D_\mathcal{S} = D_{s_1} \times D_{s_2} \times \cdots \times D_{s_i} \times \cdots \times D_{s_{N_\mathcal{S}}}$ corresponds to the state space of entity **e**.

Evaluation

Before an entity fires its behavioral rules to select its primitive behavior, it assessed its current condition (i.e., its own internal state and/or those of its neighbors and environment).

Evaluation function Autonomous entity **e** assesses its conditions using one of the following evaluation functions:

- Internal state:

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

- State of environment:

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

- States of neighbors:

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

- Internal state and that of environment:

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

- Internal state and those of neighbors:

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

where R = range of function $\mathcal{F}$ (e.g., the set of real numbers or integers). $\hat{D}_{\mathcal{S}}$ = Cartesian product of elements in a subset of $\{D_{s_i}\}$, i.e., $\hat{D}_{\mathcal{S}} \subseteq D_{\mathcal{S}}$.
Similarly, $\hat{D}_{\mathcal{ES}}$ = Cartesian product, $\hat{D}_{\mathcal{ES}} \subseteq D_{\mathcal{ES}}$. I_i^e = ith neighbor of entity **e**. $\mathcal{S}^{I_i^e}$ and $D_{\mathcal{S}^{I_i^e}}$ = the current state and the state space of entity I_i^e. $\hat{D}_{\mathcal{S}^{I_i^e}} \subseteq D_{\mathcal{S}^{I_i^e}}$. $\prod$ = Cartesian product operator.

Goal

Generally speaking, the primitive behavior of entities in AOC systems is goal directed.

- An entity, **e**, can be engaged in a set of goals over time, as denoted by $\mathcal{G} = \{g_1, g_2, \cdots, g_i, \cdots, g_{N_\mathcal{G}}\}$, where $N_\mathcal{G}$ denotes the number of goals.
- Each goal g_i is to achieve a certain state $\mathcal{S}'$ such that evaluation function $\mathcal{F}$ takes a certain predefined value α, i.e., $g_i = \{\mathcal{S}' | \mathcal{F}(\cdot) = \alpha\}$, where α is a constant.

Primitive Behavior

An entity, **e**, can perform a set of primitive behaviors, $\mathcal{B} = \{b_1, b_2, \cdots, b_i, \cdots, b_{N_\mathcal{B}}\}$, where $N_\mathcal{B}$ denotes the number of primitive behaviors.
Each primitive behavior b_i is a mapping in one of the following forms:

- Self-reproduce:

$$b_i : \mathbf{e} \longrightarrow \mathbf{e}^m, \tag{10}$$

  which is a reproduction-like behavior. It denotes that entity **e** replicates itself m times (i.e., breed m offspring);

- Die:

$$b_i : \mathbf{e} \longrightarrow \emptyset, \tag{11}$$

  which denotes that entity **e** vanishes from the environment;

- Change internal state:

$$b_i : \hat{D}_\mathcal{S} \longrightarrow \hat{D}_\mathcal{S}, \tag{12}$$

- Change state of environment:

$$b_i : \hat{D}_{\mathcal{ES}} \longrightarrow \hat{D}_{\mathcal{ES}}, \tag{13}$$

- Change internal state and that of environment:

$$b_i : \hat{D}_{\mathcal{S}} \times \hat{D}_{\mathcal{ES}} \longrightarrow \hat{D}_{\mathcal{S}} \times \hat{D}_{\mathcal{ES}}, \text{ or} \tag{14}$$

- Change internal state and those of neighbors:

$$b_i : \hat{D}_{\mathcal{S}} \times \prod_{I_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{I_i^e}}) \longrightarrow \hat{D}_{\mathcal{S}} \times \prod_{I_i^e \in \mathcal{L}^e} (\hat{D}_{\mathcal{S}^{I_i^e}}), \tag{15}$$

Behavioral Rule

The behavioral rule set for entity **e** is $\mathcal{R} = \{r_1, r_2, \cdots, r_i, \cdots, r_{N_\mathcal{R}}\}$, where $N_\mathcal{R}$ denotes the number of rules. Each behavioral rule r_i is to select one or more primitive behaviors to perform. Behavioral rules can be classified into two types:

- Evaluation-based rules:

$$r_i : Ran(\mathcal{F}) \longrightarrow \{\hat{\mathcal{B}}\}, \tag{16}$$

  where $Ran(\mathcal{F})$ denotes the range of evaluation function $\mathcal{F}$. $\hat{\mathcal{B}} \subseteq \mathcal{B}$. $\{\hat{\mathcal{B}}\} \subseteq 2^\mathcal{B}$.

- Probability-based rules:

$$r_i : [0, 1] \longrightarrow \{\hat{\mathcal{B}}\}, \tag{17}$$

  where each subset $\hat{\mathcal{B}}$ is assigned a probability, $p_{\hat{\mathcal{B}}}$, which may be fixed or dynamically changed over time. This type of rule probabilistically selects a set of primitive behaviors from $\mathcal{B}$.

System Objective Function

- As a global measurement for the performance of an AOC system, system objective function ■ guides the system to evolve towards certain desired states or patterns.
- ■ is defined as a function of the states of some entities and can be categorized into two types.
  Let $\{\mathbf{e}_i\}$ be a group of entities.
  - State-oriented function:

$$\blacksquare : \prod_{e_k \in \{\mathbf{e}_i\}} \hat{D}_{\mathcal{S}^{e_k}} \longrightarrow R^m, \tag{18}$$

    where $\hat{D}_{\mathcal{S}^{e_k}}$ is a subset of the state space, $D_{\mathcal{S}^{e_k}}$, of entity $\mathbf{e}_k$, R is the set of real numbers or integers, and m denotes the dimensionality of the space of ■.
  - Process-oriented function:

$$\blacksquare : \left\{ \prod_{e_k \in \{\mathbf{e}_i\}} \hat{D}_{\mathcal{S}^{e_k}} \right\}^{\Pi} \longrightarrow R^m, \tag{19}$$

Interactions between Entities and their Environment

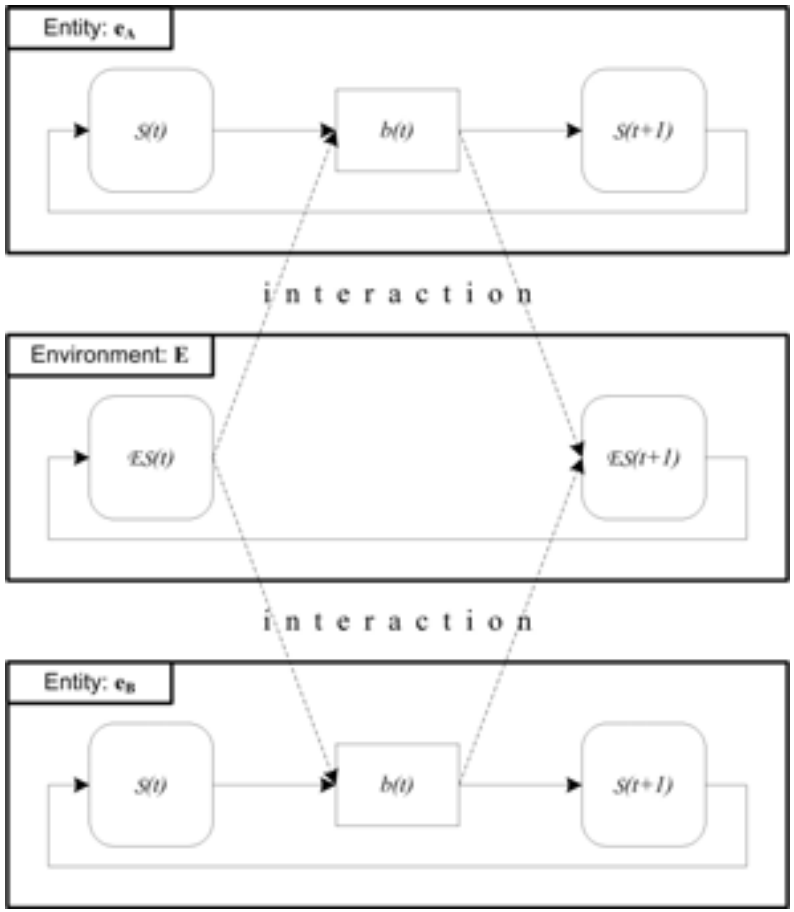
The interactions between entity **e** and its environment **E** are modeled as a sequence of mappings $\{\mathcal{I}_{\mathbf{eE}}\}$, where $\mathcal{I}_{\mathbf{eE}}$ has one of the following forms:

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

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

where '$\longrightarrow_{b_i}$' indicates that $\mathcal{I}_{\mathbf{eE}}$ is in fact a primitive behavior, b_i, of entity **e**, by performing which entity **e** can change the state of its environment.

Interactions (i.e., the dashed lines) between two entities $\mathbf{e_A}$ and $\mathbf{e_B}$ and their environment **E**, which are caused by primitive behavior (i.e., $b(t)$). The solid lines denote the state changes in entities and their environment.



Interactions among Entities

The direct interactions between entities $\mathbf{e_A}$ and $\mathbf{e_B}$ are modeled as a sequence of mapping tuples $\{\langle \mathcal{I}_{AB}, \mathcal{I}_{BA} \rangle\}$,

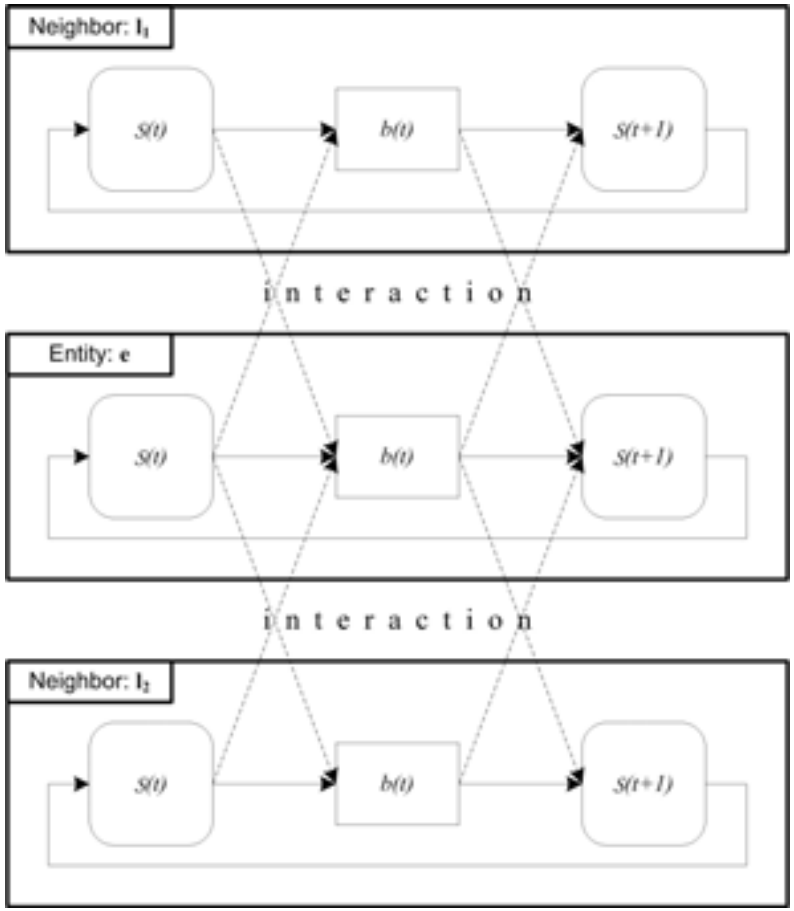
$$\mathcal{I}_{AB} : \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{S}^B} \longrightarrow_{b^A} \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{S}^B}, \tag{22}$$

and

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

where '$\longrightarrow_{b^A}$' and '$\longrightarrow_{b^B}$' denote that $\mathcal{I}_{AB}$ and $\mathcal{I}_{BA}$ are two primitive behaviors of entities $\mathbf{e_A}$ and $\mathbf{e_B}$, respectively, which are related to the states of neighboring entities. Here, if we take $\mathbf{e_A}$ and $\mathbf{e_B}$ as neighbors of each other, we can regard that '$\longrightarrow_{b^A}$' and '$\longrightarrow_{b^B}$' are originated from the equations above. $\hat{D}_{\mathcal{S}^A}$ and $\hat{D}_{\mathcal{S}^B}$ are subsets of the state spaces of entities $\mathbf{e_A}$ and $\mathbf{e_B}$, respectively.

Direct interactions (i.e., the dashed lines) between entity **e** and its two neighbors, namely, entities $\mathbf{l_1}$ and $\mathbf{l_2}$. The solid lines denote the state changes in the three entities as caused by their primitive behaviors.



The indirect interactions between entities $\mathbf{e_A}$ and $\mathbf{e_B}$ are modeled as a sequence of mapping tuples $\{\langle \mathcal{I}_{AE}, \mathcal{I}_{BE} \rangle\}$, where interaction $\mathcal{I}_{AE}$ between entity $\mathbf{e_A}$ and environment $\mathbf{E}$ occurs before interaction $\mathcal{I}_{BE}$ between entity $\mathbf{e_B}$ and environment $\mathbf{E}$. That is,

- At time t:

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

$$\mathcal{I}_{AE} : \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{ES}} \longrightarrow_{b^A} \hat{D}_{\mathcal{S}^A} \times \hat{D}_{\mathcal{ES}}; \tag{25}$$

- At time t':

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

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

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

What is Self-Organization?

- The term “self-organization” was first introduced by Ashby in 1947 [Ashby, 1947, Ashby, 1966].
- The phenomenon of self-organization exists in a variety of natural systems, such as galaxies, planets, compounds, cells, organisms, stock markets, and societies
  [Bak, 1996, Ünsal, 1993, Lucas, 1997]. It is involved in many disciplines, including biology, chemistry, computer science, geology, sociology, and economics
  [Liu, 2001, Bak, 1996, Kauffman, 1993].
- Several theories have been developed to describe self-organizing systems. They include the dissipative structure theory of Prigogine [Nicolis and Prigogine, 1977, Prigogine, 1980] and the synergetics theory of Haken [Haken, 1983a, Haken, 1983b, Haken, 1988].

The behavior of entities in a self-organizing system can be generalized into three steps [Ünsal, 1993].

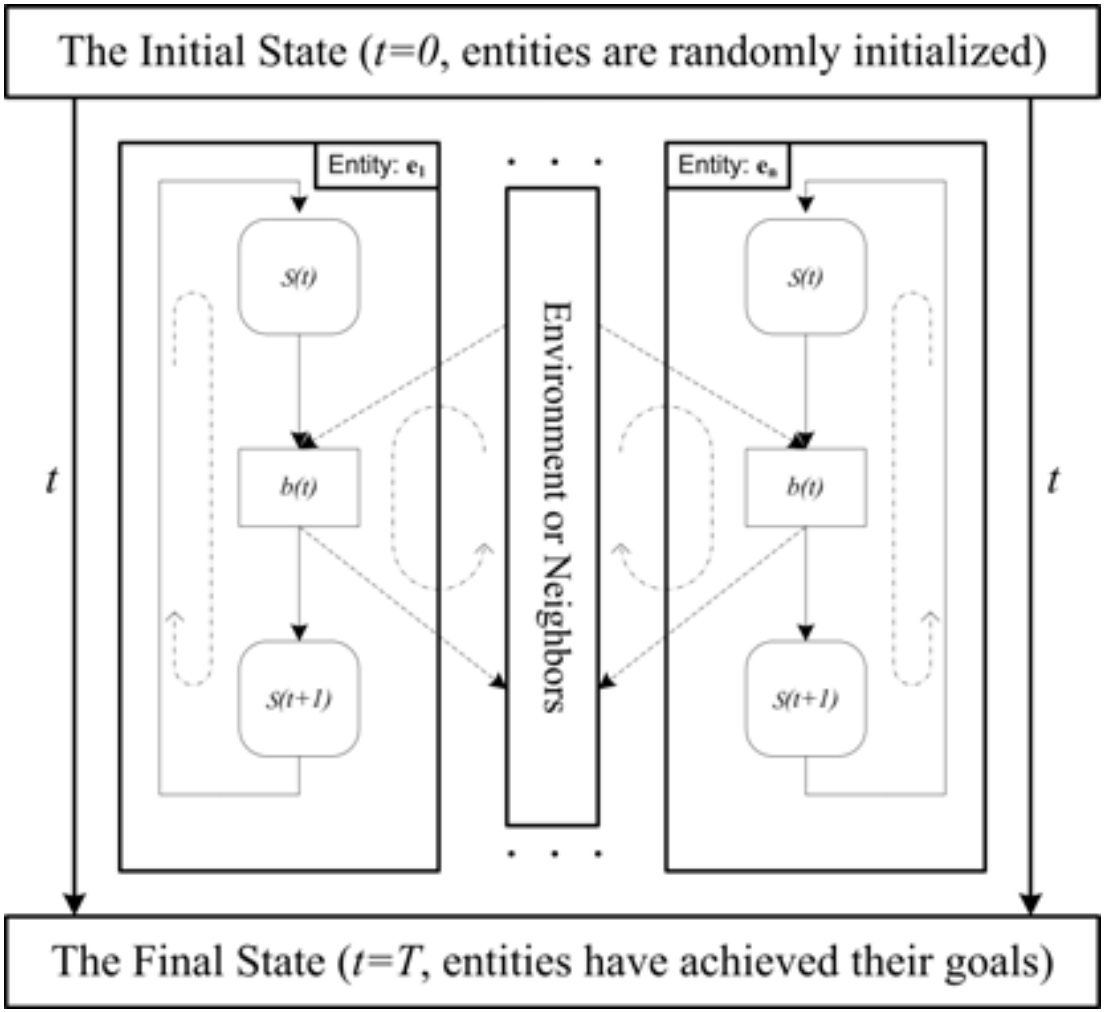
- Entities sense the environment or receive signals from other entities.
- Based on the information received, entities make rational decisions on what to do next.
- Entities behave according to their decisions. Their behavior will in turn affect the environment and the behavior of other entities. By following these three steps, an entity carries out its interaction with its environment or other entities.

How Does an AOC System Self-Organize?

Let us take the AOC-based search as an example.

- An entity tries to find a pixel that belongs to a certain homogeneous region. At such a pixel, its evaluation value will be better than those at other pixels.
- When an entity finds a desired pixel, it will reproduce some offspring within its local environment, where the offspring will most probably find other desired pixels – successful behavior is aggregated and amplified.
- If an entity cannot find a desired pixel after predefined steps, i.e., its lifespan, it will be deactivated – those entities with poor performance are eliminated.
- As the search progresses, more entities which are either reproduced or diffusing are able to locate pixels of a homogeneous region that has been found – a nonlinear process

Self-Organization in an AOC System



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

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

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

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

- Globally, for the whole system,

$$Pr\left(\mathbf{X}_{t+1} - \mathbf{X}_t \succ 0\right) > 0, \tag{29}$$

where $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$ denote the value vectors of system objective function $\mathbf{X}$ at time t and $t+1$, respectively. $\mathbf{X}_{t+1} - \mathbf{X}_t \succ 0$ means that the system evolves to a better state at time $t+1$.

Autonomy Oriented Computing

Step 1. Initially, design an AOC system $\langle \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_i, \cdots, \mathbf{e}_N\}, \mathbf{E}, \mathbf{X} \rangle$, including (i) environment states $\mathcal{ES}$; (ii) entity states $\mathcal{S}$; (iii) evaluation function $\mathcal{F}$; (iv) goals $\mathcal{G}$; (v) behavior set $\mathcal{B}$; (vi) behavioral rule set $\mathcal{R}$; (vii) system objective function $\mathbf{X}$.

Step 2. Determine the desired 'value' $\mathbf{X}^*$ of $\mathbf{X}$.

Step 3. Execute the AOC system, and then evaluate the current 'value' $\mathbf{X}'$.

Step 4. Define an optimization function $\mathbf{Y} = |\mathbf{X}' - \mathbf{X}^*|$ as a guidance for autonomy oriented computing.

Step 5. If $\mathbf{Y}$ is not optimized, update the parameters of entities or the environment according to $\mathbf{Y}$ (note: in problem solving, 'optimized' means the system can successfully and efficiently find a solution to the problem. In system modeling, 'optimized' means the prototyping system can actually simulate the system to be modeled).

Step 6. Repeat the above steps until $\mathbf{Y}$ is optimized.

- Autonomous entities and an environment are the key elements in AOC.
- Interactions between entities and their environment are the force that drives an AOC system to evolve towards certain desired states.
- Self-organization is the essential process of its working mechanism.

Based on the framework, we know:

1. How to formally characterize autonomous entities?
2. How to design and characterize an environment based on a task at hand?
3. How to design the interactions between autonomous entities and their environment To facilitate the aggregation of behavioral effects of entities?
4. How to design the primitive behaviors and behavioral rules of entities in order to achieve the self-organization of entities and emerge certain desired states or patterns?

- Many problems in Artificial Intelligence (AI) as well as in other areas of computer science and engineering can be formulated into CSPs or SATs.
- Examples: spatial and temporal planning, qualitative and symbolic reasoning, decision support, computational linguistics, scheduling, resource allocation and planning, graph problems, hardware design and verification, configuration, real time systems, robot planning, block world planning, circuit diagnosis, vision interpretation, and theorem proving.
- Problem solving is a domain with which many multi-agent applications are concerned. These applications are aimed at tackling computational problems in a distributed setting. In many cases, the problems to be solved are inherently distributed in nature [Ferber, 1999]. One way to formulate such problems is to treat them as distributed CSPs.

e-Learning

- Goal: To provide learning and training contents to learners via some electronic means, such as computers, Intranet, and Internet.
- Modules: Contents are usually structured into different modules, called learning objects, at different granularities, such as fragments (e.g., picture, figure, table, text), lessons, topics, units, courses, and curricula
  [Loser et al., 2002, IEEE, 2001, IEEE, 2002]. Several smaller granular objects can constitute a bigger granular object.
  The reason for doing so is to increase the interoperability and scalability of an e-learning service environment
  [Loser et al., 2002].
- Characteristics: Learning objects are located on different, usually geographically distributed, computers.

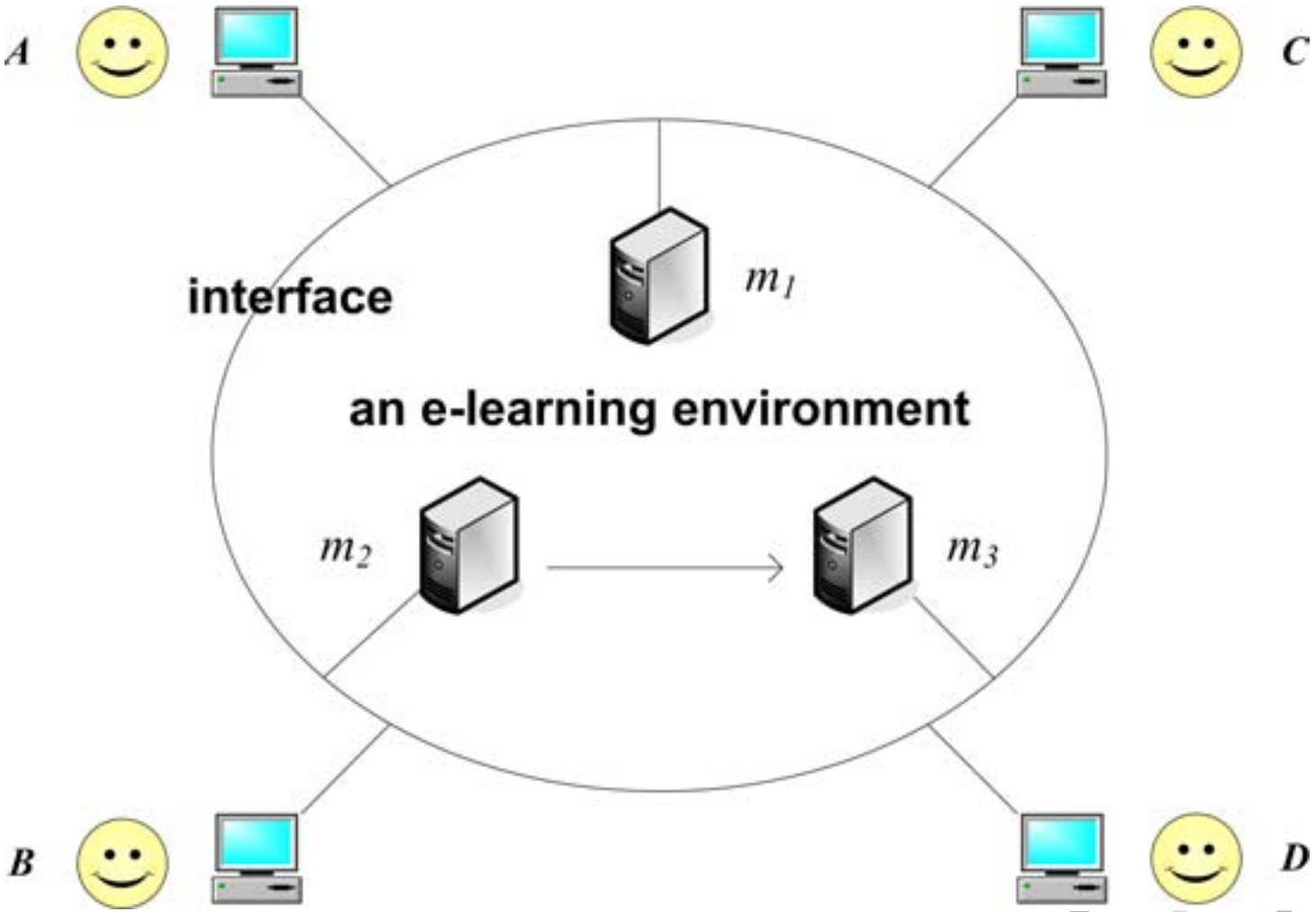
e-Learning Constraints

- **Content constraints** (i.e., relationship among learning objects [Loser et al., 2002]): (1) content relations for the semantic interdependency among modules, (2) ordering/sequence relations for accessing modules.
- **System constraints** (i.e., hardware and software of the service environment): Although a learning object in an e-learning environment can be accessed by one or more learners each time, the number of learners is limited because of some specific factors, such as hardware bandwidth.

An e-Learning Service Scenario

- Four learners, A, B, C, and D are accessing the e-learning service via clients connected to the service environment.
- The learners need to learn three knowledge modules, i.e., m_1, m_2, and m_3, provided by three geographically distributed computers. The constraints in the environment are:
  1. Content constraint: To learn modules m_2 and m_3, module m_2 must be learned before module m_3.
  2. System constraint: One module can serve at most two learners at any time.
- To successfully provide services, the distributed clients should collaborate with one another to determine a service sequence for each learner.

Four learners need to learn three modules, where module m_1 is independent, while modules m_2 and m_3 are dependent: Module m_2 has to be learned before module m_3. At any time, each module can only be accessed by at most two learners.



The Collaborative Service

- For each learner $L \in \{A, B, C, D\}$, its client should coordinate with those of other learners to generate a combination $\{S_L^1, S_L^2, S_L^3\}$ of $\{1, 2, 3\}$ ($\forall i$, $S_L^i \in \{1, 2, 3\}$), which will be used as the sequence in which learner L accesses the knowledge modules.
- For example, if learner A is assigned a combination $\{2, 3, 1\}$, it means learner A should access the three modules in a sequence of 2, 3, and 1.

Constraint Formulation

- Constraints:
  1. $\forall L \in \{A, B, C, D\}$, $S_L^i \in \{1, 2, 3\}$, and for $i \neq j$, $S_L^i \neq S_L^j$.
  2. Content constraint: $\forall L \in \{A, B, C, D\}$, if $S_L^i = 2$ and $S_L^j = 3$, then it should guarantee $i < j$.
  3. System constraint: $\forall i, j \in \{1, 2, 3\}$, $\sum_{L \in \{A,B,C,D\}} T(S_L^i = j) \leq 2$, where $T(\cdot)$ is a Boolean function to test whether or not an input proposition is true.
- The collaborative service: A distributed CSP, where distributed clients are responsible for assigning values to the variables of their learners.

CSP and SAT

- Constraint satisfaction problem (CSP) and satisfiability problem (SAT) are two types of classical NP-hard problem.
- **Constraint satisfaction (CSP):** A constraint satisfaction problem (CSP), P, consists of:
  1. A finite set of variables, X = $\{x_1, x_2, \cdots, x_i, \cdots, x_n\}$.
  2. A domain set, containing a finite number of discrete domains for variables in X: D = $\{D_1, D_2, \cdots, D_i, \cdots, D_n\}$, $\forall i \in [1, n]$, $x_i \in D_i$.
  3. A constraint set, C = $\{C(R_1), C(R_2), \cdots, C(R_i), \cdots, C(R_m)\}$, where each R_i is an ordered subset of the variables, and each constraint $C(R_i)$ is a set of tuples indicating the mutually consistent values of the variables in R_i.

CSP Solution

A solution, S, to a CSP is an assignment to all variables such that the assignment satisfies all given constraints. Specifically,

1. S is an ordered set, $S = \langle v_1, v_2, \cdots, v_i, \cdots, v_n \rangle$, $S \in D_1 \times D_2 \times \cdots D_i \times \cdots \times D_n$.
2. $\forall i \in [1, m]$, S_i is an ordered value set corresponding to variables in R_i, and $S_i \subseteq S$, $S_i \in C(R_i)$.

n-Queen Problem: A Classical CSP Example

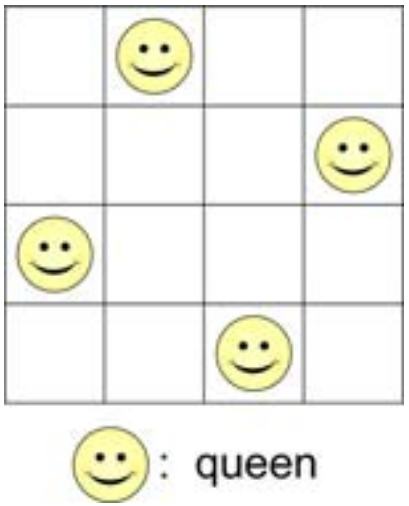
- It is regarded as a good benchmark for testing algorithms and has attracted attention in the CSP community [Sosic and Gu, 1994].
- This problem requires one to place n queens on an $n \times n$ chessboard, so that no two queens are in the same row, the same column, or the same diagonal.
- There exist solutions to n-queen problems with $n \geq 4$ [Bitner and Reingold, 1975, Sosic and Gu, 1994].

The Equivalent CSP

$$X = \{ x_1, x_2, \cdots, x_i, \cdots, x_n \}.$$

$$D = \{ D_1, D_2, \cdots, D_i, \cdots, D_n \}, \forall i, D_i = [1, n].$$

$$C = \{ C(R_u) | \forall i, j \in [1, n], C(R_u) = \{ \langle b, c \rangle | b \in D_i, c \in D_j, b \neq c, i - j \neq b - c, i - j \neq c - b \} \}.$$



SAT Problems

- An SAT is to test whether there is (at least) one solution for a given propositional formula.
- A satisfiability problem (SAT), P, consists of:
  1. A finite set of propositional variables, X = $\{x_1, x_2, \cdots, x_i, \cdots, x_n\}$.
  2. A domain set, D = $\{D_1, D_2, \cdots, D_i, \cdots, D_n\}$, $\forall i \in [1, n]$, $x_i \in D_i$ and $D_i = \{True, False\}$.
  3. A clause set, CL = $\{Cl(R_1), Cl(R_2), \cdots, Cl(R_i), \cdots, Cl(R_m)\}$, where each R_i is a subset of X, and each clause $Cl(R_i)$ is a disjunction of the literals corresponding to the variables in R_i.

SAT Solution

A solution, S, to an SAT is an assignment to all variables such that, under this assignment, the truth values of all given clauses are true, i.e.,

1. S is an ordered set, $S = \langle v_1, v_2, \cdots, v_i, \cdots, v_n \rangle$, $\forall i \in [1, n]$, $v_i \in \{True, False\}$, $S \in D_1 \times D_2 \times \cdots D_i \times \cdots \times D_n$.
2. $\forall i \in [1, m], T(Cl(R_i)) = True$, where $T(\cdot)$ is a function that returns the truth value of a clause.

The General Characteristics of AOC-Based Methods

- The domain of a CSP or its variant is represented into a multi-entity environment.
- The problem of finding a solution to a CSP is reduced to that of how a group of entities find a certain desired state by performing their primitive behaviors in such an environment.
- Like other AOC-based methods, ERE exhibits several unique characteristics as well as advantages in tackling problems that involve large-scale, highly distributed, locally interacting, and sometimes unreliable entities.

The ERE System

ERE Entity

An entity, *a*, in an ERE system is a virtual entity that represents one or more variables in a given problem. It essentially has the following abilities:

1. An ability to reside and behave (i.e., move around) according to a probability-based behavioral rule in a local environment as specified by the domains of the variable(s) it represents;
2. An ability to interact with its local environment;
3. An ability to be driven by certain goals.

ERE system

An ERE system is a system that contains the following elements:

1. An environment, E, as specified by the solution space of a problem at hand, in which entities reside;
2. A set of reactive rules (including primitive behaviors and behavioral rules), R, governing the behaviors of entities and the interactions among entities and their environment. Essentially, they govern entities to assign values to their respective variables. In this sense, they are the laws of the entity universe;
3. A set of ERE entities, $E = \{a_1, a_2, \cdots, a_i, \cdots, a_n\}$, which represent all variable of the problem.

General Ideas

- How can exact or approximate solutions to CSPs be self-organized by a multi-entity system consisting of E, R, and E.
- In other words, it will illustrate:

  Environment + Reactive rules + Entities $\Longrightarrow$ Problem solving

Basic Formulation

- We divide variables into groups. Each group contains one or more variables. A set of **entities** are employed to represent **variable groups**.
- The Cartesian products of the **domains** corresponding to all variable groups constitute an **environment** where entities reside. Each position of an entity indicates a value combination of the variables that it represents.
- An entity can move freely within a row and has its own primitive behaviors and behavioral rules.
- It tries to move to a position where the number of constraint violation is zero. The primitive behavior will locally determine how an entity moves and how the environment is updated.
- A solution state in ERE is reached when all entities (variable groups) can move to their zero-positions (consistent value combinations).

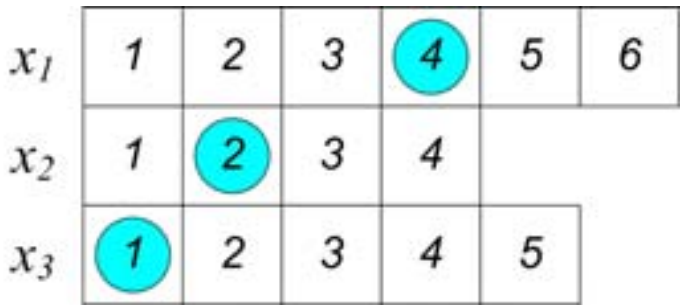
A CSP Example

X $=\{ x_1, x_2, x_3 \}$, $n = 3$.

D $=\{ D_1, D_2, D_3 \}$, $D_1 = \{1, 2, 3, 4, 5, 6\}$, $D_2 = \{1, 2, 3, 4\}$, $D_3 = \{1, 2, 3, 4, 5\}$.

C $=\{ x_1 \neq x_2, x_1 > x_3 \}$.

An Illustration of the Entity Model in the CSP Example



An SAT – a Special CSP

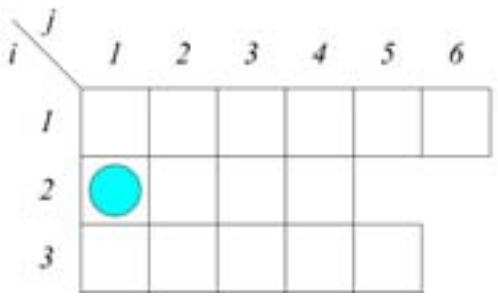
$$
\begin{aligned}
&\mathrm{X} = \{x_1, x_2, x_3, x_4\},\ n = 4.\\
&\mathrm{D} = \{D_1, D_2, D_3, D_4\},\ D_1 = D_2 = D_3 = D_4 = \{True, False\}.\\
&\mathrm{C} = \{\\
&\qquad T(x_1 \vee \neg x_2 \vee x_3) = True,\\
&\qquad T(x_1 \vee x_2 \vee \neg x_3) = True,\\
&\qquad T(x_2 \vee x_3 \vee \neg x_4) = True,\\
&\qquad T(\neg x_2 \vee \neg x_3 \vee x_4) = True,\\
&\qquad T(x_1 \vee x_3 \vee \neg x_4) = True,\\
&\qquad T(x_1 \vee x_3 \vee x_4) = True,\\
&\qquad T(\neg x_1 \vee x_2 \vee \neg x_3) = True,\\
&\qquad T(\neg x_1 \vee \neg x_2 \vee x_3) = True,\\
&\qquad T(\neg x_1 \vee \neg x_2 \vee \neg x_3) = True\\
&\}.
\end{aligned}
$$

An Illustration of the Entity Model in the SAT Example

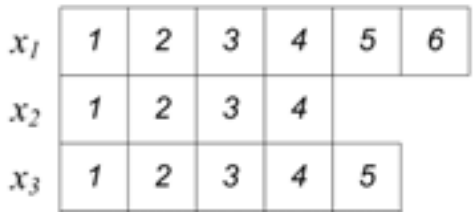
x_1, x_2	T,T	T,F	F,T	F,F
x_3, x_4	T,T	T,F	F,T	F,F

An Illustration of the ERE method in the CSP Example

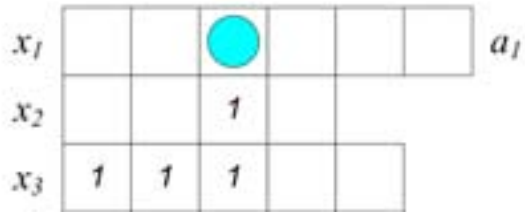
(a) The position of an entity; (b) the representation of domain values; (c)-(d) violation values marked in the environment.



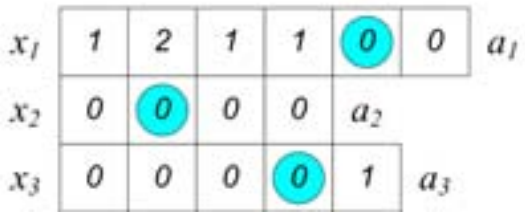
(h)



(i)



(j)



(k)

An Illustration of the ERE method in the SAT Example

(a) The representation of domain values; (b) violation values if entity a_1 is placed on (1, 1); (c) violation values of the whole environment.

x_1, x_2	T,T	T,F	F,T	F,F
x_3, x_4	T,T	T,F	F,T	F,F

(l)

x_1, x_2	●				a_1
x_3, x_4	1	2	1	1	

(m)

x_1, x_2	1	0	2	● 1	a_1
x_3, x_4	1	0	2	● 1	a_2

(n)

Primitive Behaviors

Least-move

- An entity moves to a minimum-position with a probability of *least-p*.
- If there exists more than one minimum-position, the entity chooses the first one on the left of the row.

$$\Psi(j,i) = \Phi(i). \tag{30}$$

In this function, the result has nothing to do with the current position j, and the maximum number of computational operations to find the position for each i is $|D_{i1} \times D_{i2} \times \cdots \times D_{ik}|$. We use a special symbol to represent this movement:

$$\Psi_{\_l}(j,i) = \Phi(i). \tag{31}$$

Primitive Behaviors

Better-move

- An entity moves to a position, which has a smaller violation value than its current position, with a probability of *better-p*.
- It will randomly select a position (using $Random(k)$ that complies with the uniform distribution to get a random number between 1 and k) and compare its violation value with that of its current position, and then decide whether or not to move to this new position.
- A better-move behavior can be defined using function $\Psi_{\_b}$:

$$\Psi_{\_b}(j,i) = \begin{cases} j, & \text{if } e(r,i).violation \geq e(j,i).violation, \\ r, & \text{if } e(r,i).violation < e(j,i).violation, \end{cases} \tag{32}$$

  where $r = Random(|D_{i1} \times D_{i2} \times \cdots \times D_{ik}|)$.
- The computational cost required for this primitive behavior is less than that of least-move.

Primitive Behaviors

Random-move

- An entity moves randomly with a probability of *random-p*.
- *random-p* will be relatively smaller than the probabilities of selecting least-move and better-move behaviors.
- It is somewhat like random walk in local search (to avoid getting stuck in, or to escape from, local optima).
- A random-move behavior can be defined using function $\Psi_{\_r}$:

$$\Psi_{\_r}(j,i) = Random(|D_{i1} \times D_{i2} \times \cdots \times D_{ik}|). \quad (33)$$

How to Update the Environment?

The violation values will be updated with:

- **Updating-rule 1: Remove-From** (j_1, i):
  For $(\forall i' \in [1, u])(\forall j' \in [1, |D_{i'1} \times D_{i'2} \times \cdots \times D_{i'k}|])$:
  If there are v constraints: (1) they are based on variables included in row_i and $row_{i'}$ and (2) their values are changed from false to true
  Then $e(j', i').violation \leftarrow e(j', i').violation - v$.
- **Updating-rule 2: Add-To** (j_2, i):
  For $(\forall i' \in [1, u])(\forall j' \in [1, |D_{i'1} \times D_{i'2} \times \cdots \times D_{i'k}|])$:
  If there are v constraints: (1) they are based on variables included in row_i and $row_{i'}$ and (2) their values are changed from true to false
  Then $e(j', i').violation \leftarrow e(j', i').violation + v$.

The ERE Algorithm I

$step = 0$;
Initialize positions and behavior probabilities of entities;
Initialize domain values and violation values of environment;
while *true* **do**
  for all $a_i \in A$ **do**
    Probabilistically determine a primitive behavior, b, to perform;
    Perform primitive behavior b;
    New position $(j'', i) = (\Psi(a_i.j, i), i)$;
    if current position $(a_i.j, i) = (j'', i)$ **then**
      Stay;
    else
      $a_i.j = j''$;
    end if
  end for

The ERE Algorithm II

Update violation values of environment;
if current state satisfies predefined stopping criterion **then**
  Output variable values corresponding to entity positions;
  break;
end if
step ++;
end while

An Illustrative SAT Example

X $=\{x_1, x_2, x_3, x_4, x_5\}$, $n = 5$.

D $=\{D_1, D_2, D_3, D_4, D_5\}$, $D_1 = D_2 = D_3 = D_4 = D_5 = \{True, False\}$.

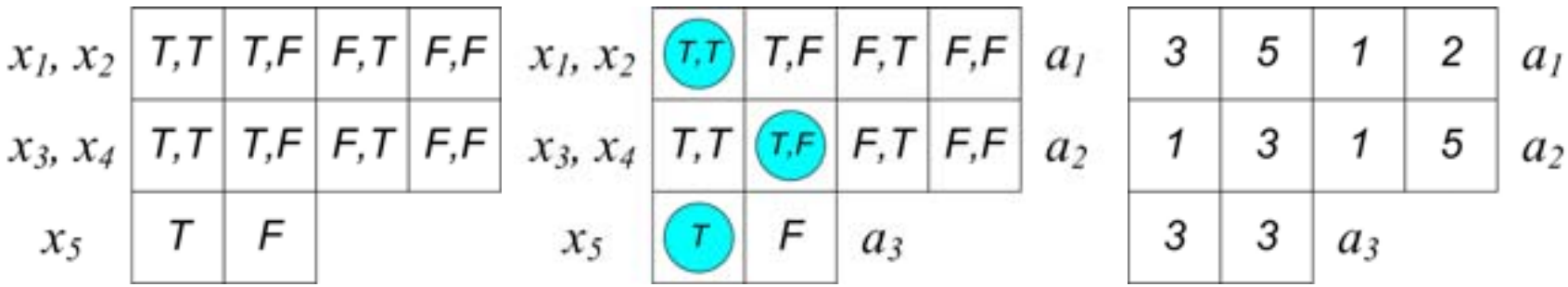
C= {

$$
\begin{aligned}
T(x_3 \vee x_4 \vee \neg x_5) &= True,\\
T(x_2 \vee \neg x_3 \vee \neg x_5) &= True,\\
T(\neg x_1 \vee \neg x_2 \vee x_3) &= True,\\
T(\neg x_1 \vee \neg x_2 \vee x_4) &= True,\\
T(\neg x_3 \vee x_4 \vee x_5) &= True,\\
T(x_1 \vee \neg x_2 \vee \neg x_3) &= True,\\
T(\neg x_2 \vee x_4 \vee x_5) &= True,\\
T(\neg x_1 \vee \neg x_3 \vee \neg x_5) &= True,\\
T(x_2 \vee \neg x_3 \vee x_4) &= True,\\
T(\neg x_1 \vee x_4 \vee \neg x_5) &= True,\\
T(x_2 \vee x_3 \vee x_5) &= True,\\
T(x_1 \vee x_2 \vee \neg x_4) &= True,\\
T(\neg x_1 \vee x_2 \vee \neg x_5) &= True,\\
T(\neg x_1 \vee x_3 \vee x_4) &= True,\\
T(x_1 \vee \neg x_4 \vee \neg x_5) &= True
\end{aligned}
$$

}.

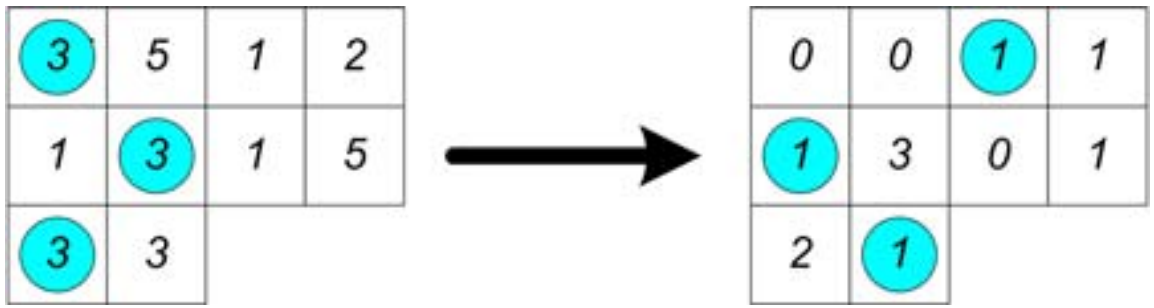
An Illustration of the ERE method in the SAT Example

(a) Domain values; (b) an initialized state; (c) violation values.



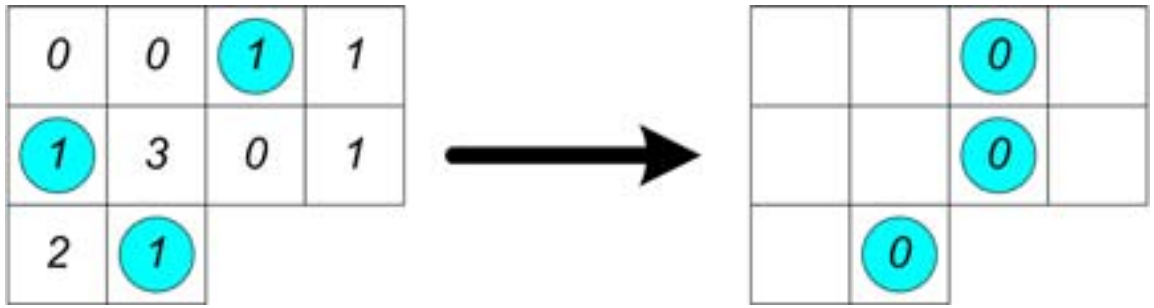
(o) (p) (q)

a_1, a_2, and a_3 perform least-move, least-move, and random-move behaviors, respectively.



(a) (b)

a_1 selects a better-move behavior, but fails to move. a_2 performs a least-move. a_3 also selects a least-move behavior, but fails to move.



(a) (b)

Characteristics of ERE

- ERE is very efficient in finding an approximate solution.
- After the first three steps, about $n - 7$ queens are at zero-positions in n-queen problems, and about 97-98% clauses are satisfied in SAT problems.
- Other characteristics of ERE:
  1. **Self-organizing:** The process of solving CSPs by ERE is entirely determined by the locality and parallelism of individual entities.
  2. **Indirectly interacting:** One entity may affect the whole environment. And, the change in the environment will in turn affect the movements of other entities.
  3. **Open and flexible:** New primitive behaviors or combinations of primitive behaviors can readily be created.

Conventional Methods

- The *generate-and-test (GT)* method generates each possible combination of variables systematically and then checks whether it satisfies all constraints, i.e., whether it is a solution – less efficient.
- The *backtracking (BT)* method assigns values to variables sequentially and then checks constraints for each variable assignment. If a partial assignment does not satisfy any of the constraints, it will backtrack to the most recently assigned variable and repeat the process again – exponential complexity.

Improved Methods I

- To avoid thrashing in BT [Gaschnig, 1979, Kumar, 1992], consistency techniques (Arc Consistency and k-Consistency) have been developed by some researchers [Cooper, 1989, Han and Lee, 1988, Kumar, 1992, Mackworth, 1977, Mohr and Henderson, 1986].
- To avoid both thrashing and redundant work in BT [Kumar, 1992], a dependency directed scheme and its improvements have been proposed [Bruynooghe, 1981, Kumar, 1992, Rossi et al., 1990, Stallman and Sussman, 1977].
- Some examples of systematic search for SATs are POSIT, TABLEAU, GRASP, SATZ, and REL_SAT [Hoos and Stützle, 1999], all of which are based on the Davis-Putnam (DP) algorithm [Davis et al., 1962].

Improved Methods II

- Stochastic and heuristic algorithms perform *local search* [Selman et al., 1992, Gu, 1992] – better results than a complete, or even incomplete, systematic BT method.

Local Search

- start with a complete and randomly initialized assignment, then check whether it satisfies all clauses.
- if not, heuristically or randomly select a variable to flip (i.e., change its value).
- repeat this process until a solution is found.

A local search method contains three key concepts [Barták, 1998]:

1. Configuration: one possible assignment to all variables, not required to be a solution;
2. Evaluation value: the number of satisfied constraints (in a CSP) or clauses (in an SAT);
3. Neighbor: a configuration obtained by flipping the assignment of a variable in the current configuration.

Improved Local Search

There are two main streams

- GSAT [Gu, 1992, Selman et al., 1992] and WalkSAT [Selman et al., 1994].
- They all have many variants,
  - GWSAT [Selman et al., 1994], GSAT/Tabu [Mazure et al., 1997, Steinmann et al., 1997], HSAT [Gent and Walsh, 1993], and HWSAT [Gent and Walsh, 1995] following GSAT, and
  - WalkSAT/Tabu [McAllester et al., 1997], Novelty [McAllester et al., 1997], and R-Novelty [McAllester et al., 1997] following WalkSAT.

Self-Organization Based Methods

- Liu and Han developed an artificial life model for solving large-scale n-queen problems [Liu and Han, 2001].
- Other related examples of self-organizing systems: cellular automata [Gutowitz, 1991, Liu et al., 1997] and Swarm [Swarm, 1994].
- Cellular automata are dynamical systems that operate in discrete space and time. The state of the cell is locally specified according to a set of behavioral rules [Liu et al., 1997, Shanahan, 1994].
- Swarm is a system for simulating distributed multi-entity systems.

ERE vs. other Methods I

- Entities in ERE can synchronously behave according to their behavioral rules, whereas local search is sequential.
- Each entity follows its local behavioral rules, and as a result, the system gradually evolves towards a solution state.
- ERE may be regarded as an extended GT algorithm, somewhat like local search.
- ERE evaluates not the number of dissatisfied constraints for the whole assignment as in local search, but the number of dissatisfied constraints for every value combination.
- The values of evaluation are stored in the environment.
- In ERE, the neighbors of an assignment can be different in the values of more than one variable.

ERE vs. other Methods II

- Other methods: Yokoo et al. have proposed several algorithms (i.e., asynchronous backtracking, asynchronous weak-commitment search, and multi-agent real-time-A* algorithm with selection) for solving distributed CSPs [Yokoo, 1995, Yokoo et al., 1998, Yokoo and Hirayama, 1998, Yokoo and Hirayama, 2000, Yokoo and Kitamura, 1996]. Later, Silaghi et al. improved the work of Yokoo et al. by introducing a mechanism to check whether the message an agent receives is legal [Silaghi et al., 2001a, Silaghi et al., 2001b, Silaghi et al., 2001c].

Entity Network

- In multi-entity systems for problem solving, entities will implicitly or explicitly form an entity network that connects all interacting entities.
- As an entity may or may not cooperate, coordinate, or compete with other entities, the resulting entity network may not be fully connected.
- An **entity network** is a virtual graph corresponding to a multi-entity system, where vertices are entities, and edges (also called links) are the implicit or explicit relationships of cooperation, coordination, or competition among entities.
- **Issues:**
  - What is the topology of a network formed by entities?
  - How does the resulting entity network reflect the computational complexity of a given problem?

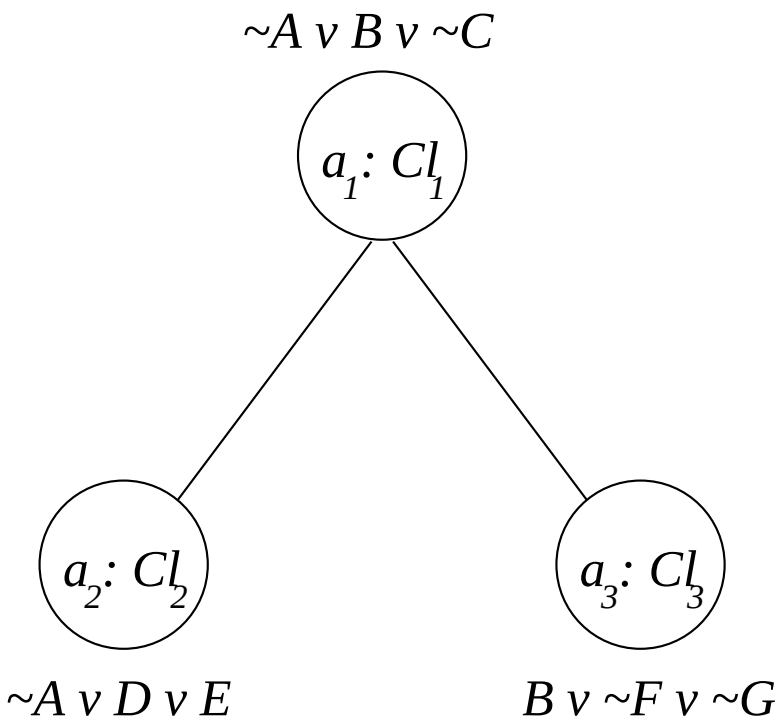
Entity Network Topology

- It is the geometrical property of the network that reflects the connectivity of the vertices and edges.
- **Different topologies based on different representations:**
  - constraint-based representation where entities represent constraints, and
  - variable-based representation where entities represent variables.
- ERE utilizes a variable-based representation.
- Which representation is better based on some benchmark SAT problems?

Clause Based Representation

- An entity represents a clause in a given SAT problem.
- To satisfy a clause, an entity needs to assign values to the variables in this clause such that at least one literal is true.
- Because a variable can appear in multiple clauses simultaneously, the entities that have common variables should cooperate and find consistent values to the variables.
- If two entities have a common variable, we regard it as an edge between the corresponding entity vertices.

Representing three clauses into an entity network, where each vertex denotes an entity, and each edge denotes a common variable shared by two corresponding entities. a_i:Cl_i denotes that entity a_i represents clause Cl_i.



Characterization [Watts and Strogatz, 1998]

Given an entity network, $G = \langle V, R \rangle$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a set of entities and $R = \{r_1, r_2, \cdots, r_m\}$ is a set of edges between entities in V.

1. Characteristic path length:

$$L_G = \frac{2}{n \cdot (n-1)} \sum_{i,j \in \{1,\cdots,n\}, i \neq j} d_{i,j}, \tag{34}$$

   where $d_{i,j}$ is the shortest distance between entities a_i and a_j.

2. Clustering coefficient:

$$C_G = \frac{1}{n} \sum_{i=1}^{n} c_{a_i}, \tag{35}$$

   where c_{a_i} is the clustering ratio (also called clustering in short) of entity a_i.

- Assume $d(a_i)$ is the degree of a_i (i.e., the number of neighboring entities of a_i), and $b(a_i)$ is the number of existing edges between the neighbors of a_i. Therefore,

$$c_{a_i} = \frac{b(a_i)}{\frac{(d(a_i)+1)\cdot d(a_i)}{2}} = \frac{2 \cdot b(a_i)}{(d(a_i)+1) \cdot d(a_i)}. \quad (36)$$

- In general, L_G is a global property of entity network G that indicates the connectivity of G.
- C_G is a local property that reflects the average connectivity of cliques in G. In essence, C_G denotes the possibility that two entities, which have a common neighboring entity, are neighbors.

Related Work on Small World Topology

- Milgram first proposed the notion of small world [Milgram, 1967].
- Later, Watts and Strogatz mathematically formulated a small world topology based on the means of characteristic path length and clustering coefficient [Watts and Strogatz, 1998].
- Small world phenomena have been extensively found in natural systems (e.g., human society [Milgram, 1967], food Web in ecology [Montoya and Sole, 2000]) as well as in man-made systems (e.g., the World Wide Web [Adamic, 1999]).
- Walsh observed such phenomena in search problems, such as graph coloring, time tabling, and quasigroup problems [Walsh, 1999]. He further experimentally showed that the small world topology could make a search process very difficult.

- Watts and Strogatz *qualitatively* defined that a graph G with n vertices and m edges has a small world topology if and only if:

$$L_G \approx L_{random} \quad \text{and} \quad C_G \gg C_{random}, \tag{37}$$

where L_{random} and C_{random} are the average characteristic path length and clustering coefficient of random graphs with the same size as G (i.e., n vertices and m edges). Here, G must be connected, i.e., $k \gg ln(n)$, where $k = \frac{2m}{n}$ is the average degree of vertices in G [Watts and Strogatz, 1998].

- Walsh provided a quantitative measurement, i.e., proximity ratio, μ [Walsh, 1999]:

$$\mu = \frac{\frac{C_G}{L_G}}{\frac{C_{random}}{L_{random}}} = \frac{C_G \cdot L_{random}}{C_{random} \cdot L_G}. \tag{38}$$

A small world topology requires $\mu \gg 1$. The larger the μ, the more "small worldy" the graph (i.e., the graph has more clusters).

Complexities under Different Representations

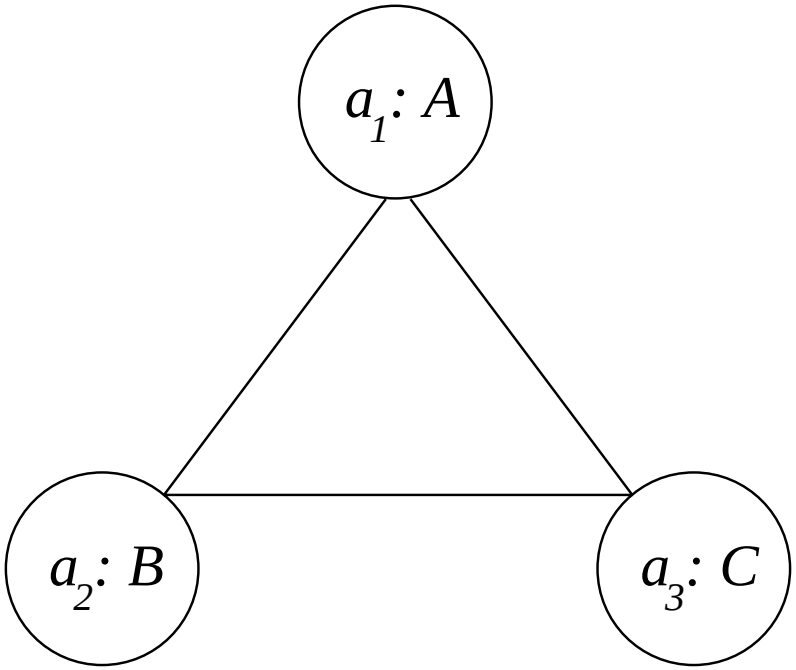
- In [Walsh, 1999], Walsh empirically verified that a small world topology increases the computational complexity of a search algorithm that involves certain heuristics.
- This is because heuristics normally guide a search process locally. But in a small world network, based on local information an algorithm cannot well predict global properties of the problem at hand.
- Using the same SAT problems, we have examined a clause-based representation as opposed to a variable-based representation as used in ERE.
- Our experimental results suggest that with such a clause-based representation, it is normally hard (in term of entity movements) to solve a problem, which is however relatively easier to solve by ERE with a variable-based representation.

Variable Based Representation

- In order to satisfy a clause, the related variables should be assigned compatible values to guarantee that at least one literal is true – clauses act as constraints among variables.
- ERE represents variable groups with entities; the constraints among variables are implicitly transformed into constraints among entities. To satisfy a constraint (i.e., a clause), entities that represent the variables in the constraint will restrain each other.
- If each entity represents only one variable: A vertex denotes an entity. An edge exists between two entities if and only if the corresponding variables appear in a certain clause simultaneously.

Representing a clause into an entity network, where each entity corresponds to one variable. a_i:X denotes that entity a_i represents variable X.

Clause: $\sim A \; v \; B \; v \sim C$



Variable-Based Representation for Benchmark SATs

Uniform-3-SAT and Flat Graph Coloring. In these experiments, we randomly selected 10% of the instances from each testset, and calculated average L_G, C_G, and μ of the selected instances. For each instance, we generated 10 random entity networks with the same number of vertices and edges.

Testset	Vertices	Edges	L_G	L_{random}	C_G	C_{random}	μ
Uf50	50	507	1.693	1.587	0.501	0.469	1.140
Uf75	75	831	1.758	1.702	0.403	0.360	1.154
Uf100	100	1,131	1.821	1.776	0.335	0.294	1.168
Uf125	125	1,457	1.862	1.822	0.297	0.255	1.190
Uf150	150	1,780	1.900	1.860	0.271	0.227	1.219
Uf175	175	2,095	1.939	1.895	0.250	0.207	1.238
Uf200	200	2,414	1.968	1.925	0.237	0.191	1.267
Uf225	225	2,727	2.001	1.957	0.223	0.179	1.276
Uf250	250	3,037	2.031	1.987	0.214	0.169	1.297
Flat30	90	270	3.662	2.677	0.387	0.334	1.585
Flat50	150	495	3.935	2.836	0.350	0.296	1.641
Flat75	225	765	4.186	3.025	0.337	0.279	1.671
Flat100	300	1,017	4.375	3.179	0.330	0.274	1.659
Flat125	375	1,278	4.532	3.290	0.332	0.269	1.697
Flat150	450	1,530	4.647	3.390	0.327	0.268	1.676
Flat175	525	1,776	4.735	3.479	0.329	0.267	1.674
Flat200	600	2,037	4.849	3.539	0.325	0.265	1.681

Observation: Non-Small-World Topology I

- For all testsets of Uniform-3-SAT, $L_G \approx L_{random}$, $C_G \approx C_{random}$, and $1.1 < \mu < 1.3$.
- For all testsets of Flat Graph Coloring, $L_G \not\approx L_{random}$, $C_G \not\approx C_{random}$, and μ is around 1.65.
- Thus, we can conjecture that with a variable-based representation, the resulting network does not have a small world topology.
- **If an entity represents several variables:** Two entities will be connected by an edge if and only if two variables, respectively represented by these two entities, appear in an identical clause.
- The resulting entity network can be derived from the entity network where each entity represents one variable by merging some of the original entities into a single one.

Observation: Non-Small-World Topology II

- The former network is denser than the latter one. Since the latter entity network does not have a small world topology, the former will not either (A small world topology normally exists in a connected, but 'sparse', network). This has been demonstrated in other experiments, where μ is still less than 2.0.

Balanced Complexities in Intra- and Inter-Entity Computations I

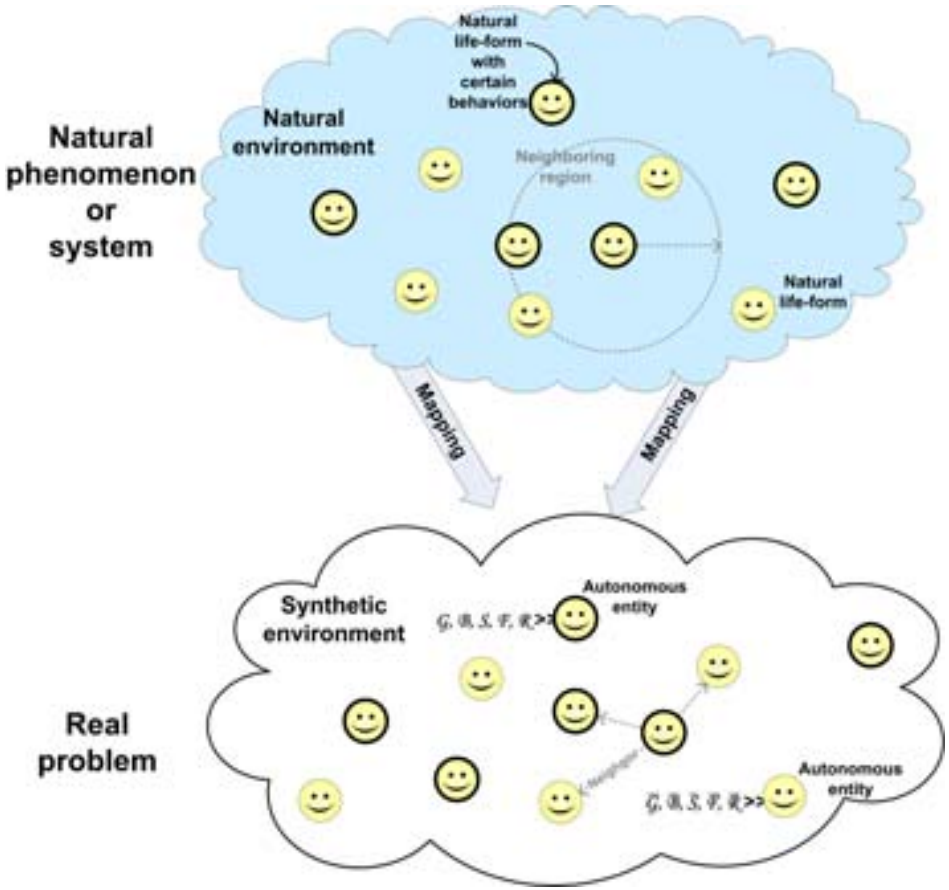
- Experiments have suggested that as the number of variables represented by an entity in ERE increases, the resulting entity networks will be less ‘small worldy’, because the networks become smaller and denser.
- In an extreme case, the network becomes to an isolated vertex. Is this the best situation?
- If an entity represents multiple variables, the entity should assign values to all its variables. Because the variables of an entity are usually not independent, as the number of variables represented by the entity increases, the intra-entity computational complexity to assign values to its variables increases, too.

Balanced Complexities in Intra- and Inter-Entity Computations II

- **A good design should balance the intra- and inter-entity computational complexities to achieve the lowest total computational cost.**
- Experiments have suggested that when an entity represents four or five variables, the total computational cost, in terms of entity movements and variable flips, becomes the lowest.

Remarks on AOC by Fabrication

The *AOC-by-fabrication* approach is intended to build a mapping between a real problem and a natural phenomenon or system.



Common Characteristics of AOC by Fabrication I

1. Each of autonomous entities is characterized by sets of goals, states, behaviors, and behavioral rules (e.g., $\mathcal{G}$, $\mathcal{B}$, $\mathcal{S}$, $\mathcal{F}$, and $\mathcal{R}$).
2. Entities may be homogeneous or heterogeneous. Even in the homogeneous case, entities may differ in some detailed parameters.
3. The composition of the entity group *may* change over time, through the process analogous to birth (amplification of the desired behavior) and death (elimination of the undesired behavior).
4. The interactions between autonomous entities are local; neither global information nor central executive control is needed.

Common Characteristics of AOC by Fabrication II

5. The environment is dynamical and records the information related to the current status of the problem. It serves as a medium for information sharing.
6. The local goals of autonomous entities drive the selection of their primitive behaviors at each step.
7. The goal of the whole AOC system is represented by a universal fitness function that measures the progress of the computation.

Web Regularities

- Self-organized regularities from the World Wide Web, ranging from the structure and growth of the Web to the access patterns in Web surfing.
- Many identified interesting regularities are best represented by characteristic distributions following either a Zipf-like law [Zipf, 1949] or a power law. That is, the probability P of a variant taking value k is proportional to $k^{-\alpha}$, where α is from 0 to 2. A distribution presents a heavy tail if its upper tail declines like a power law [Crovella and Taqqu, 1999].

The Empirical Regularities on the Web I

Zipf-like or power-law distributions are found in:

1. The popularity of requested and transferred pages across servers and proxy caches [Barford et al., 1999, Breslau et al., 1998, Cuhna et al., 1995, Glassman, 1994].
2. The popularity of websites or requests to servers, ranging from Web user groups to fixed user communities, such as within a proxy or a server [Adamic and Huberman, 2000, Breslau et al., 1998, Maurer and Huberman, 2000].
3. The request inter-arrivals and Web latencies [Barford and Crovella, 1998, Helbing et al., 2000, Yan et al., 1996].
4. The distribution of document sizes either across the Web or limited to pages requested in a proxy or a certain user community [Arlitt and Williamson, 1996, Barford et al., 1999, Barford and Crovella, 1998, Cuhna et al., 1995].

The Empirical Regularities on the Web II

5. The number of pages either across all websites or within a certain domain of the Web [Huberman and Adamic, 1999].
6. The trace length of users within a proxy or a website, or across the Web [Adar and Huberman, 2000, Huberman et al., 1997, Levene et al., 2001, Lukose and Huberman, 1998].
7. The dynamical response of the Web to a Dirac-like perturbation [Johansen and Sornette, 2000].
8. The distribution of links (both incoming and outgoing) among websites or pages [Adamic and Huberman, 1999, Albert et al., 1999, Barabasi and Albert, 1999, Barabasi et al., 2000, Broder et al., 2000].

Web Regularity Characterization I

- Viewing the Web as a large directed graph of nodes (i.e., Web pages) connected with links (i.e., hyperlinks), Huberman et al. proposed a random-walk model to simulate certain regularities in user navigation behavior and suggested that the probability distribution of surfing depth (steps) follows a two-parameter inverse Gaussian distribution [Huberman et al., 1997].
- They conjectured that the probability of finding a group surfing at a given level scales inversely in proportion to its depth, i.e., $P(L) \sim L^{-3/2}$, where L is depth.

Web Regularity Characterization II

- The random-walk model
  [Huberman et al., 1997, Lukose and Huberman, 1998] and the Markov chain model
  [Levene et al., 2001, Levene and Loizou, 1999] do not relate the emergent regularities to the dynamical interactions between users and the Web, nor do they reflect the inter-relationships between user behavior and the contents or structure of the Web – black-box methods.

Objectives

- How to explain user behavior underlying observed Web usage regularities?
- By experimenting with the entity-based decision model of Web surfing, we aim to further characterize user navigation regularities as well as to understand the effects of user interests, motivation, and content organization on user behavior.
- The information foraging entity based model that takes into account the interest profiles, motivation aggregation, and navigation strategies of users.

Specific Issues I

1. Is it possible to experimentally observe regularities similar to empirical Web regularities if we formulate the aggregation of user motivation? In other words, is it possible to account for empirical regularities from the point of view of motivation aggregation?
2. Are there any navigation strategies or decision making processes involved that determine the emergence of Web regularities, such as the distributions of user navigation depth?
3. If the above is validated, will different navigation strategies or decision making processes lead to different emergent regularities? In other words, when we observe different power-law distributions, can we tell what are dominant underlying navigation strategies or decision making processes that have been used by users?

Specific Issues II

4. What is the distribution of user interest profiles underlying emergent regularities?
5. Will the distribution of Web contents as well as page structure affect emergent regularities?
6. If we separately record users who can successfully find relevant information and those who fail to do so, will we observe different regularities?

Artificial Web Space

- The Web space is a collection of websites connected by hyperlinks.
- Each website contains certain information contents, and each hyperlink between two websites signifies certain content similarity between them.
- The contents contained in a website can be characterized using a multi-dimensional content vector where each component corresponds to the relative information weight on a certain topic.
- To build the artificial Web space that characterizes the topologies as well as connectivities of the real-world Web, we introduce the notion of an artificial website that may cover contents related to several topics and each topic may include a certain number of Web pages.
- Such a website may also be linked to other websites of similar or different topics through URLs.

Web Space and Content Vector Representations

- We consider the Web space as a graph consisting of nodes and links, as suggested in [Broder et al., 2000].
- The nodes correspond to websites or pages, whereas the links correspond to hyperlinks between them.
- The information contents in a certain node are represented using the weights of a content vector as follows:

$$\mathbb{C}_n = [cw_n^1, cw_n^2, \ldots, cw_n^i, \ldots, cw_n^{\mathbb{M}}], \tag{39}$$

where

$\mathbb{C}_n$: content vector for node n (i.e., website or page);
cw_n^i: relative content information weight on topic i;
$\mathbb{M}$: number of topics.

- To determine the content similarity between two nodes, we will make use of the following distance function:

$$d(\mathrm{C}_i, \mathrm{C}_j) = \left(\sum_{k=1}^{\mathrm{M}} (cw_i^k - cw_j^k)^2 \right)^{1/2}, \tag{40}$$

  where $d(\mathrm{C}_i, \mathrm{C}_j)$ denotes the Euclidean distance between the content vectors of nodes i and j.
- When two nodes are linked through a hyperlink, it is reasonable to assume that the contents contained in the two nodes is somewhat related, that is to say, their content vector distance is below a certain positive threshold.

Modeling Content Distributions

- **Normal distribution:** The content weight cw_n^i with respect to topic j in node n is initialized as follows:

$$cw_n^i = \begin{cases} \mathtt{T} + \mid \mathtt{X}_c \mid, & \text{if } i = j, \\ \mid \mathtt{X}_c \mid, & \text{otherwise}, \end{cases} \tag{41}$$

$$f_{\mathtt{X}_c} \sim normal(0, \sigma_p), \tag{42}$$

$$\mathtt{T} \sim normal(\mu_t, \sigma_t), \tag{43}$$

$f_{\mathtt{X}_c}$: probability distribution of weight $\mathtt{X}_c$;
$normal(0, \sigma_p)$: normal distribution with mean 0 and variance σ
$\mathtt{T}$: content (increment) offset on a topic;
μ_t: mean of normally distributed offset $\mathtt{T}$;
σ_t: variance of normally distributed offset $\mathtt{T}$.

- All content weights on a topic are nonnegative.
- We can adjust σ_t and μ_t to get various topic distributions in Web pages; the smaller σ_t is or the larger μ_t is, the more focused the node will be on the topic.

Modeling Content Distributions

- **Power-law distribution:** The content weight of node n on topic j, cw_n^j, will follow a power law:

$$cw_n^j = \begin{cases} \mathtt{T} + \mid \mathtt{x}_c \mid, & \text{if } i = j, \\ \mid \mathtt{x}_c \mid, & \text{otherwise}, \end{cases} \tag{44}$$

$$f_{\mathtt{x}_c} \sim \alpha_p (\mathtt{x}_c + 1)^{-(\alpha_p + 1)}, \ \mathtt{x}_c > 0, \ \alpha_p > 0, \tag{45}$$

$f_{\mathtt{x}_c}$: probability distribution of weight $\mathtt{x}_c$;
α_p: shape parameter of a power-law distribution;
$\mathtt{T}$: content (increment) offset on a topic.

- We can adjust α_p to generate different forms of a power-law distribution.

- How to construct an artificial Web?
  1. **Create groups of nodes** where each group focuses on a certain topic. The distribution of the contents in the nodes follows a specific model as given above. An information entity starts its foraging from a Web homepage that contains links to the nodes of several topics. We assign this homepage an equal distance to individual topics.
  2. **Build links between the nodes** We build a link between two nodes only if the content vector distance between them is below a positive distance threshold, r.
  3. Increasing r (degree-of-coupling of websites) leads to increasing the number of links in a website (i.e., different degrees of connectivity).
- Assumption: When there is a link between two nodes, the information contents of the nodes should be related.

Algorithm for Constructing the Artificial Web Space

for each topic k **do**
  Create a node group and content vectors;
end for
for each node i in the group **do**
  Initialize the link list of node i;
  for each node j ($j \neq i$) in the group **do**
    if $d(\mathrm{c}_i, \mathrm{c}_j) < r$ **then**
      Add node j to the link list of node i;
      Add $d(\mathrm{c}_i, \mathrm{c}_j)$ to the link list of node i;
    end if
  end for
end for

Foraging Entities

- Each entity forages in the Web space with **different interests** in mind.
- The interest profile of an entity will determine its behavior in Web surfing.
- How to model the interest profile of an entity using a multi-dimensional preference vector that specifies the interests of the entity in various topics?
- How to define the interest entropy that characterizes whether or not an entity has a balanced interest profile?

Interest Profiles

We define the preference vector of an entity as follows:

$$\mathtt{P}_m = [pw_m^1, pw_m^2, \ldots, pw_m^i, \ldots, pw_m^{\mathtt{M}}], \tag{46}$$

$$p_{mi} = \frac{pw_m^i}{\sum_{j=1}^{\mathtt{M}} pw_m^j}, \tag{47}$$

$$\mathtt{H}_m = -\sum_{i=1}^{\mathtt{M}} p_{mi} log(p_{mi}), \tag{48}$$

$\mathtt{P}_m$: preference vector of entity m;
pw_m^i: preference weight of entity m on topic i;
$\mathtt{H}_m$: interest entropy of entity m.

- H_m indicates the breadth and balance of an entity's interests in different topics.
- The larger H_m is, the more evenly distributed the entity's interests will be. As a result, the entity is more likely to have multiple goals and jump from one topic to another in its surfing.
- When the entity has equal interests in all topics, the value of H_m will be the largest, i.e.,

$$\mathrm{H}_{max} = -\sum_{i=1}^{\mathrm{M}} \frac{1}{\mathrm{M}} log\left(\frac{1}{\mathrm{M}}\right) = log(\mathrm{M}). \tag{49}$$

- The quantity of interest entropy will affect the decision of an entity on which Web page it will select among several others.

Modeling Interest Distributions

1. **Normal distribution:** The weight of a preference vector, pw_m^i, for entity m on topic i is defined as follows:

$$pw_m^i = \mathrm{X}_p, \tag{50}$$

$$f_{\mathrm{X}_p} \sim normal(0, \sigma_u), \tag{51}$$

where $normal(0, \sigma_u)$ denotes the normal distribution with mean 0 and variance σ_u.

2. **Power-law distribution:** The probability distribution pw_m^i:

$$pw_m^i = \mathrm{X}_p, \tag{52}$$

$$f_{\mathrm{X}_p} \sim \alpha_u(\mathrm{X}_p + 1)^{-\alpha_u+1},\ \mathrm{X}_p > 0,\ \alpha_u > 0, \tag{53}$$

where α_u denotes the shape parameter of a power-law distribution.

Motivational Support

- S_t serves as the driving force for an entity to forage further. When an entity has found some useful information, it will get rewarded, and thus the support value will be increased.
- As the support value exceeds a certain threshold, which implies that the entity has obtained a sufficient amount of useful information, the entity will stop foraging.
- If the support value is too low, the entity will lose its motivation to forage further and thus leave the Web space.

Support Aggregation

$$\mathtt{S}_{t+1} = \mathtt{S}_t + \theta \cdot \Delta\mathtt{M}_t + \phi \cdot \Delta\mathtt{R}_t, \tag{54}$$

$\mathtt{S}_t$: support value at step t;
$\Delta\mathtt{M}_t$: motivational loss at step t;
$\Delta\mathtt{R}_t$: reward received at step t;
θ, ϕ: coefficients of motivation and reward terms, respectively.

$$rcl\,init\_support_m = \frac{1}{2}\sum_{i=1}^{\mathtt{M}} pw_m^i, \tag{55}$$

$$max\_support_m = \sum_{i=1}^{\mathtt{M}} pw_m^i, \tag{56}$$

$$min\_support_m = 0, \tag{57}$$

where pw_m^i denotes the preference weight of entity m with respect to topic i.

Random Entities

- Random entities have no strong interests in any specific topics. They wander from one page to another.
- The probability, p_k, of reaching node k at the next step:

$$p_k = \frac{1}{h},\ k = 1, \ldots, h. \tag{58}$$

Rational Entities

- When rational entities reach a new website, they will try to decide whether or not the content sufficiently meets their interests and, if not, predict the next-level page based on the anchor texts.
- How to compute the probability, p_k, of reaching the next-level node k given the interest entropy, $\mathtt{H}_m$, of entity m?

$$d^*(\mathtt{P}_m, \mathtt{C}_k) = \begin{cases} d(\mathtt{P}_m, \mathtt{C}_k), & \text{if } k \in h_1, \\ \frac{\mathtt{H}_m}{\mathtt{H}_{max}} d(\mathtt{P}_m, \mathtt{C}_k), & \text{if } k \in h_2, \end{cases} \tag{59}$$

where $d^*(\mathtt{P}_m, \mathtt{C}_k)$ denotes the weighted distance between the preferences of entity m and the contents of node k given the entity's interest entropy $\mathtt{H}_m$.

$$\mathbb{Q}_j = d^*(\mathbb{P}_m, \mathbb{C}_j) - mean_{\forall l \in [1,h]}(d^*(\mathbb{P}_m, \mathbb{C}_l)), \quad j = 1, \ldots, h, \tag{60}$$

$$\mathbb{U}_j = \begin{cases} \mathbb{Q}_j, & \text{if } \mathbb{Q}_j < 0, \\ 0, & \text{if } \mathbb{Q}_j \geq 0, \end{cases} \tag{61}$$

$$p_k = \frac{\mathbb{U}_k}{\sum_{j=1}^{h} \mathbb{U}_j}, \tag{62}$$

Recurrent Entities

- Recurrent entities are familiar with the Web structure. Each time when they decide to forage further, they know exactly the whereabouts of the pages that closely match their interest profiles.
- The probability of reaching node k at the next step:

$$p_k = \begin{cases} 1, & \text{if } d^*(\mathtt{P}_m, \mathtt{C}_k) = min(d^*(\mathtt{P}_m, \mathtt{C}_j)),\ j = 1, \ldots, h, \\ 0, & \text{otherwise.} \end{cases} \tag{63}$$

Preference Updating

- An entity updates its preference weights in the interest profile over time, depending on how much information on interesting topics it has found and how much it has absorbed such information.
- When entity *m* reaches and finishes reading page *n*, it will update its interest according to the content vector of page *n*:

$$\mathtt{P}_m(\tau) = \mathtt{P}_m(\tau - 1) - \lambda \cdot \mathtt{C}_n, \tag{64}$$

$$pw_m^i(\tau) = 0, \text{ for } pw_m^i(\tau) < 0, \quad i = 1, \ldots, \mathtt{M}, \tag{65}$$

where λ denotes an absorbing factor in [0,1] that implies how much information is accepted by an entity on average. $\mathtt{P}_m(\tau)$ and $\mathtt{P}_m(\tau - 1)$ denote the preference vectors of an entity after and before accessing information on page *n*, respectively.

Motivation Function

$\Delta \mathrm{M}_t$ (motivational or patience loss) changes along with the latency, i.e., the time to find information.

$$\Delta \mathrm{M}_t = -(\Delta \mathrm{M}_t^c + \Delta \mathrm{M}_t^v), \tag{66}$$

where $\Delta \mathrm{M}_t^c$ denotes the constant decrement in $\Delta \mathrm{M}_t$ at each step, and $\Delta \mathrm{M}_t^v$ denotes the variable factor that dynamically changes at each step.

$\Delta \mathbb{M}_t^v$ Models

1. A log-normal function:

$$f_{log(\Delta \mathbb{M}_t^v)} \sim normal(\mu_m, \sigma_m), \tag{67}$$

where μ_m and σ_m denote the mean and variance of the log-normal distribution of $\Delta \mathbb{M}_t^v$, respectively.

2. An exponential function:

$$\Delta \mathbb{M}_t^v = \alpha_m e^{\gamma_m \cdot step}, \tag{68}$$

where α_m and γ_m denote the coefficient and rate of an exponential function, respectively. *step* denotes the number of pages or nodes that an entity has continuously visited.

Reward Function

- $\Delta \mathrm{R}_t$ corresponds to the reward received by an entity at each step as a function proportional to the relevant information that the entity has absorbed.
- The change in the preference weights of an entity reflects the information that the entity has gained:

$$\Delta \mathrm{R}_t = \sum_{i=1}^{\mathrm{M}} (pw_m^i(\tau - 1) - pw_m^i(\tau)). \tag{69}$$

- $\Delta \mathrm{R}_t$ (≥ 0) provides the entity with the energy to forage on the Web. $\Delta \mathrm{M}_t$ (≤ 0) prevents the entity to forage further. The total support for an entity at the current step can be aggregated based on the support received at the previous steps and the changes in motivational loss and reward.

The Foraging Algorithm I

Initialize nodes and links in artificial Web space;
Initialize information foraging entities and their interest profiles;
for each entity m **do**
  while support $s < max\_support_m$ and $s > min\_support_m$ **do**
    Find hyperlinks inside node n where entity m is visiting;
    Select, based on p_k, a hyperlink connected to a next-level page;
    Forage to the selected page;
    Update preference weights in the entity's interest profile;
    Update the support function of entity m;
  end while
  if support $s > max\_support_m$ **then**
    Entity m is satisfied with the contents and leaves the Web space;
  else

The Foraging Algorithm II

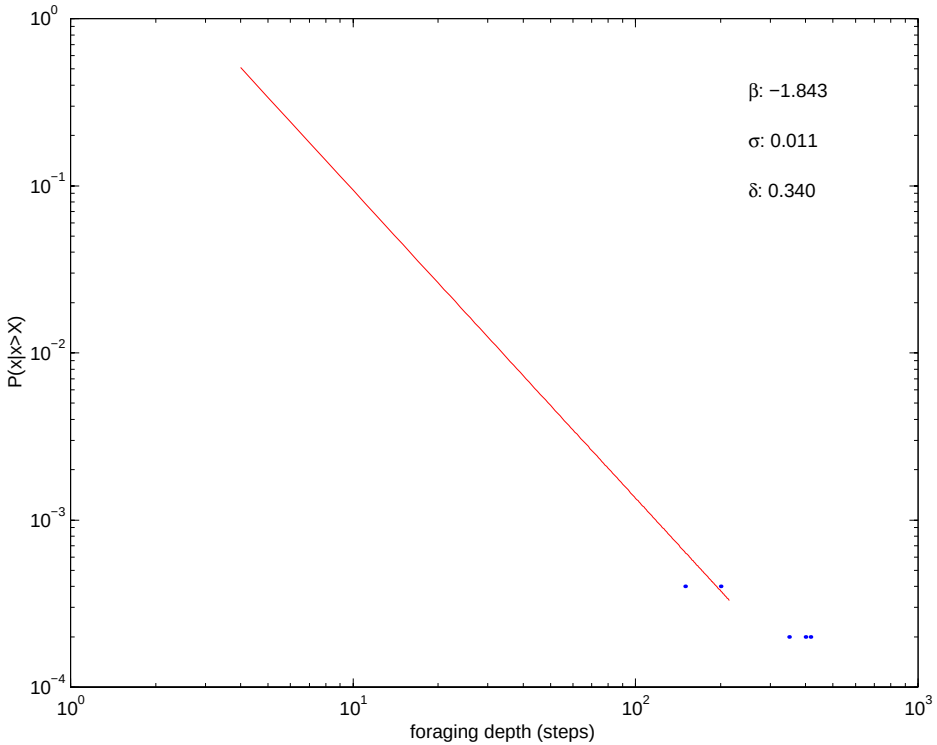
Entity *m* is dissatisfied and leaves the Web space;
end if
end for

- $5,000$ entities foraging according to the above given motivational support and decision models for three categories of foraging entities.
- We assume that
  - the **interest profiles** of the entities follow a **power-law** distribution and
  - the **contents** of Web pages on various topics follow a **normal-like** distribution.

  We are interested in studying the distributions of entity foraging depth and link-click-frequency.

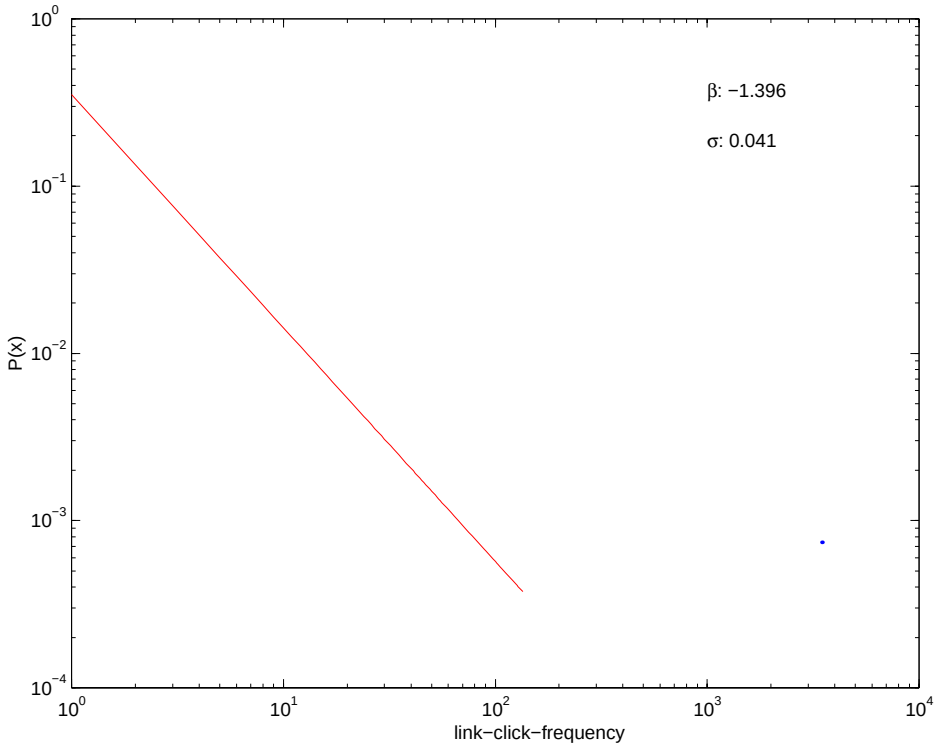
Cumulative Distribution of Foraging Depth (Steps)

Recurrent entities: The tail of the distribution follows a power-law distribution with power $\beta_c = -1.843$ and the residual of linear regression $\sigma = 0.011$. δ denotes entities' satisfaction rate (i.e., the ratio of the number of satisfied entities to the total number of entities what have surfed on the Web).



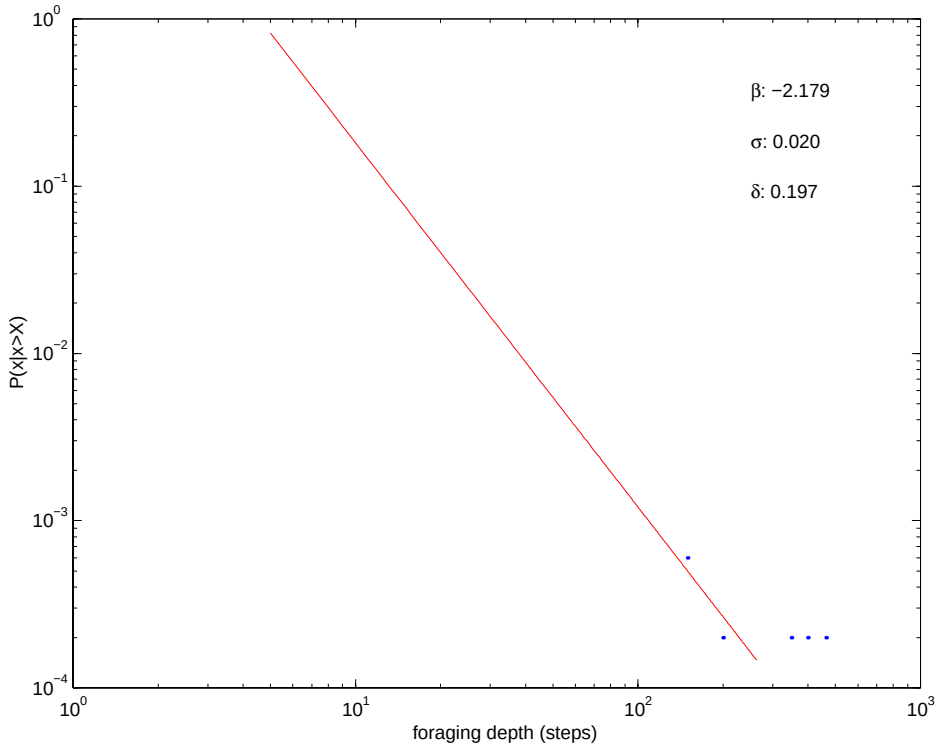
Distribution of Link-Click-Frequency

Recurrent entities: The tail follows a power-law distribution with power $\beta_l = -1.396$.



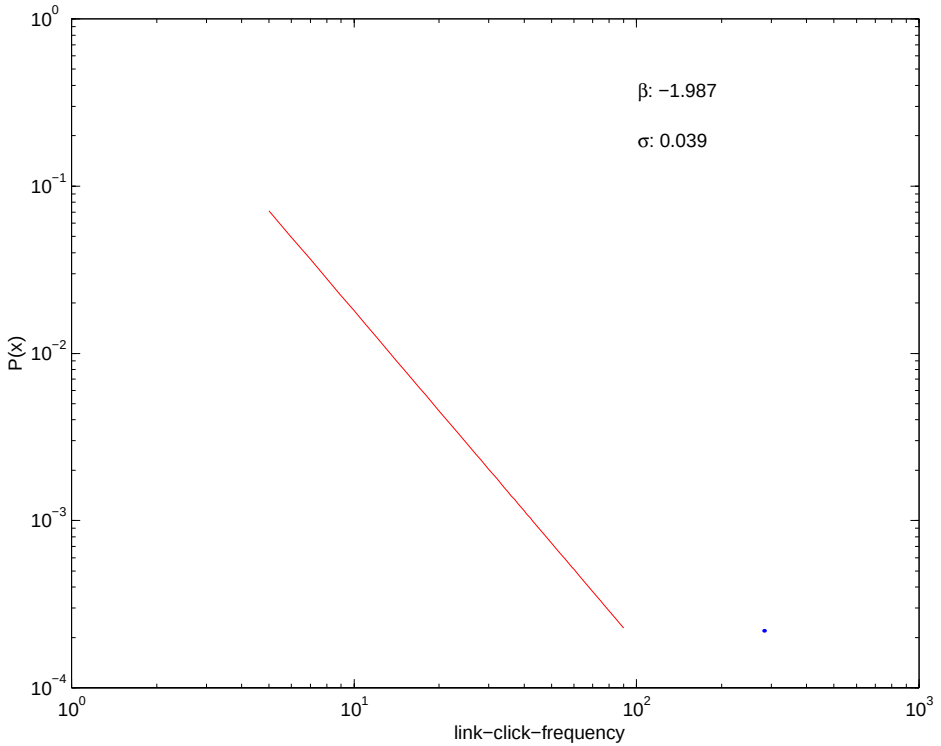
Cumulative Distribution of Foraging Depth (Steps)

Rational entities: The tail of the distribution follows a power-law distribution with power $\beta_c = -2.179$ and the regression residual $\sigma = 0.02$. δ denotes entities' satisfaction rate.



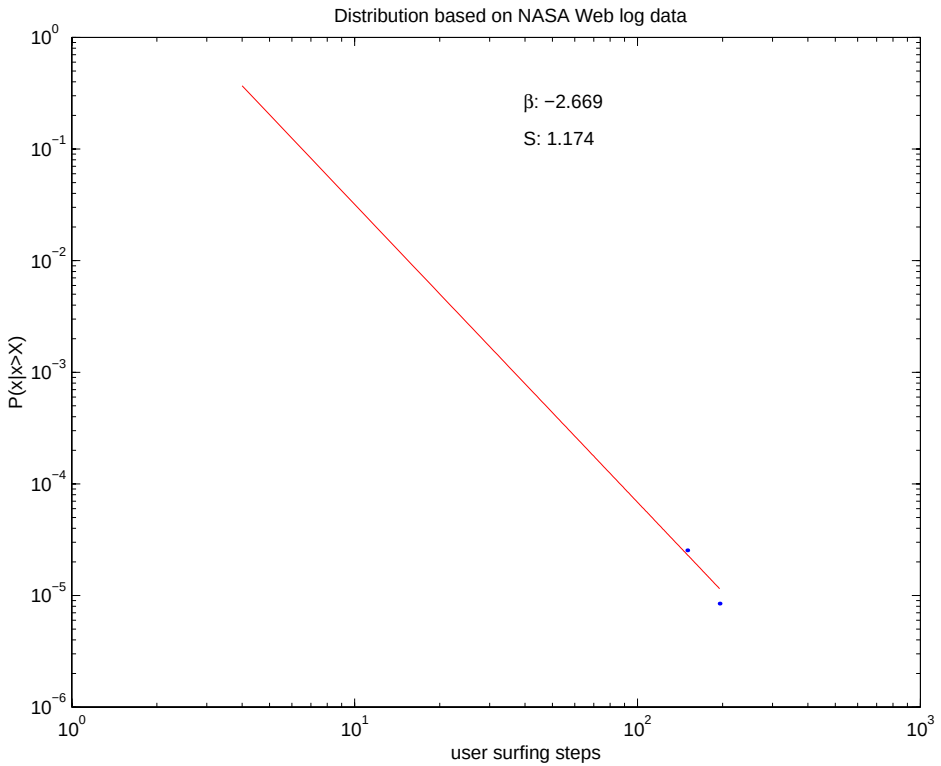
Distribution of Link-Click-Frequency

Rational entities: The distribution follows a power-law distribution with power $\beta_l = -1.987$.



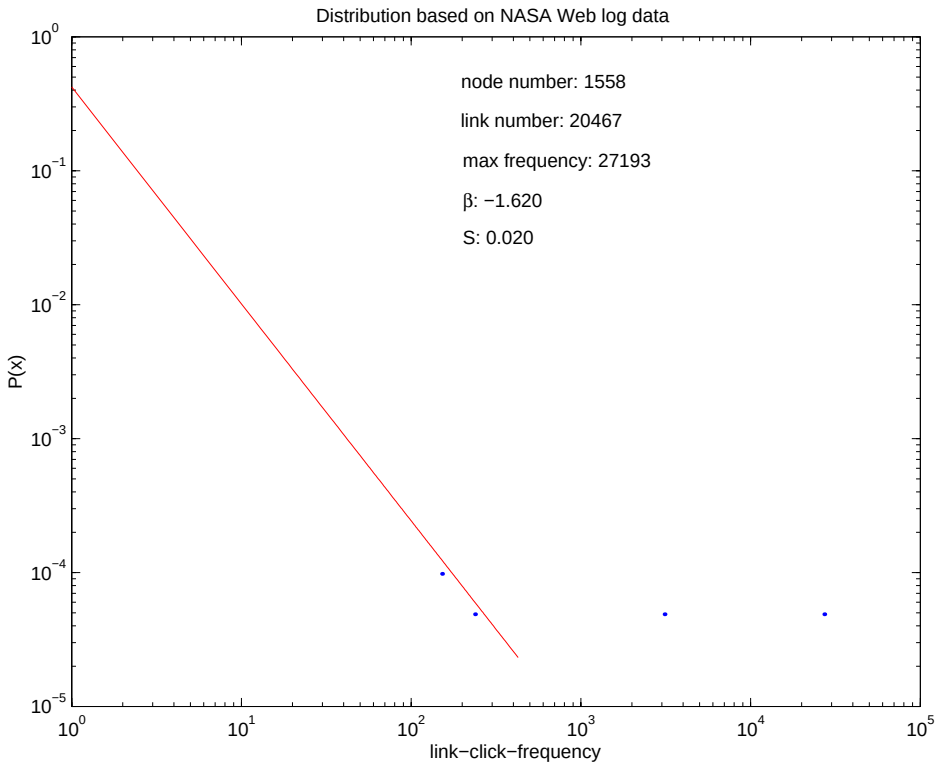
Cumulative Distribution of User Surfing Steps – NASA Web Log Data

The distribution follows a heavy tail with the tail's scale of $\beta_c = -2.669$. The linear regression residual s is about 1.174.



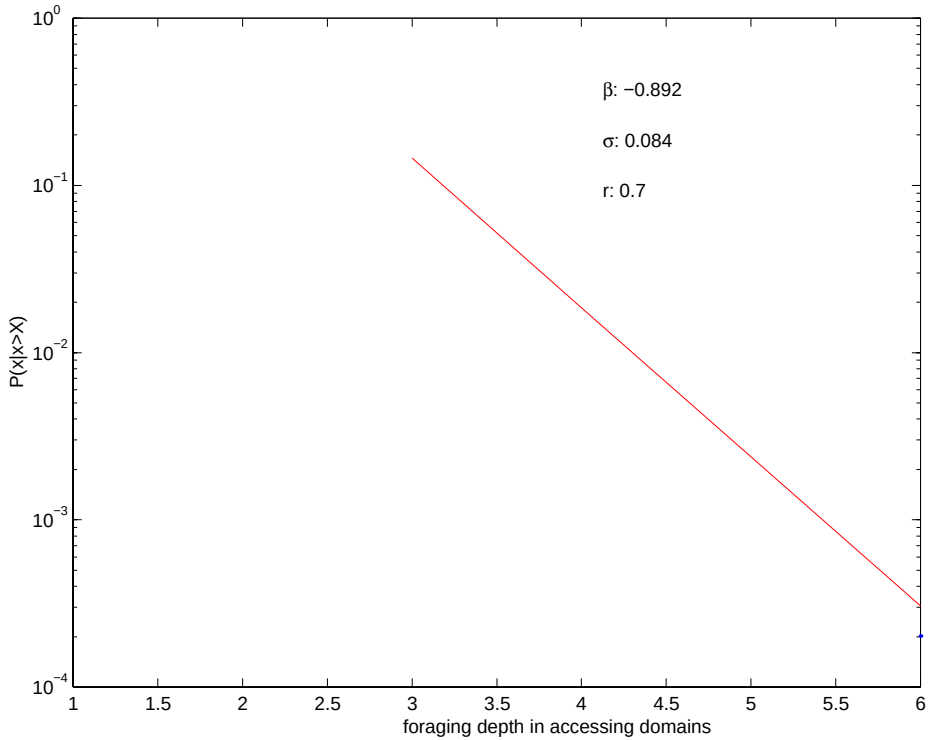
Link-Click-Frequency – NASA Web Log Data

It agrees well with a power law of power $\beta_l = -1.620$.



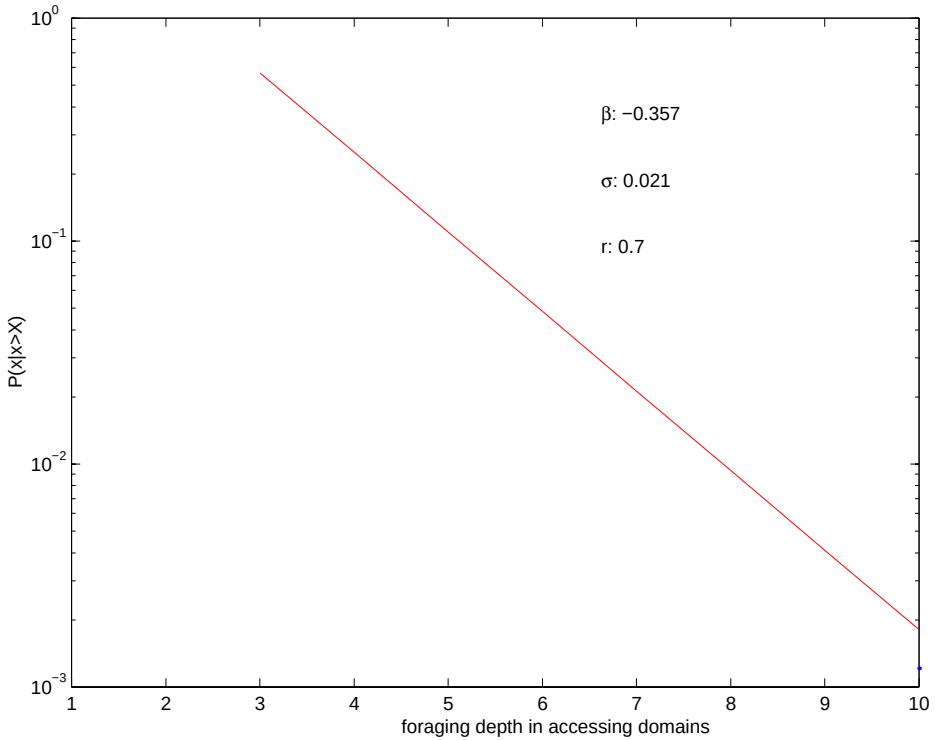
Cumulative Distribution of Foraging Depth in Accessing *Domains*

Recurrent entities: The distribution follows an exponential function with exponent $\beta_d = -0.892$ and residual $\sigma = 0.084$.



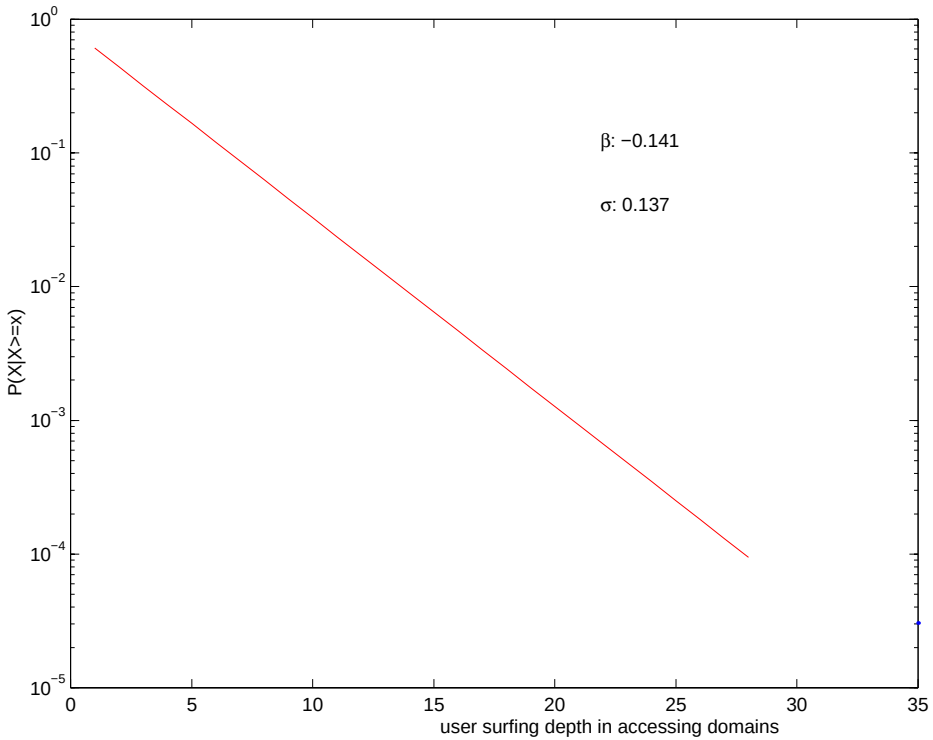
Cumulative Distribution of Foraging Depth in Accessing *Domains*

Rational entities: The distribution follows an exponential function with a smaller exponent $\beta_d = -0.357$ and residual $\sigma = 0.021$.



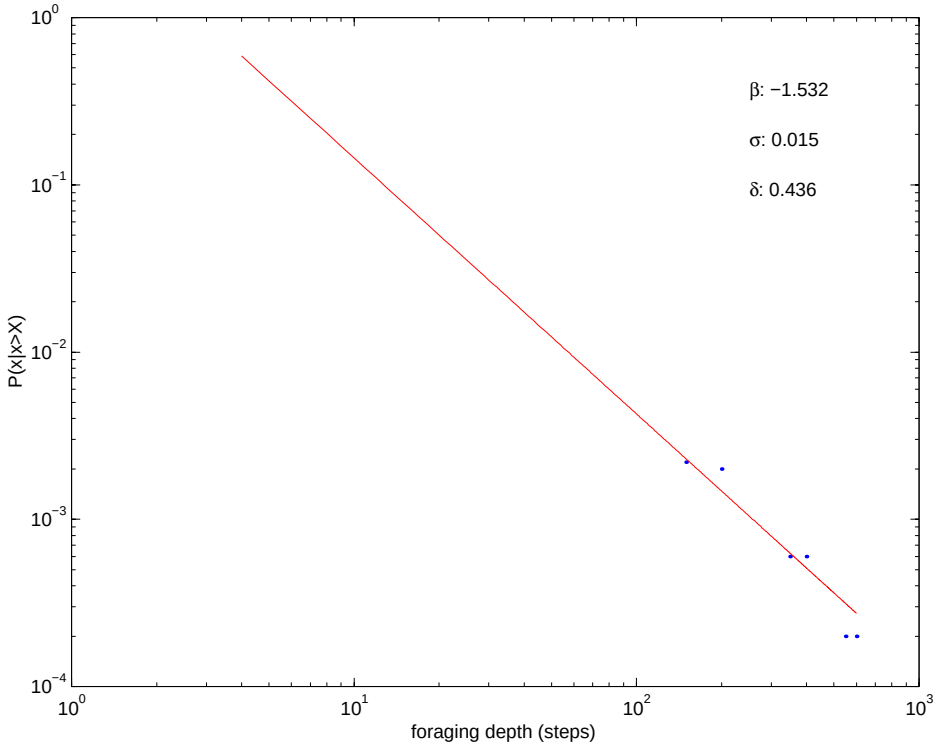
Surfing Steps in Accessing *Domains* – Microsoft Web Log Data

The distribution follows an exponential function with $\beta_d = -0.141$ and residual $\sigma = 0.137$.

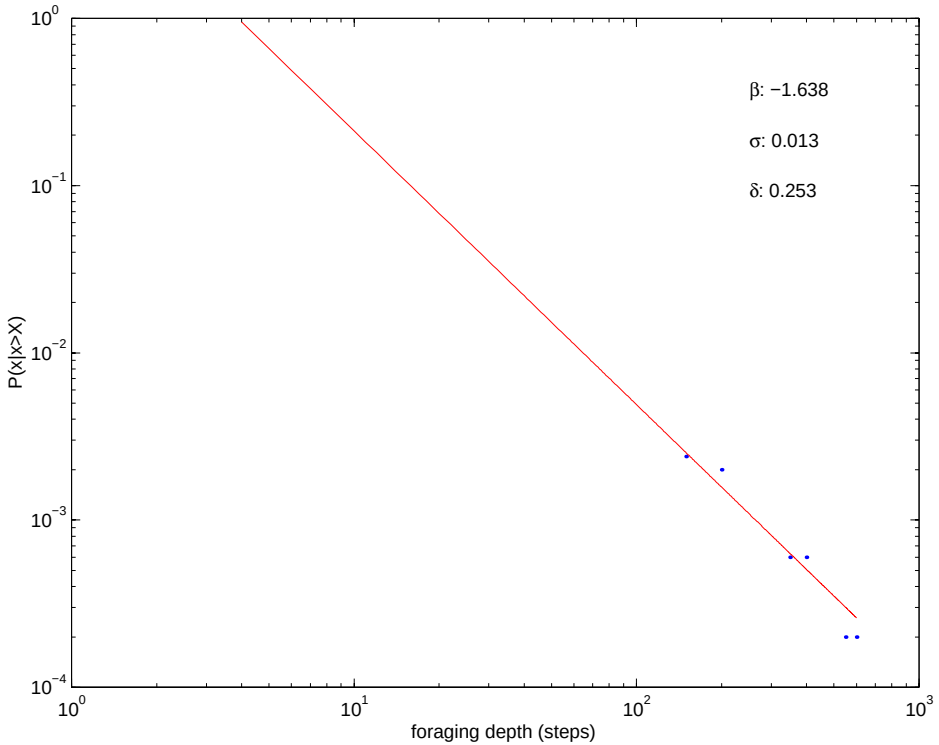


Influence of Different Content Distribution Models

Recurrent entity foraging depth in accessing Web pages where the content distribution follows a *power law*. The obtained distribution follows a power law with power $\beta_c = -1.532$ and residual $\sigma = 0.015$.

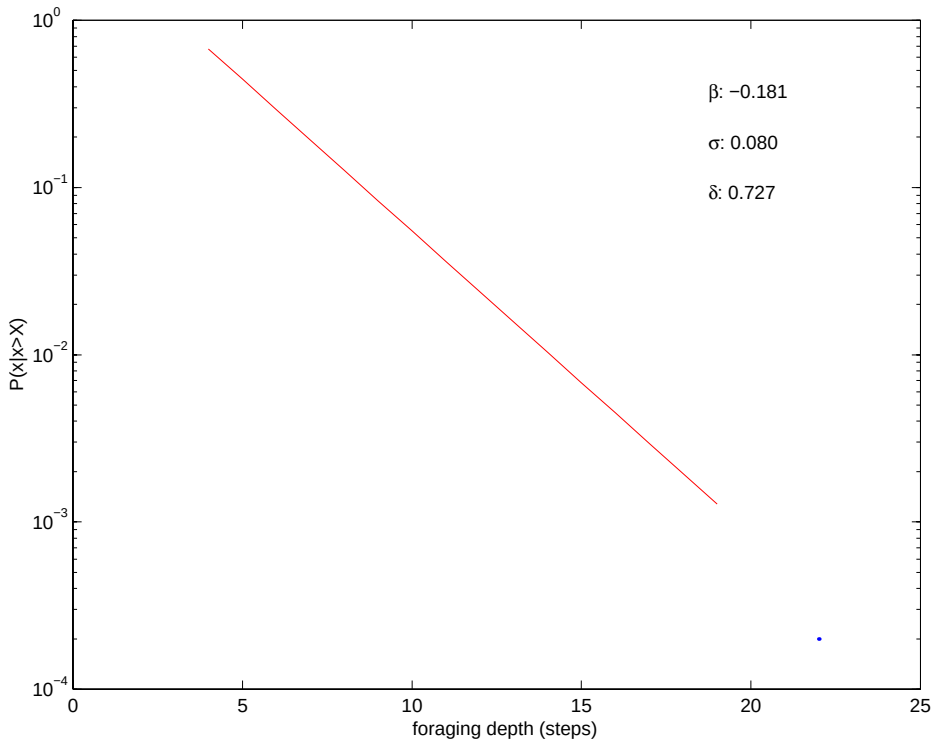


Rational entities: The distribution follows a power law with power $\beta = -1.638$ and residual $\sigma = 0.013$.

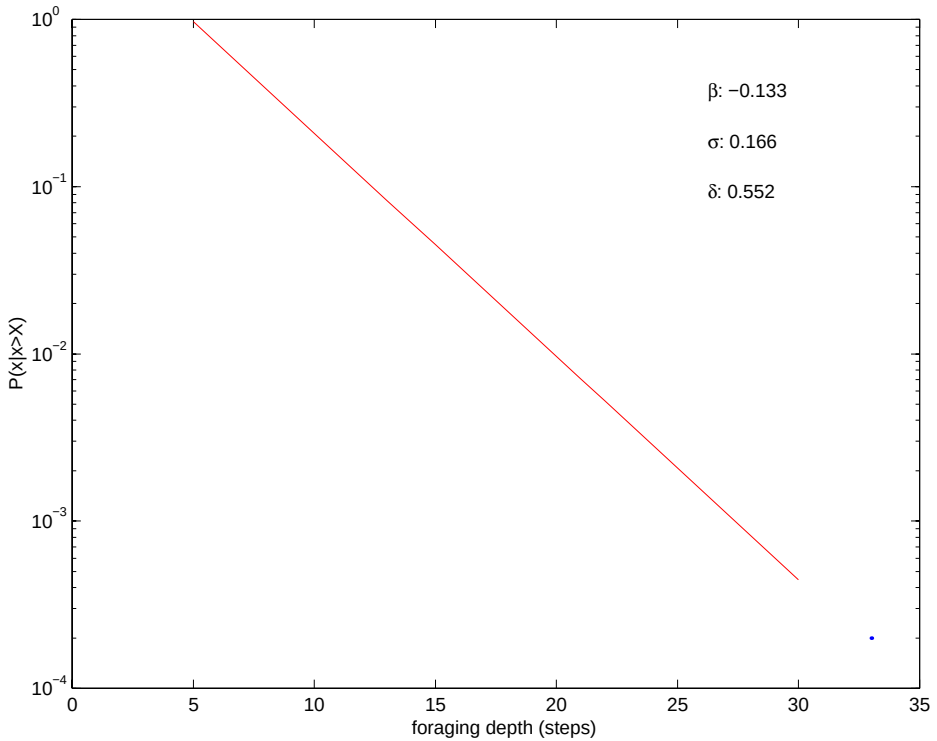


Interest Profiles Following a Normal-Distribution

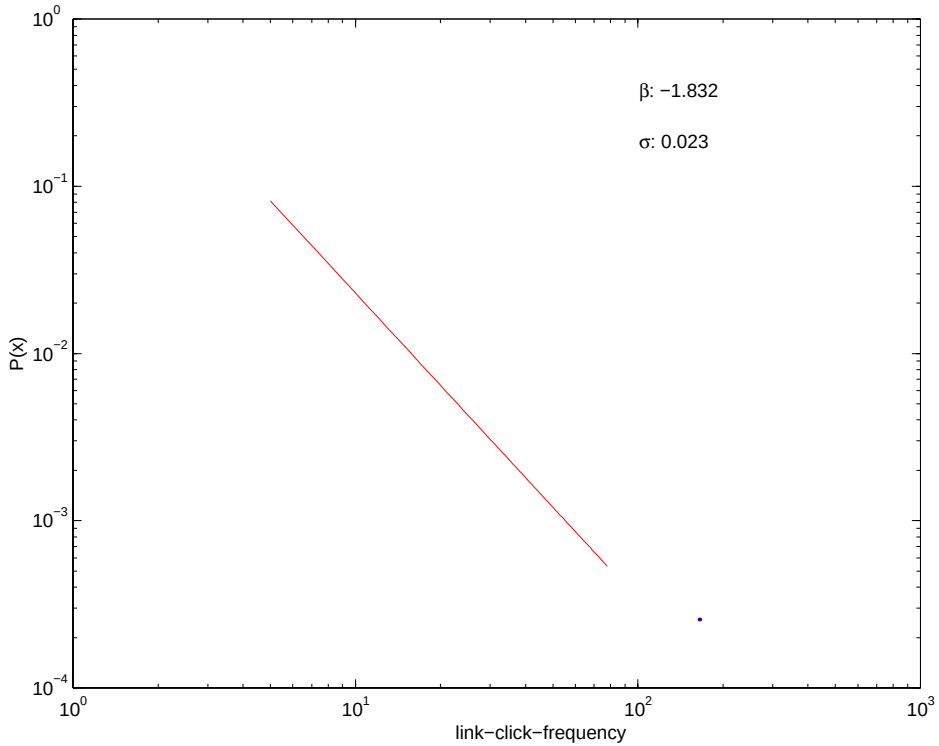
Cumulative distribution of foraging depth by recurrent entities. The obtained distribution follows an exponential function with exponent $\beta_c = -0.181$ and residual $\sigma = 0.08$.



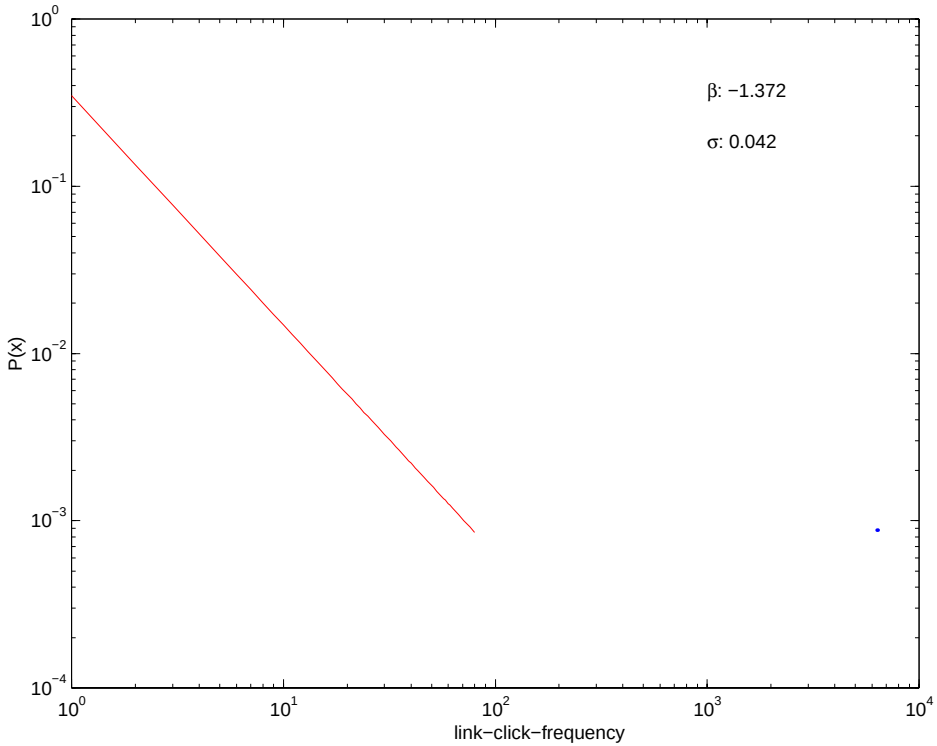
Cumulative distribution of foraging depth by rational entities. The distribution follows an exponential function with exponent $\beta = -0.133$ and residual $\sigma = 0.166$.



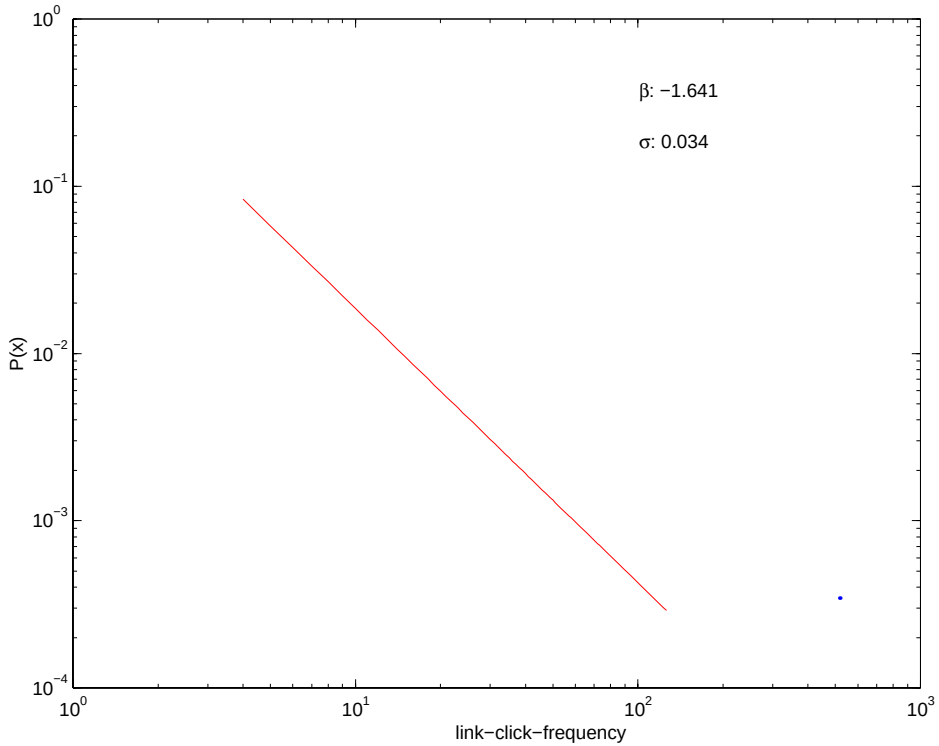
Distribution of link-click-frequency with recurrent entities. The distribution tail is approximately a power law with power $\beta_l = -1.832$.



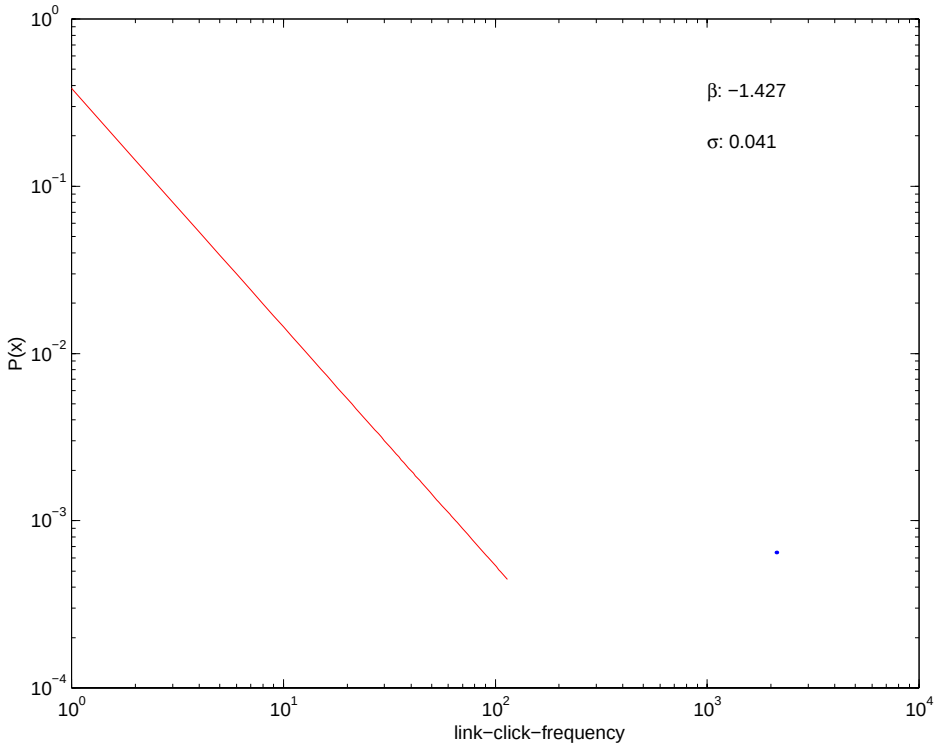
Distribution of link-click-frequency with rational entities. The distribution is approximately a power law with power $\beta_l = -1.372$.



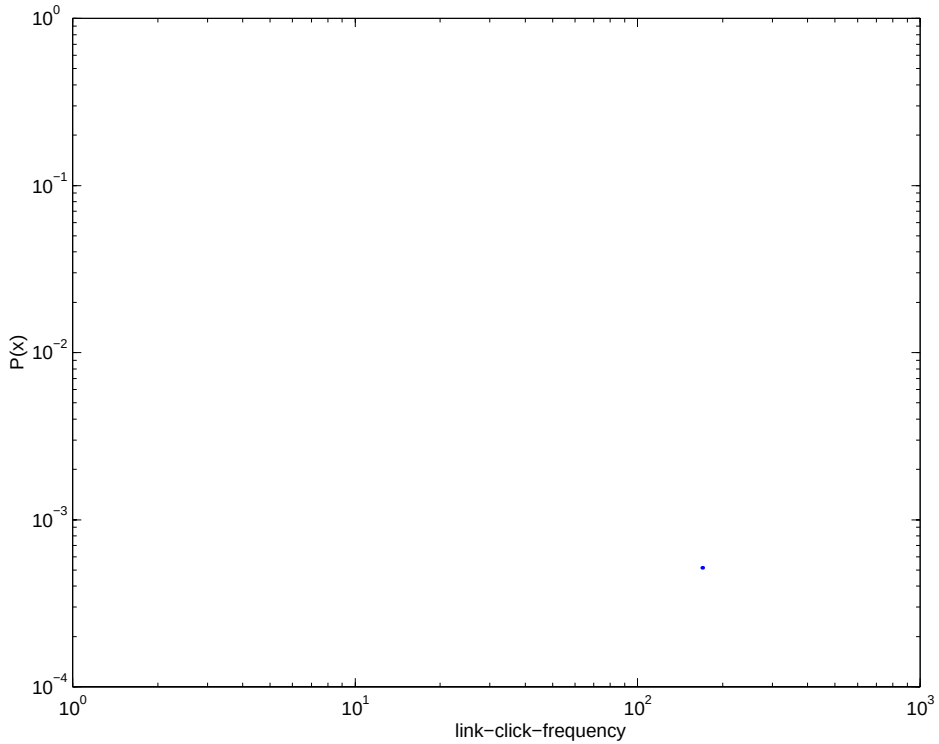
Distribution of link-click-frequency with recurrent entities. The distribution tail is approximately a power law with power $\beta_l = -1.641$.



Distribution of link-click-frequency with rational entities. The distribution is approximately a power law with power $\beta_l = -1.427$.

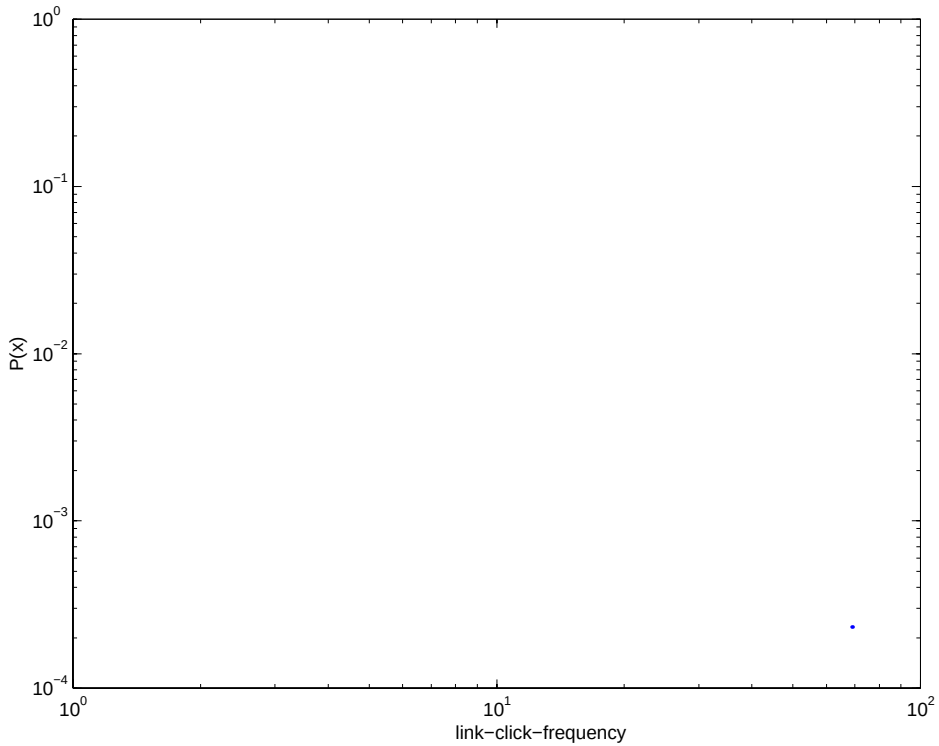


Link-Click-Frequency with *Random* Entities

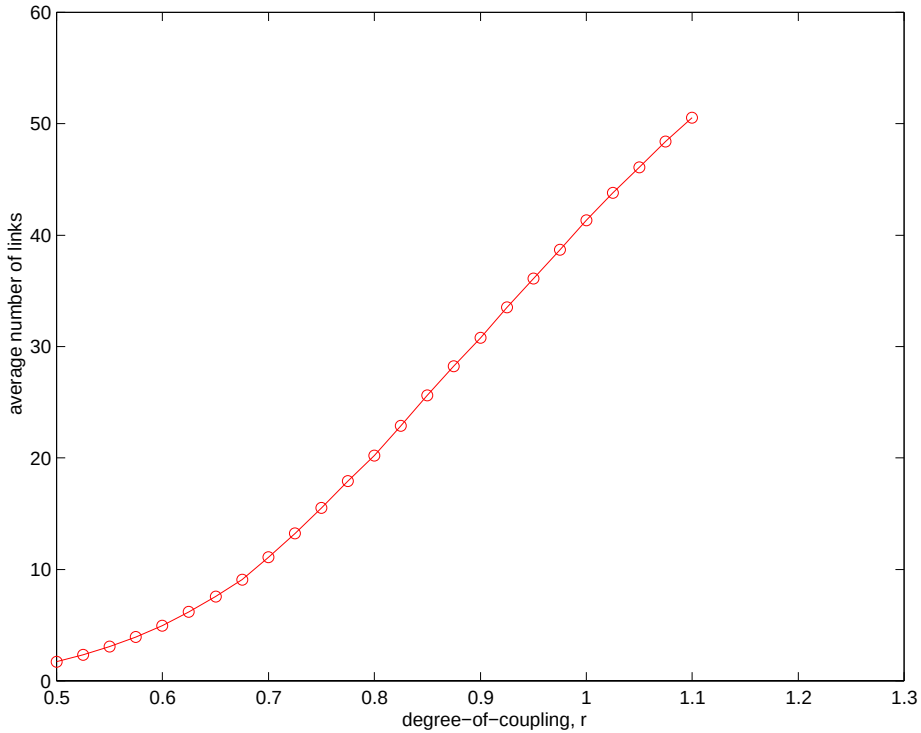


Link-Click-Frequency with *Random* Entities

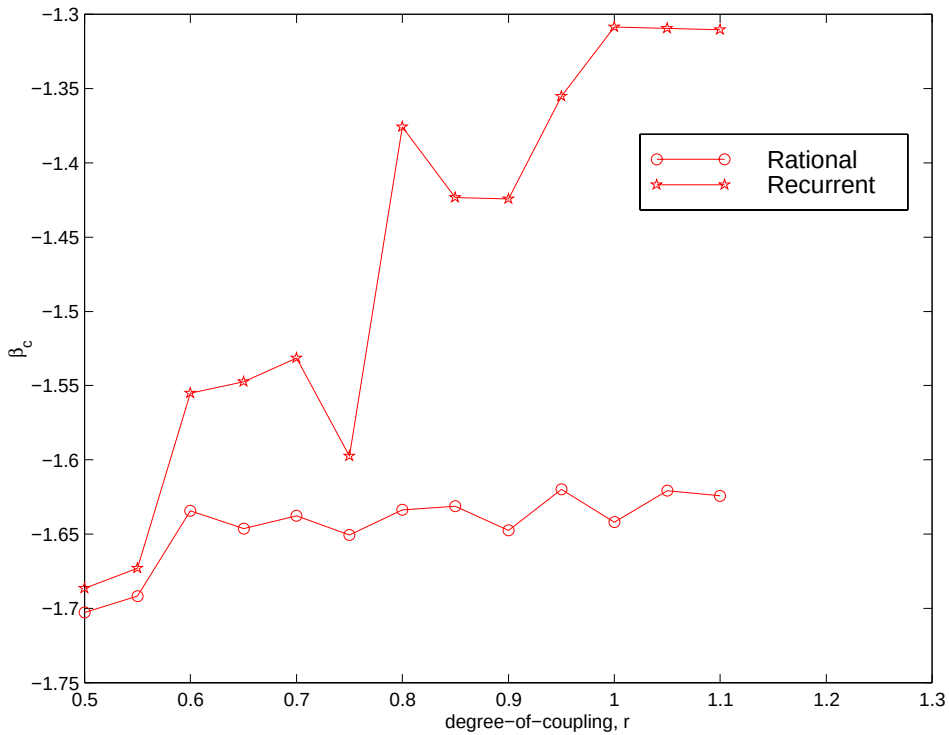
(content: power-law)



The Average Number of Links

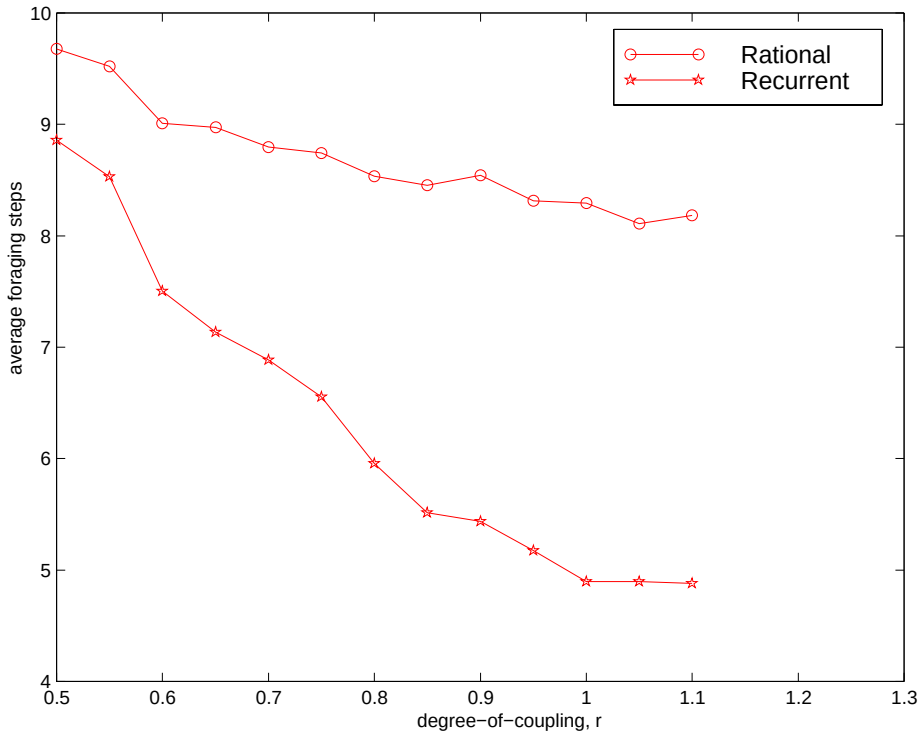


Power Values β_c in Power-Law Distributions of Foraging Depth



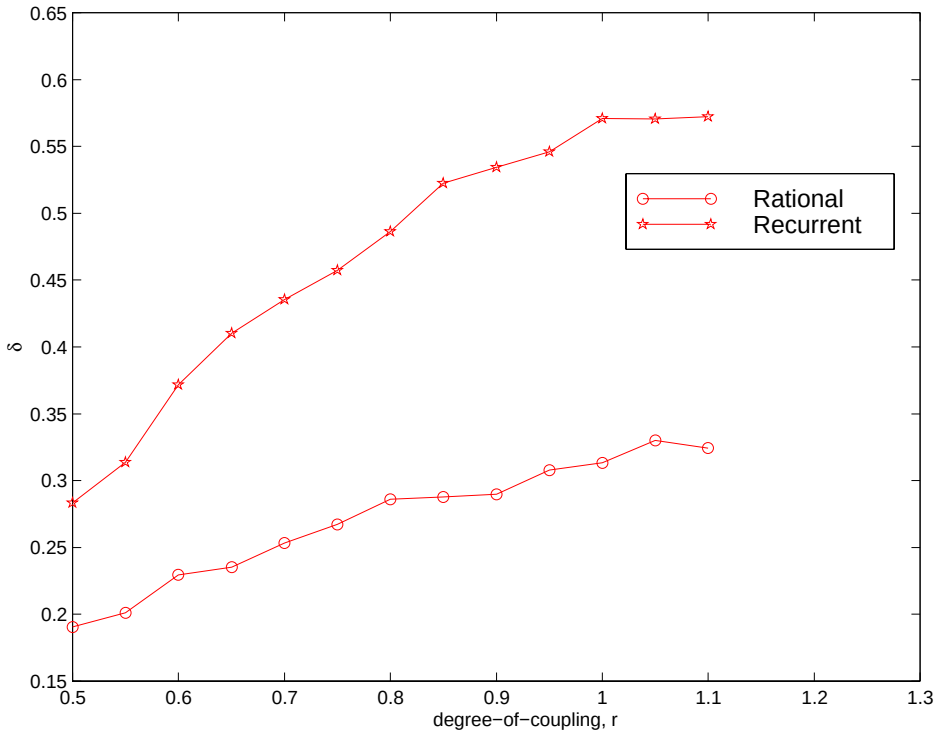
Average Foraging Steps

'$\circ$' corresponds to rational entities and '$\star$' corresponds to recurrent entities.



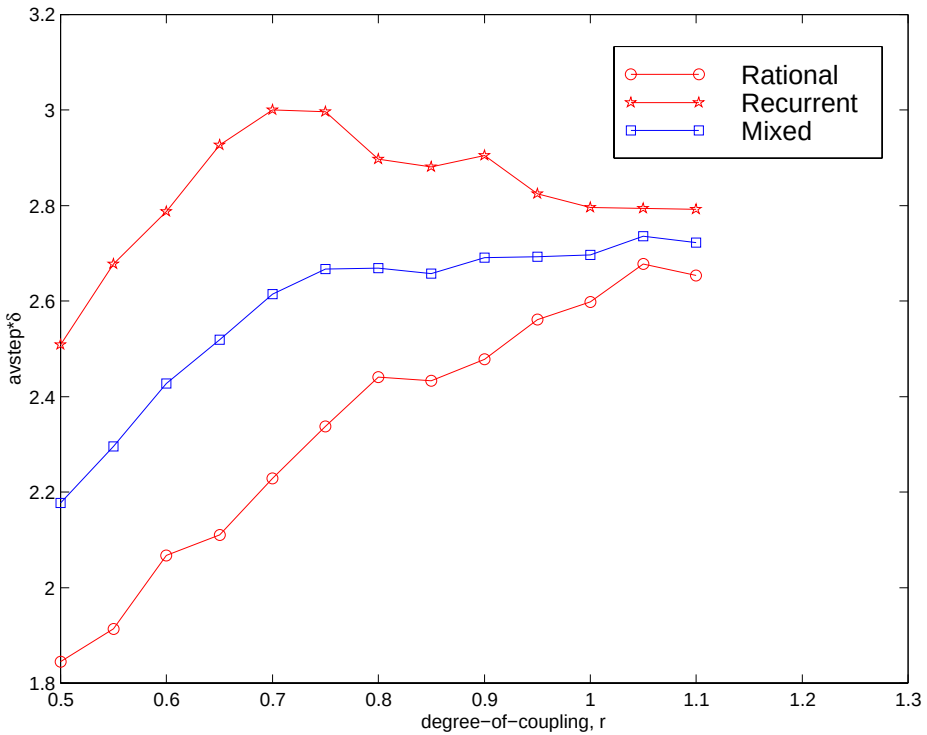
Satisfaction Rate

‘$\circ$’ corresponds to rational entities and ‘$\star$’ corresponds to recurrent entities.



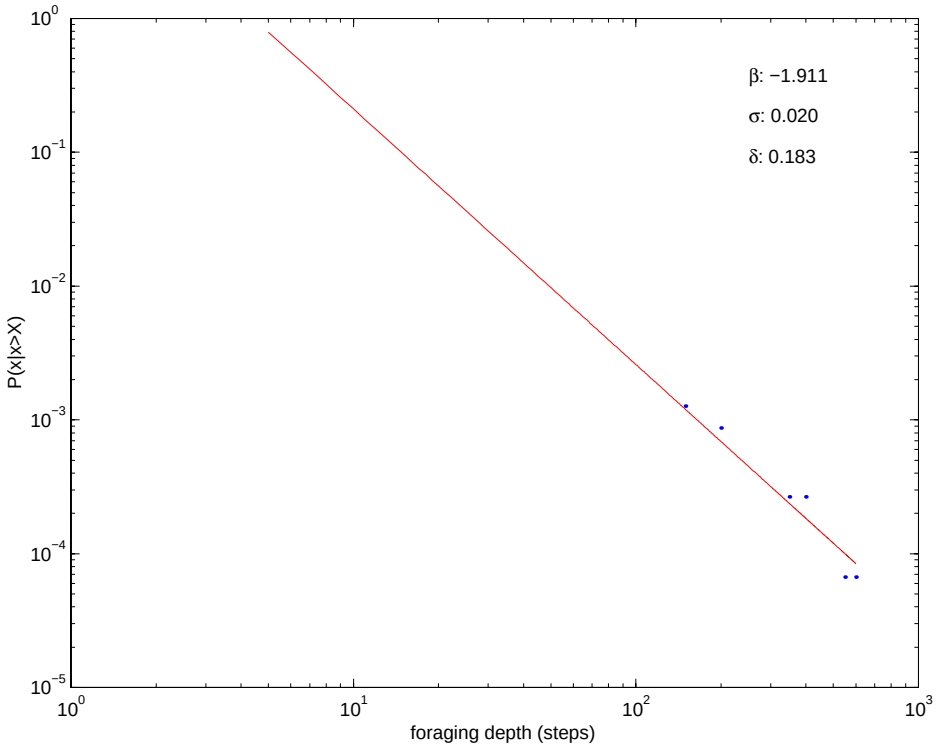
Combined Measure of Entity Foraging Depth and Satisfaction Rate

'○' corresponds to rational entities and '⋆' corresponds to recurrent entities.

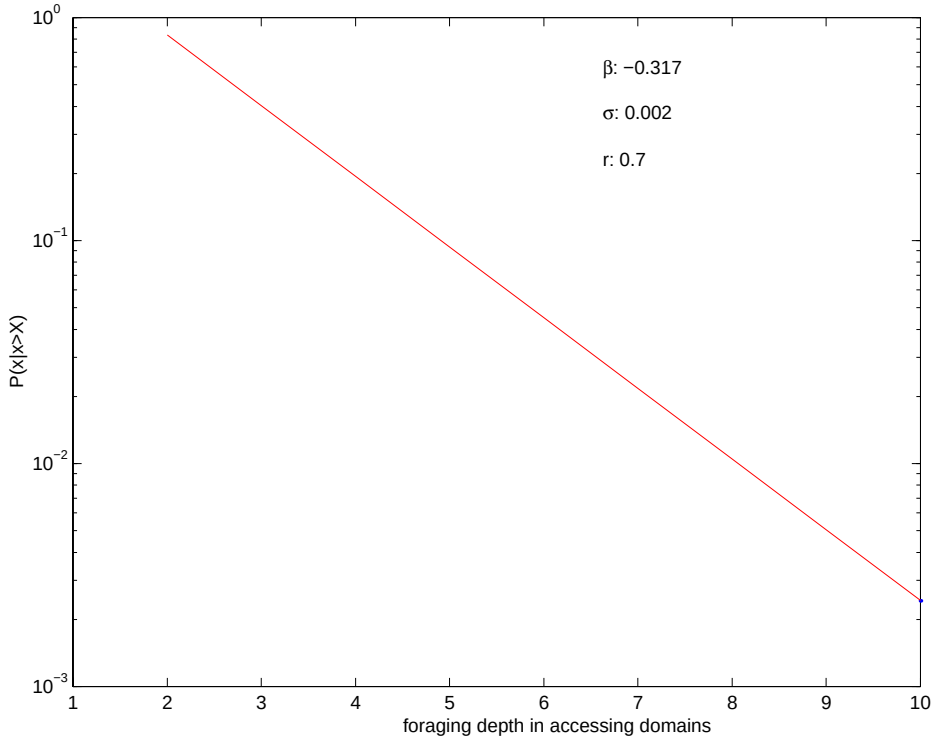


Number of Entities in Each Group = 5,000

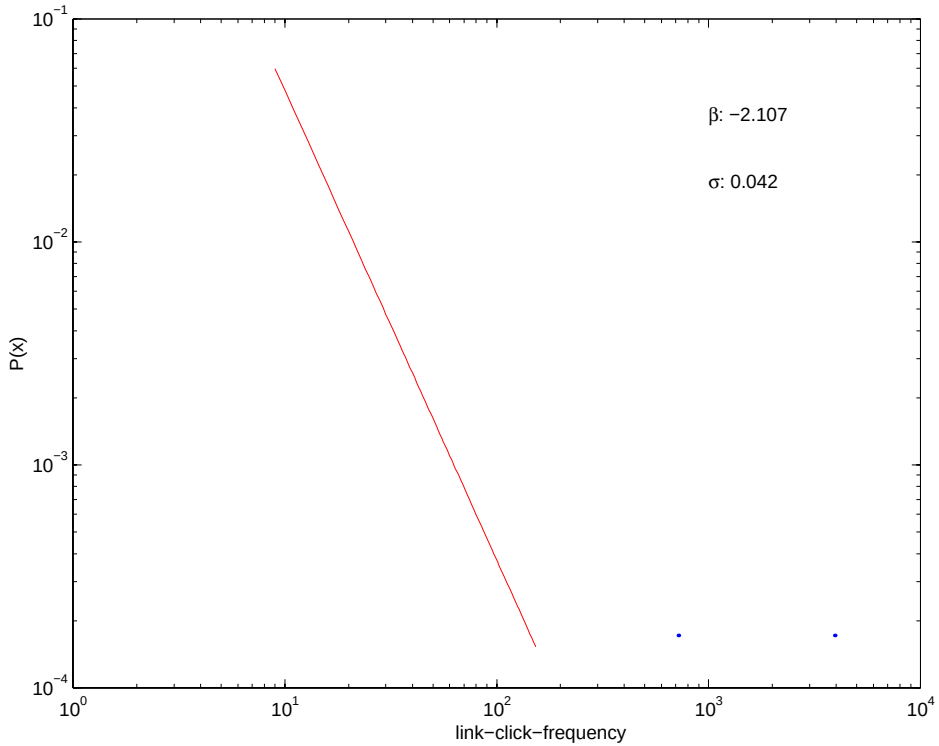
Cumulative distribution of foraging depth.



Cumulative distribution of foraging depth in accessing domains.

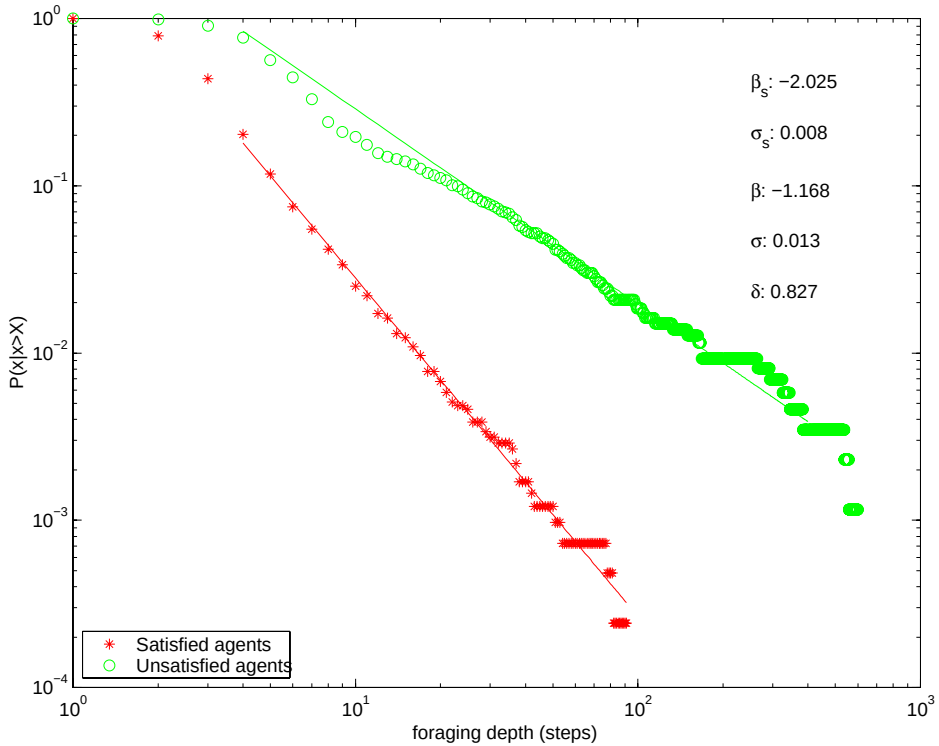


Distribution of link-click-frequency.

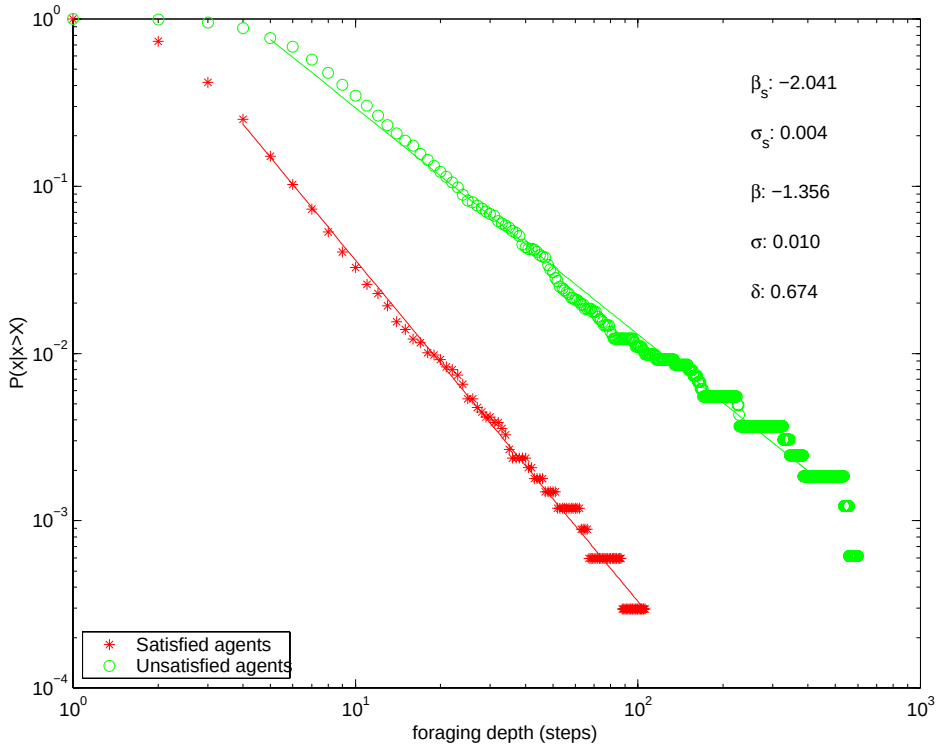


Difference in Foraging-Depth Distributions

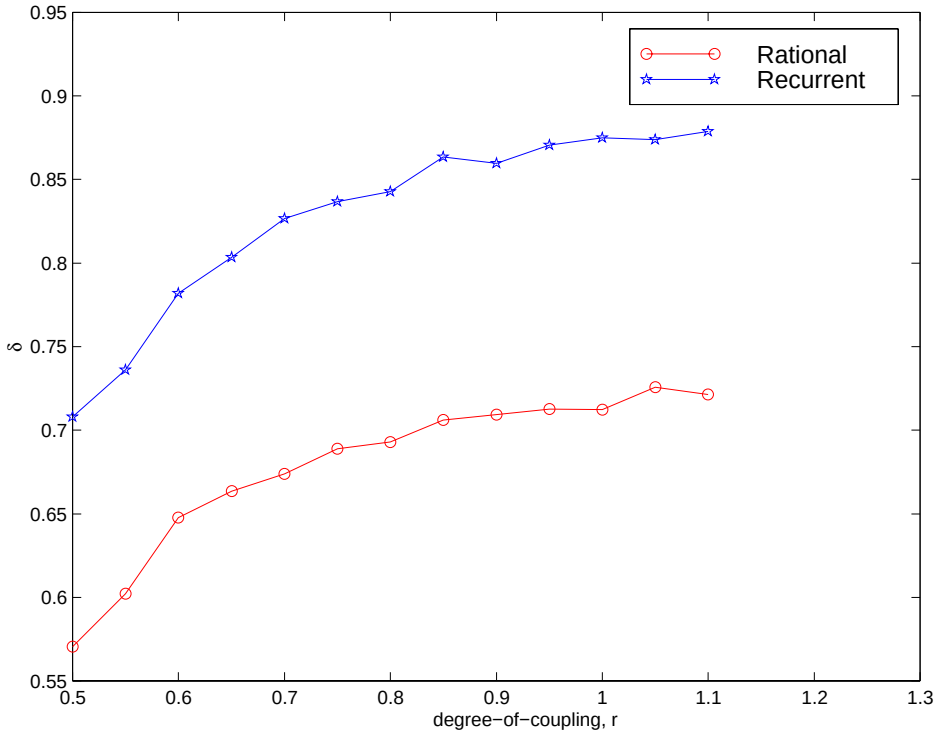
Cumulative distribution of foraging depth with recurrent entities. r=0.7. '$\circ$' corresponds to unsatisfied entities and '$\star$' corresponds to satisfied entities.



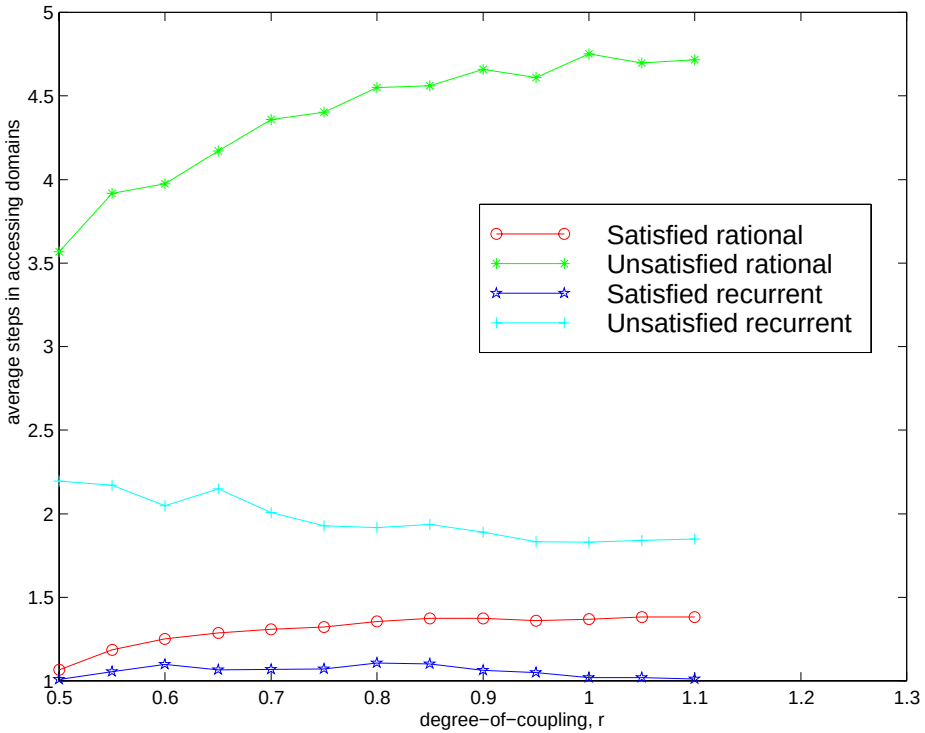
Cumulative distribution of foraging depth with rational entities. r=0.7. '$\circ$' corresponds to unsatisfied entities and '$\star$' corresponds to satisfied entities.



Satisfaction Rate



Average Steps in Accessing Domains



Remarks on Regularity Characterization

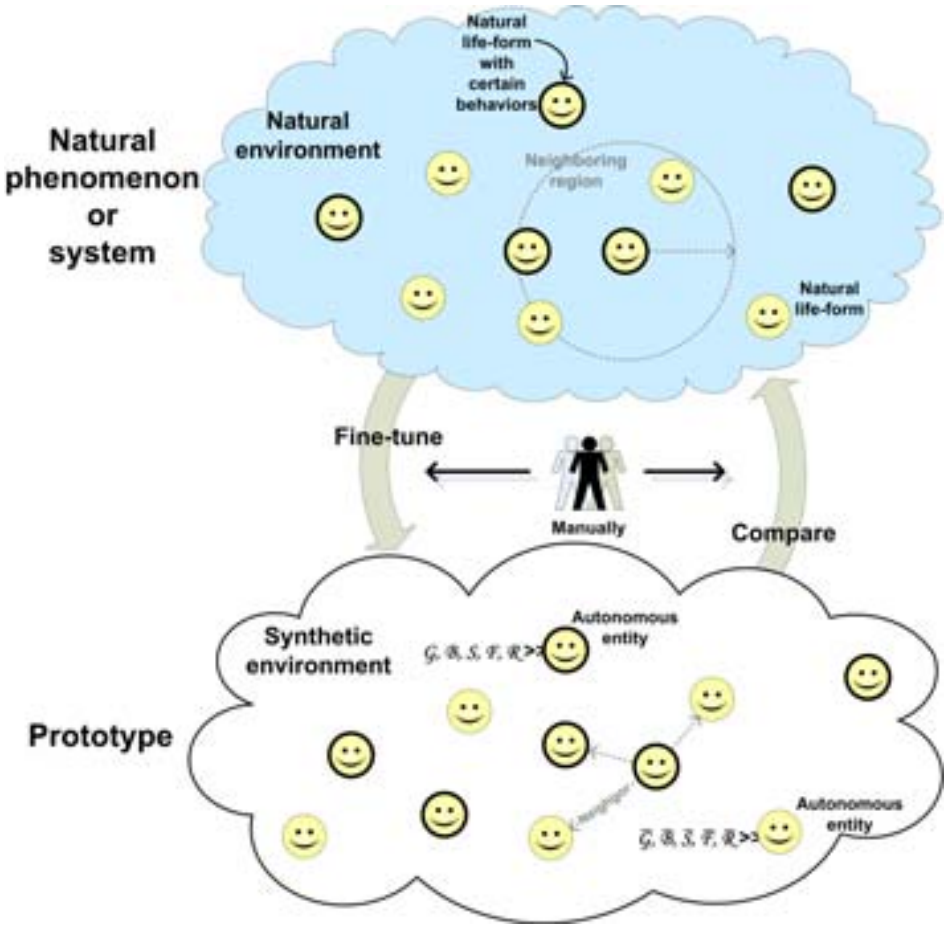
- Our goal is to characterize the underlying user behavior from the obtained data.
- We have shown a white-box, AOC model that takes into account the interest profiles, motivational support, and navigation strategies of users.
- We can experimentally obtain strong regularities in Web surfing and link-click-frequency distributions.
- We can further examine the effects on emergent regularities after certain aspects of the Web space or the foraging behavior are changed.
- Implications: It is useful for us to develop and structure Web contents, and at the same time, to analyze emergent user behavioral patterns.

Remarks on AOC by Prototyping

- It is commonly applied to uncover the working mechanism behind an observed, complex phenomenon or system.
- It starts with a step of hypothesizing and formulating a computational model of autonomous entities involved in the system, based on our prior knowledge and observations.
- It makes a transformation from the hypothesized model to an implemented prototype to characterize its natural counterpart.
- By observing the difference between the natural phenomenon or system and the synthetic prototype, we will manually fine-tune the prototype, especially the parameters in the behavior and interactions of autonomous entities.
  1. States, evaluation functions, goals, behaviors, and behavioral rules of an entity can be changed from one prototype to the next.
  2. The definition of the environment can also be changed from one version to the next. Even the model can be modified completely.

The AOC-by-Prototyping Approach

The trial-and-error process, i.e., iterative fine-tune and compare steps, is manually performed.



Example: Telecommunication Network Design

- A telecommunication network connects base stations on different positions together where the communication equipment, such as host computers, concentrators, routers, and terminals, is located.
- Designing a telecommunication network is to find the most efficient way to connect the base stations.
- This problem is sometimes called a *minimum spanning tree problem*.

Formulation

Given a group of base stations, $V = \{1, 2, \cdots, n\}$, and the cost, c_{ij}, for connecting two stations i and j, the problem is actually to determine a graph $G = \langle V, E \rangle$, where $E = \{\langle i, j \rangle \mid i, j \in V\}$:

$$\begin{array}{ll} \min & \sum_{i=1}^{n-1} \sum_{j=2}^{n} c_{ij} x_{ij}, \\ \text{s.t.,} & \sum_{i=1}^{n-1} \sum_{j=2}^{n} x_{ij} = n - 1, \\ & \sum_{j=1}^{n} x_{ij} < d_i, i \in V, \\ & x_{ij} = 0 \text{ or } 1, i, j \in V, \end{array}$$

where d_i is the upper bound of the number of links connected to base station i, and

$$x_{ij} = \begin{cases} 1, & \textit{if link} \langle i, j \rangle \textit{ is selected in a spanning tree}, \\ 0, & \textit{otherwise}. \end{cases} \tag{70}$$

Objectives

- Consider the goal being written as a function $F(x)$ where $x = \{x_1, x_2, \cdots, x_n\}^T$ is an n-dimensional vector representing the parameters of function F. The optimal solution is represented by $F(x^*)$ such that

$$\forall\, x,\ F(x^*) \leq F(x). \tag{71}$$

  The search for x^* can be viewed as the minimization of function F (Turning the sign in Equation 71 around makes the search for x^* a maximization task) $-$ global optimization tasks [Torn and Zilinskas, 1989].
- To solve the minimum spanning tree problem by a population-based algorithm, it is appropriate to encode the tree structure within an autonomous entity.
- Possible autonomous behaviors include pruning one's own tree or exchanging a partial tree with another autonomous entity.

Challenges

- The *landscape* of the function to be optimized is unknown.
- There is usually *no linear relationship* between changes made to the function variables and the corresponding change in the function value.
- Search algorithms do not normally jump directly to the optimal solution, but make *incremental* changes in small steps instead.
- A population-based search algorithm needs to maintain a sufficient *diversity* during the whole course of the search so that the search space is adequately sampled.

Related Work

- Many algorithms have been developed over the years to tackle the challenging task of global optimization [Horst and Pardalos, 1995, Horst and Tuy, 1990, Mockus, 1989].
- In the absence of prior knowledge about the search landscape, stochastic methods, such as simulated annealing [Kirkpatrick et al., 1983] and population-based incremental learning [Baluja, 1994, Baluja and Caruana, 1995], have been proven to be effective. They attempt to locate the optimal solution by generating sampling points probabilistically.
- Methods inspired by nature that are equally successful include evolutionary algorithms [Bäck et al., 1997], bacterial chemotaxis [Müller et al., 2002], and differential evolution [Storn and Price, 1997].

- **Objectives:**
  1. To tackle the task of optimizing multi-dimensional functions.
  2. To learn the search landscape by group efforts and information sharing (to facilitate the small step exploration);
  3. To maintain a high population diversity;
  4. To automatically adjust search step sizes.
- **Principle:** Diffusion in nature and the successful application of the diffusion models to image segmentation have inspired the evolutionary diffusion optimization (EDO) algorithm.

The EDO Algorithm I

while (the number of entities $>$ 0) and (the number of iterations $<$ limit) **do**
  Evaluate an entity;
  if the entity performs better than its parent **then**
    Get positive feedback;
    Reproduce;
    Its parent becomes inactive;
  else
    Get negative feedback;
    Diffuse;
    Age;
  end if
end while

Procedure *Diffuse*

The EDO Algorithm II

if rand() $>$ P_{rand_move} **then**
  Random-move;
else
  for each variable **do**
    Select its step direction and size;
  end for
end if

Procedure *Reproduce*
quota $\leftarrow$ $f(fitness)$;
Create a new probability matrix;
for each offspring **do**
  Copy variables and point to the new probability matrix;
  Diffuse;
end for

The EDO Algorithm III

Procedure *Age*
age $\leftarrow$ age +1;
if (its fitness $<$ its parent's fitness $\times$ threshold) or
((age $>$ lifespan) and (its fitness $<$ the average value)) **then**
  Remove;
end if

Diffusion

A **diffusion** behavior is an operation through which an entity modifies its object vector, which is a set of values corresponding to the variables of the function to be optimized.

$$\mathbb{V} = \{v_1, v_2, \ldots, v_n\}, \ \forall\, i, \ v_i = [LB, UB], \tag{72}$$

where LB and UB are the lower and upper bounds, respectively. All function variables can take values within these bounds.

Types of Diffusion Behavior

- **Rational-move** chooses the number of steps to take, according to a probability matrix. The actual amount of change is the product of the current step size and the number of steps chosen:

$$\forall\, i,\ v_i = v_i + \delta v_i \cdot \Delta, \tag{73}$$

$$\delta v_i = \min\{k \mid \text{rand}() < \sum_{j=-y}^{k} p_{i,j},\ k \leq y\}, \tag{74}$$

where v_i is the ith function variable, δv_i is the number of steps to be taken, Δ is the step size, y is the maximum number of allowable steps towards either end of the bounds in the search space, and $p_{i,j}$ is the probability of v_i making j step(s).

Types of Diffusion Behavior

- **Random-move** is performed with an increasing probability:

$$\mathbb{P}_{rm} = exp\left[-\frac{\Theta - a}{\alpha}\right], \tag{75}$$

where α is a scaling factor that decides the degree to which the random-move is to be exercised, Θ is the maximum lifespan of an entity, and a is the age of an entity.

$$\forall\, i,\ v_i = v_i + \text{rand}() \cdot (r_{<l>} - v_i), \tag{76}$$

$$l = \min\{k \mid \text{rand}() < \sum_{j=1}^{k} b_j,\ k \leq 3\}, \tag{77}$$

$$r = \{LB, v_i, UB\},\ \forall\, j,\ s_j = 1/3, \tag{78}$$

where r is the set of boundaries between which a new value will be chosen for object variable v_i, and s is the set of probabilities for choosing the entries in r.

Reproduction

- At the end of an iteration in EDO, the fitness values of all active entities are compared with those of their parents.
- All entities with higher fitness values will perform reproduction.
- Fitness, f, measures an entity's degree of success in the course of the search for the optimal solution. It is also used as a basis to determine various primitive behaviors.
- For simplicity, the objective function value is used as fitness in EDO if the task is minimization (The reciprocal of the function value can be used as fitness if EDO is used in a maximization task).

Reproduction Quota

- The number of offspring entities to be reproduced, i.e., quota q, is governed by the entity's fitness and two system-wide parameters: maximum offspring, Ω, and maximum population size, Π.
- Two rules are applied in succession:
  - **Differentiation Rule:** An entity is given the full quota to reproduce, only if its fitness is significantly above the population average and will gradually decrease as the fitness decreases.

Therefore, the quota for an entity e having fitness f is:

$$q_x = \begin{cases} \Omega, & \text{if} \quad \frac{f_e}{\bar{f}} \leq \omega_1, \\ \Omega - 1, & \text{if} \quad \omega_1 < \frac{f_e}{\bar{f}} \leq \omega_2, \\ \Omega - 2, & \text{otherwise}, \end{cases} \tag{79}$$

where $\bar{f}$ is the population average fitness, and ω_1 and ω_2 are the intervals in the step function.

Population Size Rule: The reproduction quota is subjected to a further restriction to avoid overcrowding:

$$q_x = \lfloor \frac{q_x * (\Pi - \Sigma)}{\Pi} \rfloor, \tag{80}$$

where Σ is the size of the entity population, and Π is the maximum population size.

Rejuvenation

- An inactive entity will be allowed to perform a primitive behavior to spawn new offspring if the following two conditions are satisfied:
  - All its offspring entities are dead.
  - Its fitness is better than the population average.
- **Main idea:** An inactive entity has been receiving positive or negative reinforcement signals from its offspring , its probability matrix contains the latest information regarding the neighborhood landscape. It would be a waste if this potentially useful information is discarded.
- A rejuvenated parent will be given the full quota, Ω, and then subjected to the population size rule (Equation 80) to reproduce its offspring.

Aging

- EDO keeps track of the unproductive moves throughout the search.
- The age, a, of an entity in EDO is the number of iterations for which this entity has survived since its birth.
- Once an entity becomes a parent, the age does not need to be updated any more.
- All entities in EDO will only be allowed to perform search for a certain number of iterations, because we do not want to have too many non-contributing entities in the system.
- The global lifespan information, Θ, is the maximum number of allowable iterations for which any entity can survive.

- **Extended life:** The lifespan limit of an entity is extended by one iteration when it expires, if its fitness is higher than the population average.
- **Sudden death:** An unsuccessful entity will be eliminated, if its fitness is less than a certain percentage of the fitness of its parent. The threshold is set at 80% in the experiments reported later.

Feedback

- Each entity in EDO performs an **information-passing** behavior to pass information back to its parent, if it has moved to a better or worse position.
- This information allows the parent to update its probability matrix, which is shared among its offspring.
- A **probability matrix**, p, contains the likelihood estimate of success with respect to the direction of a move.
- Note that it is an $n \times m$ matrix representing n function variables to be optimized and m possible steps (including $y = (m-1)/2$ steps towards the upper bound and the lower bound, respectively, and the current position).

A global step size parameter, Δ, governs the unit of change in all function variables. The product of Δ and the number of steps becomes the final modification to affect on v. Formally,

$$p = \{p_1, p_2, \ldots, p_n\}, \tag{81}$$

$$p_i = \{p_{i,-y}, \ldots, p_{i,0}, \ldots, p_{i,y}\}, 0 \leq p_{i,j} \leq 1, \tag{82}$$

$$\forall\, i, \sum_{j=-y}^{y} p_{i,j} = 1. \tag{83}$$

Information Sharing I

All offspring entities of the same parent use the same probability matrix – note that a trend is a kind of local information, it will become irrelevant to places further away from a parent.

- **Positive feedback:** To bias the future moves of an entity's siblings to its own successful move (i.e., with a gain in fitness), we update the probabilities in the parent's probability matrix (which correspond to the changes in the successful entity's object vector):

$$p_{i,j} = \frac{p_{i,j} + \beta}{1 + \beta}, \tag{84}$$

where $p_{i,j}$ is the probability that relates to the *ith* function variable and *jth* step size, and β is the learning rate.

Information Sharing II

- **Negative feedback:** To steer the siblings of an entity away from a non-optimal area in the search space, an entity will update the probability matrix of its parent after each unsuccessful move:

$$p_{i,j} = p_{i,j} \cdot (1 - \beta), \tag{85}$$

where β is the same learning rate as used in positive feedback.

EDO Summary

- An entity, e, in the population, $\mathtt{P}$, maintained by EDO is a tuple $(\mathtt{V}, p, a, f)$, where
  - $\mathtt{V}$ is the object vector,
  - p is the probability matrix,
  - a and f are scalars representing the age and fitness of entity e, respectively.
- While $\mathtt{V}$ contains the values of the potential solution, p, a, and f are crucial to the search process of EDO.

Four benchmark functions are chosen to test the EDO algorithm [Yao and Liu, 1997, Yao et al., 1999]. f_1 and f_2 are unimodal functions, while f_3 and f_4 are multimodal functions:

$$f_1(x) = \sum_{i=1}^{n} x_i^2, \tag{86}$$

$$f_2(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2, \tag{87}$$

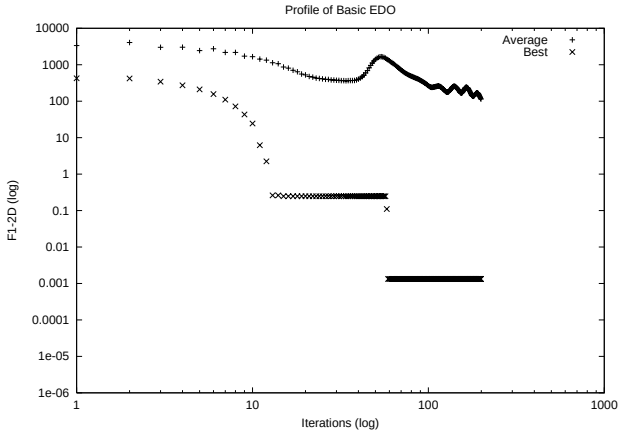
$$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10)], \tag{88}$$

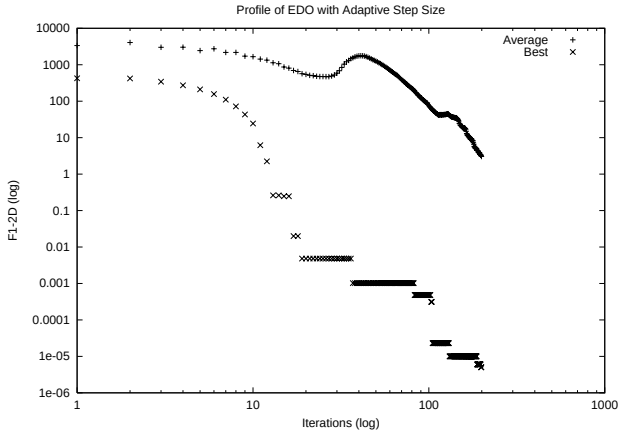
$$f_4(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}, \tag{89}$$

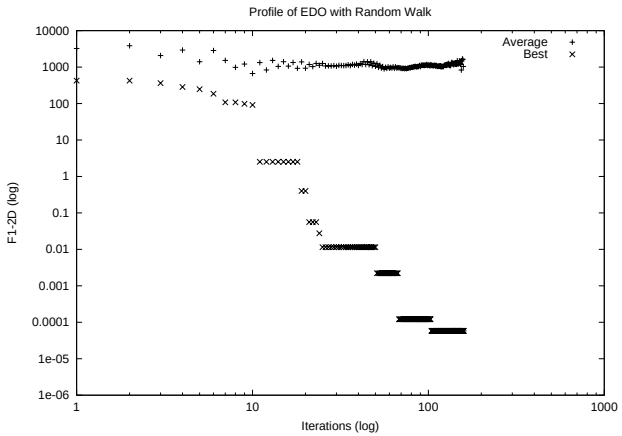
where

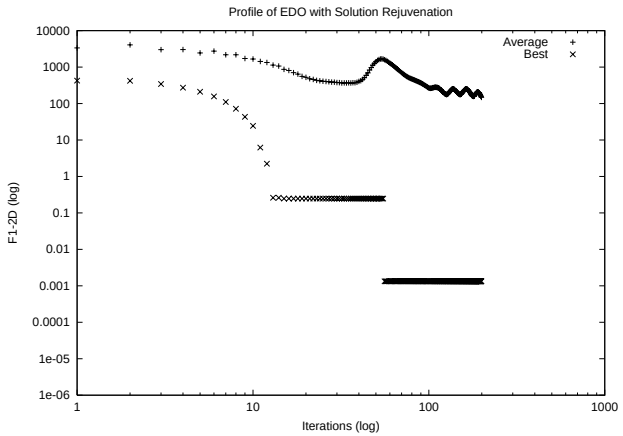
$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix} \tag{90}$$

The best and average values for a solution to unimodal function f_1. (a) basic EDO; (b) adaptive step size; (c) random-move; (d) rejuvenation.

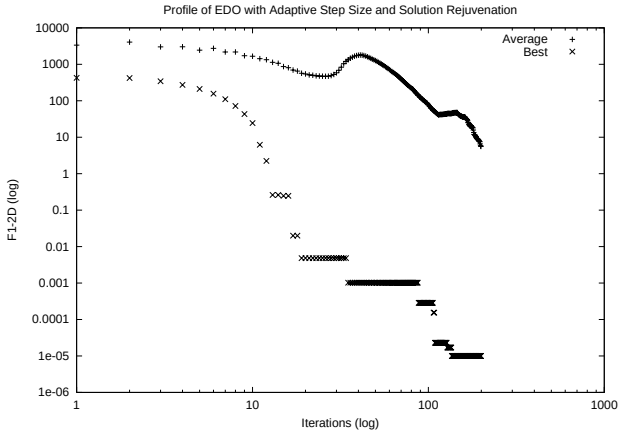


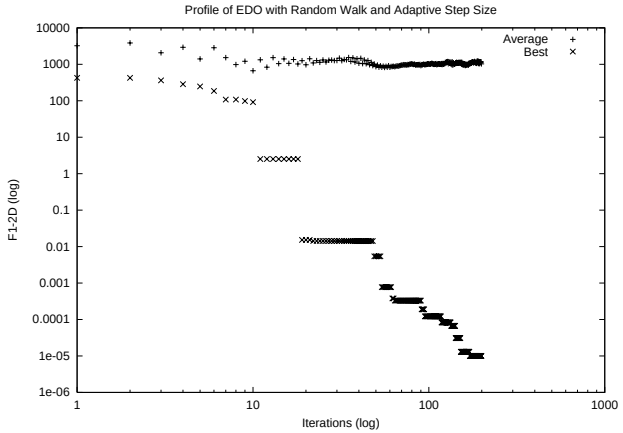


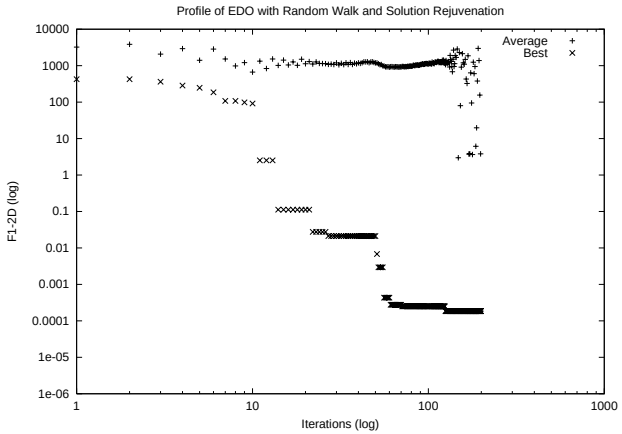


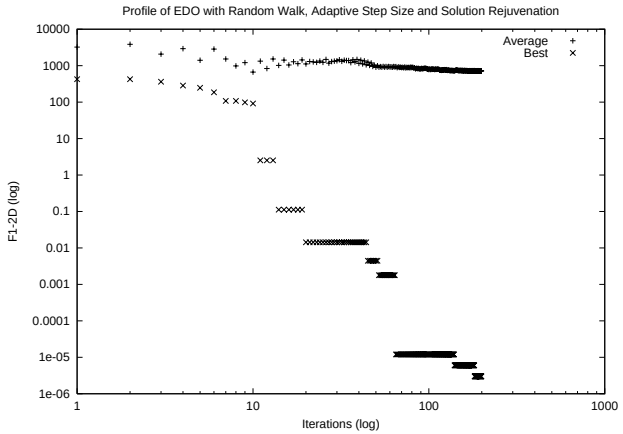


(a) Combination of step size adaptation and solution rejuvenation; (b) random-move and step size adaptation; (c) random-move and solution rejuvenation; (d) all three strategies together.



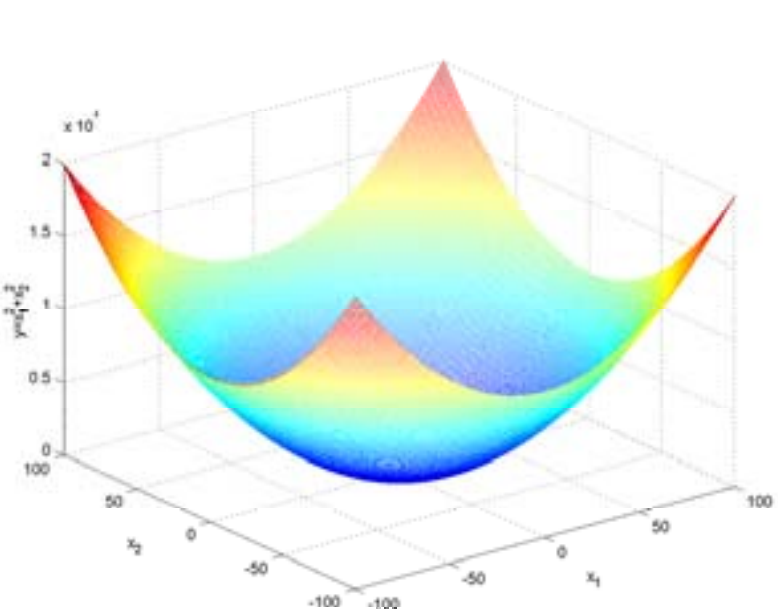


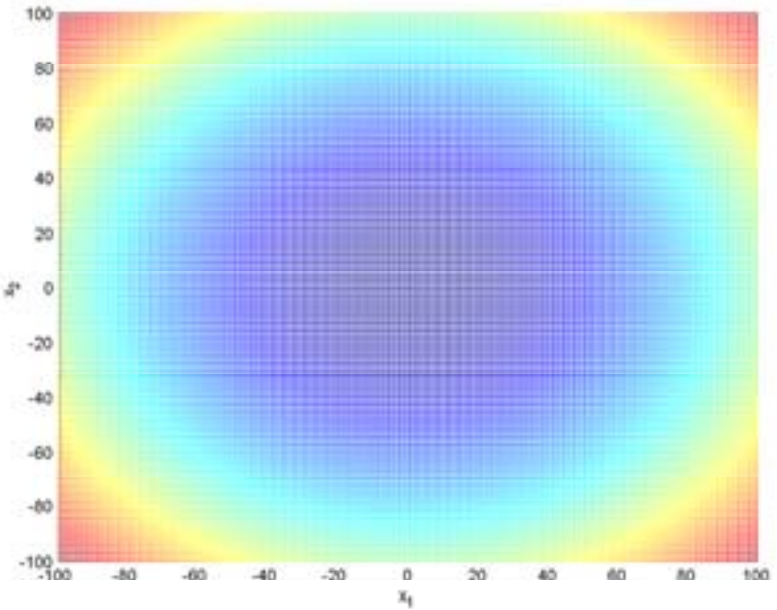




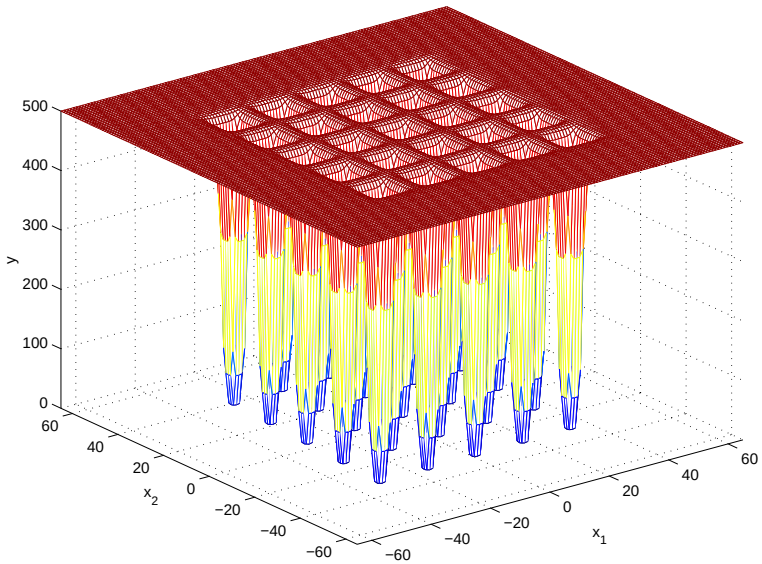
Basic EDO with More Features

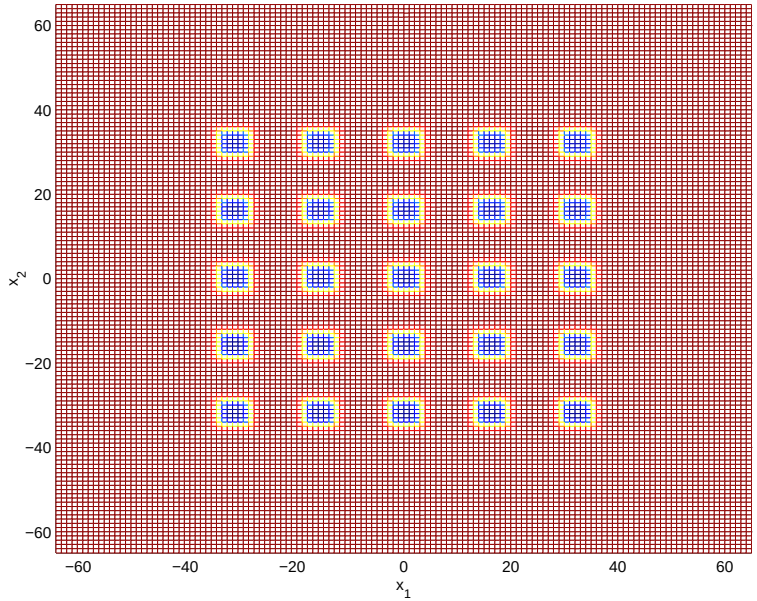
Clustering around the Optimal Solution: The two-dimensional version of f_1.



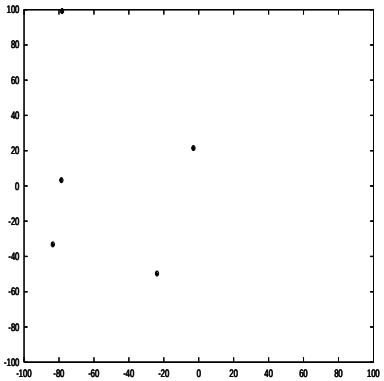
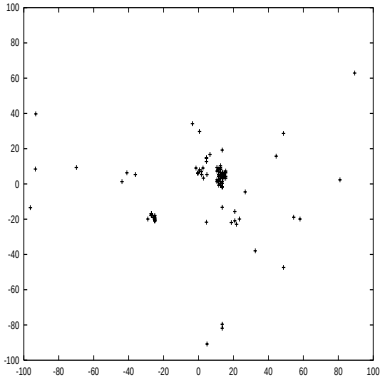
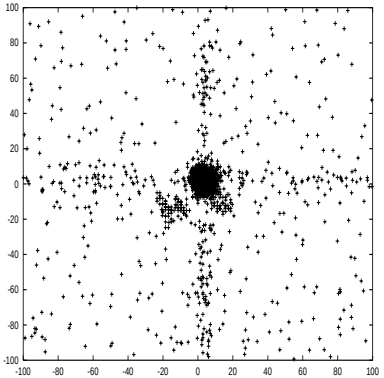
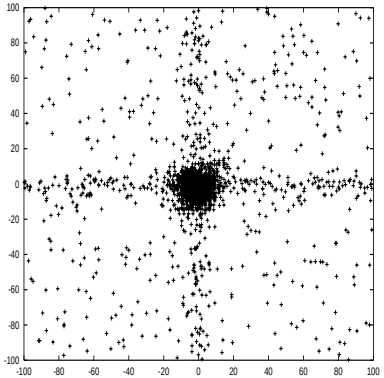
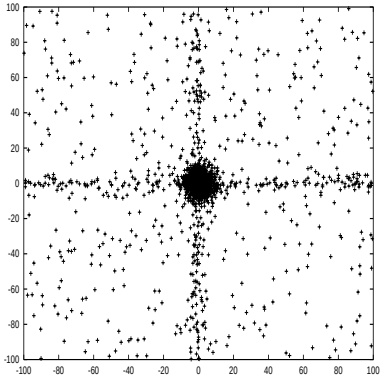
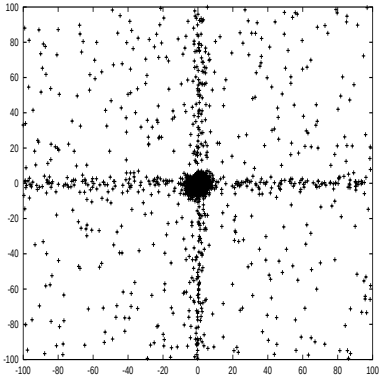


The multimodal function f_4.

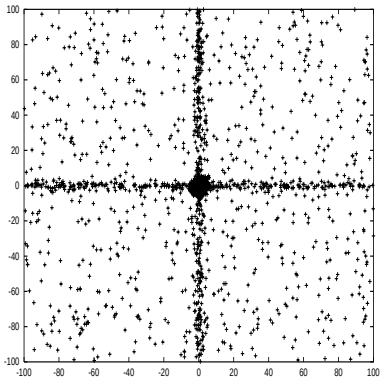
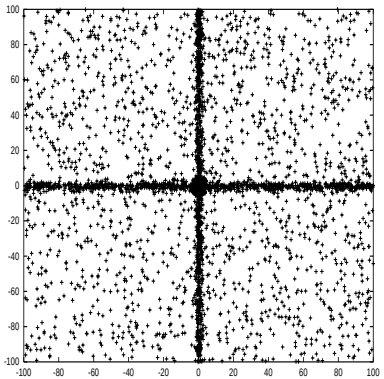
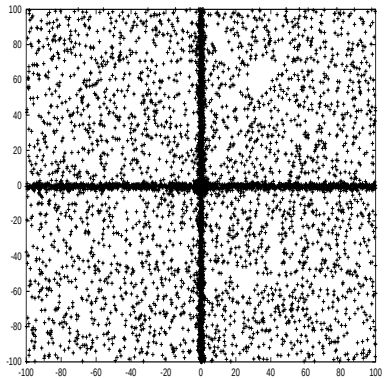
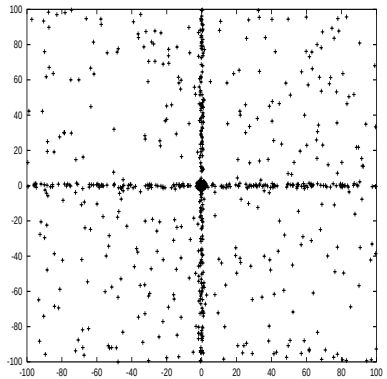
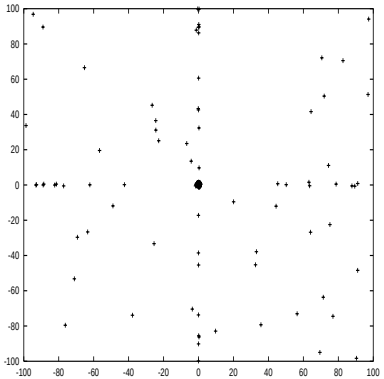
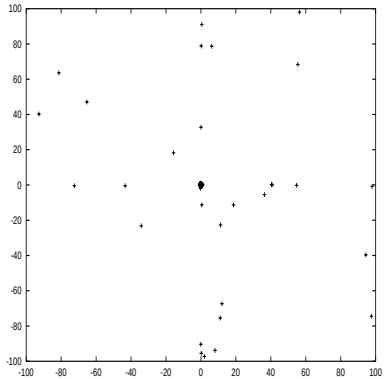




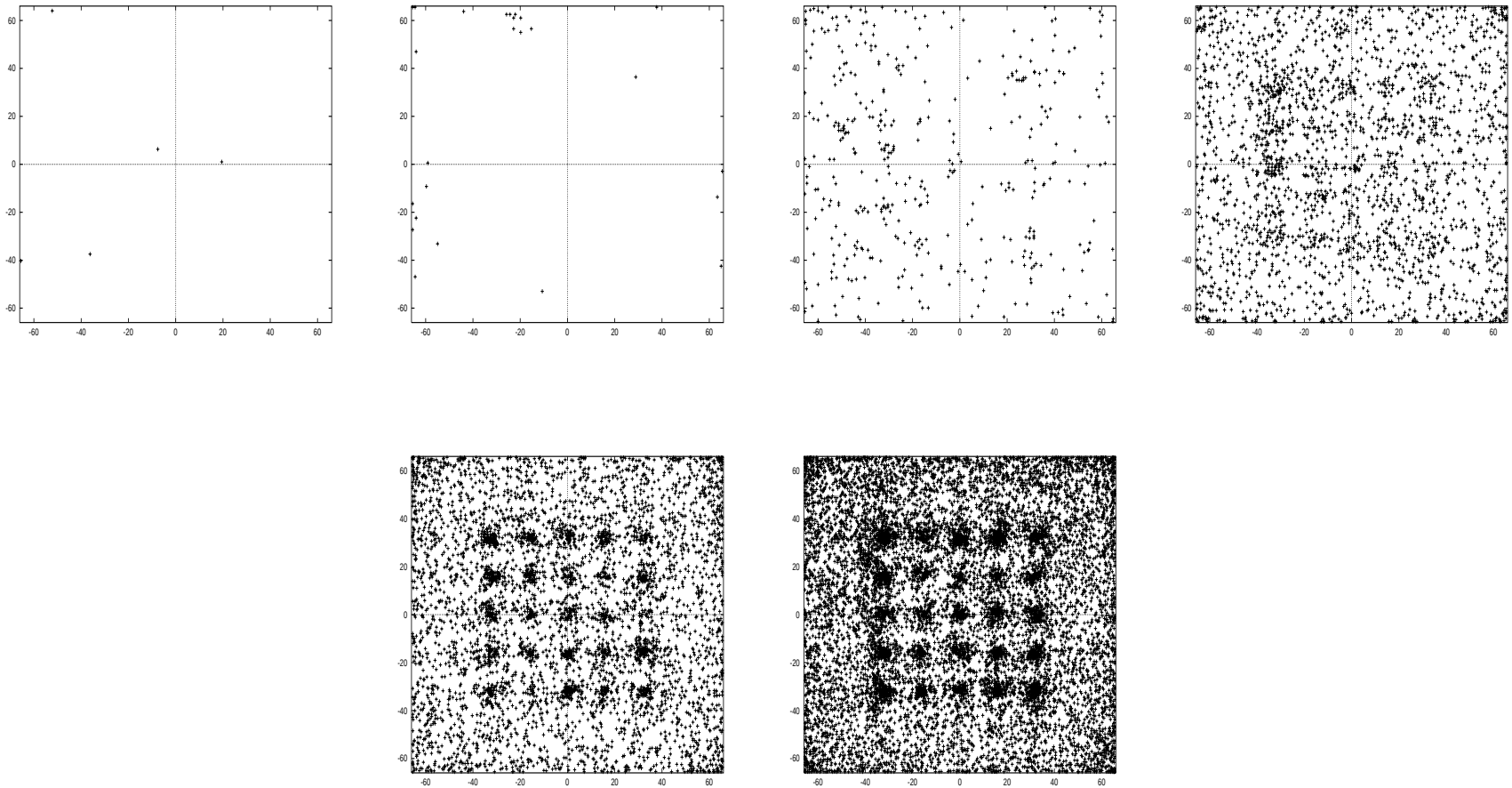
Entity distribution in the search space for two-dimensional unimodal function f_1 (iteration =0-50).

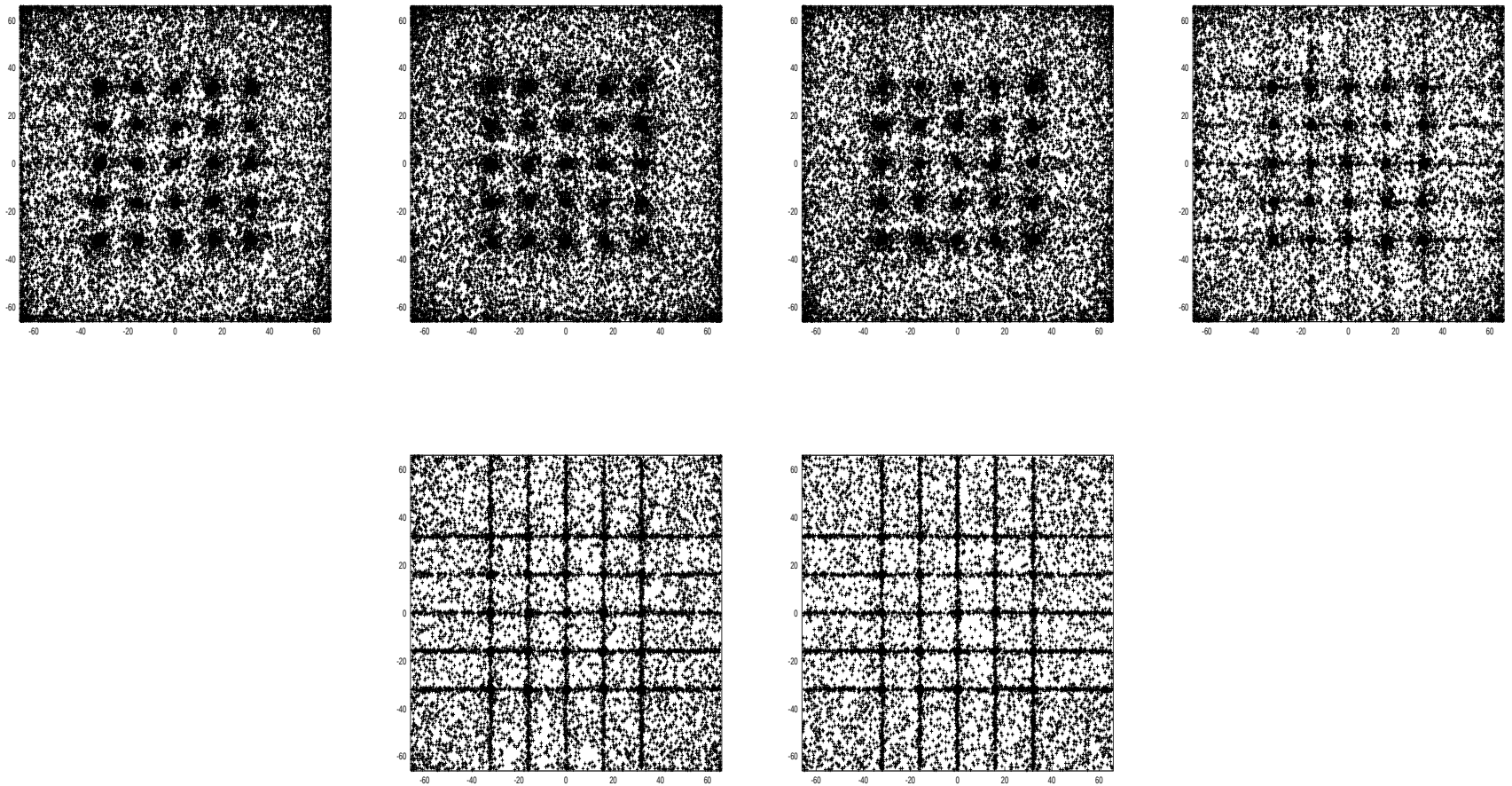
Entity distribution in the search space for two-dimensional unimodal function f_1 (iteration =60-110). As the search progresses towards the end, the number of entities remaining around the suboptimal solutions decreases.

Entity distribution in the search space for multimodal function f_4 (iteration=0-90).



Entity distribution in the search space for multimodal function f_4 (iteration=110-200).



Observations

- Adaptive step size achieves the best solution followed by random-move, if the function value is chosen as the evaluation criterion for the features.
- When more than one feature are considered, all three features together have a slight advantage over adaptive step size alone.
- Step size adaptation helps exploit existing solutions, while random-move behavior is useful for exploring new areas in the solution space.
- Entities will tend to spread randomly across the search space. But once a suboptimal solution is found, they will be reluctant to leave there.
- The random-move behavior then helps push the entities out of the suboptimal region.

- Evolutionary diffusion optimization (EDO) usually involves a large number of distributed autonomous entities.
- Through local and nonlinear interactions among entities, the behavioral results of entities will be self-aggregated and consequently certain complex patterns or behaviors emerge.
- EDO utilizes this mechanism and relates the resulting emergent complex patterns or behaviors to the solutions to an optimization problem at hand.
- EDO is well suited for solving the type of optimization problems that are characterized as being large-scale, highly distributed.

Remarks on EDO Algorithm

- Each entity is equipped with the primitive behaviors to diffuse to its local neighborhood and reproduce.
- It can also choose to perform random-move behaviors.
- Decisions on the course of behaviors are made by an entity based on the common information shared with its parent and its siblings.
- The common information contains the likelihood estimates of finding a good solution in a certain direction, and is updated by every member of the family – reinforcing positively the good moves and negatively the bad moves.
- EDO also has a mechanism to adapt its step size during the search.

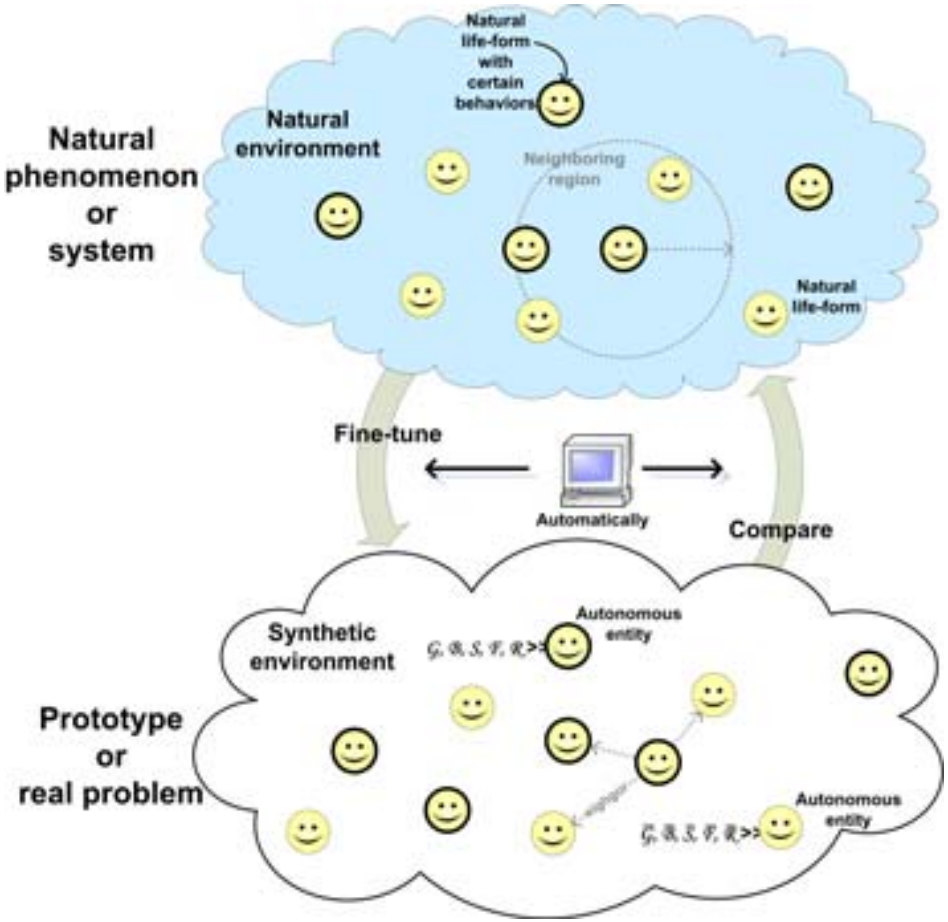
Remarks on EDO Performance

- EDO performance shows that it can maintain a high diversity in the population of entities throughout the search – a crucial feature to avoid premature convergence.
- Our experiments reveal that EDO is able to automatically maintain a good balance between exploration and exploitation – through probabilistically performing random-move behaviors that help maintain the population diversity.
- At the same time, the ability to automatically adapt the search step size has been proven to be very useful.

Remarks on AOC by Self-Discovery

- The process of trial-and-error in AOC-by-self-discovery is automated by having one autonomous entity to control or fine-tune the parameters of other autonomous entities.
- Systems parameters are self-adapted according to some performance feedback.
- The EDO example described shows that AOC-by-self-discovery is a viable approach.

The AOC-by-Self-Discovery Approach



The Indiscernible Ingredients of AOC

- A population of autonomous entities;
- A behavior model for the autonomous entities;
- A model of local interactions between entities and their environment;
- A definition of roles and responsibilities of the environment;
- A set of criteria for measuring and self-organizing the nonlinear behavior of AOC problem solving.

Autonomous Behavior is Local

Primitive behaviors based on some limited amount of information observable from the limited range surroundings are sufficient for solving a global constraint problem.

Primitive Behavior is Neighbor-Driven

Controlling the direction of entity propagation can help nonlinearly amplify a desirable pixel labeling behavior and produce more entities of this locally successful entity type. The behavior of an autonomous entity does not need to be driven by some global information in order to produce a desirable global behavior.

Autonomous Behavior is Self-Directed

The Web foraging entities have two measures associated with them: motivational support and interest profile. The values of these two measures directly affect how an entity picks its next move.

Stochastic Behavior is Beneficial

- All the AOC algorithms described in this book share a common feature – the presence of stochastic behavior.
- For example, the ERE model has a random-move behavior, which randomly chooses a new position (for n-queen problems) or a new variable value (for SAT problems).
- Similarly, the entity in EDO has a choice to select a random-move behavior when it has not been making progress for some time.
- Stochastic behaviors enable an autonomous entity to get out of local minima. As a result, an AOC algorithm will have a better chance of locating a potentially better solution in the search space.

Some Behaviors can be Self-Reinforcing

- A search algorithm must find an appropriate step size in order to perform the search efficiently. However, a one-for-all step size is bound to err as the search landscape varies drastically among different sites.
- The step size adaptation behavior is, therefore, implemented and it considers the degree of success in locating a locally better solution as an indication of whether a bigger or smaller step size is required.
- By including such a behavior adaptation strategy, one can take a lot of guess work away from the trial-and-error process that is normally required in an AOC-by-prototyping method.

Separating the Good from the Poor Solutions is Required

- Population explosion is probably one of the most troublesome problems in an AOC algorithm.
- While better solutions are always welcome and seen as stepping stones to an even better solution, it is not advisable to keep replicating them indiscriminately.

Exchange of Information is Important but can be Minimal

- While information sharing is crucial to the success of the EDO algorithm, the amount of information flow can be minimal. In the presence of spatially distributed computing resources, this feature becomes very important.
- Autonomous entities in EDO limit their communications indirectly to those that share the same starting point (or parent). While the parent uses all the information fed by its offspring entities to bias its choice of behavior, the entities communicate with their respective parent to obtain the current best solution.
- In other words, all the autonomous entities in the search space are affecting each other indirectly.

Self-Organization is the Key to Success

- The centerpiece of an AOC algorithm is the notion of self-organization.
- When autonomous entities with self-directed behaviors are allowed to aggregate and react to the information and stimulation of other autonomous entities, a desired global behavior emerges as a result.
- The success in building a model with such an emergent behavior helps researchers explain Web surfing behavior. Similar complex systems modeling methods can be adopted in other studies, such as stock investor behavior or car driving behavior analysis.

- New approaches to systems dynamics and performance measurements are particularly needed so that clearer guidelines can be developed to help practitioners gain better insights into AOC, and AOC-by-self-discovery in particular.
- The measurements of emergence, evolvability, self-organization, tractability, and scalability in AOC are useful for tracking the progress of AOC.
- Theories on the formation of roles and social structures in a community of autonomous entities would expand the capability of an AOC system.
- Benchmark AOC problems should be identified for the purpose of comparing with other multi-entity paradigms.

- To foster and encourage the adoption of AOC for problem solving and complex systems modeling, more real-world applications as well as the characterization of potential areas need to be identified.
- More guidelines and tools for developing AOC are needed so that people can readily benefit from this new computing paradigm.
- The requirements for simulation environments, languages, and data structures in AOC need to be studied so that a more efficient implementation of AOC can result.
- Other implementation issues that need to be addressed include: architecture, visualization of activities, and the design of local and global nonlinear interaction rules.

Hardware and Software Environments

- AOC usually involves a large number of autonomous entities and static or dynamical environments.
- Implementing an AOC system in a single processor machine requires the support of virtual parallelism.
- Modern operating systems and programming languages support multi-thread technique. This allows slicing up CPU cycles and allocates them to the individual processes that represent components in the system.

- When multiple processors are available, individual components can be allocated to different processors on the same machine or across the network of processors with the support of some facilities, such as parallel virtual machine (PVM) [Geist et al., 1994, PVM, 1989] and message passing interface (MPI) [Snir et al., 1996, MPI, 1996]. However, it requires a central control program to coordinate task allocation and result consolidation.
- This makes parallel implementation of, for example, genetic algorithms possible without requiring expensive parallel machines.
- With peer-to-peer and grid computing networks, mobile entities can be sent to run on any machines over the Internet, provided that permission is granted by the host. Running simulated evolution in this way has been attempted [Smith and Taylor, 1998].

Update Schedule

- An individual entity changes its state at each step based on its current state and its neighboring environment.
- In a synchronous update scenario, the current state of all individuals is frozen to allow all individuals to obtain state information and change states, if appropriate.
- In a parallel implementation, synchronization may become an overhead too big to handle.
- Alternatively, asynchronous updates that allow processes on each processor to proceed independently may be implemented.
- The choice of an update schedule and the choice of a hardware platform are related. If a multi-process hardware environment is chosen, synchronous updates would slow the simulation down as all processes have to start and stop at the same time.

Management Services

- With the vast number of autonomous entities in an AOC system, AOC needs to keep track of the creation and deletion of objects.
- Moreover, a messaging mechanism is needed to facilitate message passing between objects.
- A central clock is also required to help the autonomous entities manage their state updates, no matter if it is synchronous or asynchronous.
- Some centralized whiteboards may also be needed if the autonomous entities are to share information in an implicit way and to contain a global view of the system's status.
- The whiteboards may also be used to simulate a dynamical environment in such systems as the ant system [Dorigo et al., 1996].

Visualization

- Visualization is a good way for people running simulations to 'see' what is going on with the experiment.
- Items of interest that are related to individual autonomous entities include actual movements, states, actions taken, fitness, ages, etc.
- Some global information (such as population size, best fitness, and average fitness) and some progress measurements (such as measurements of emergence, evolvability, diversity, and convergence) are also of interest to modelers.
- The visual display of such information will be of tremendous help to modelers to obtain a quick view of the system.

- We have presented a new computing paradigm, called autonomy oriented computing (AOC), based on the notions of autonomy and self-organization in entities.
- AOC is intended to meet the demands of real-world computing that is naturally embodied in large-scale, highly distributed, locally interacting entities, as in sensor networks, grid computing, and amorphous computing.
- Nevertheless, as we have demonstrated through examples, AOC is also applicable to tackling conventional computing problems.

Three Approaches

1. AOC-by-fabrication: some more or less known complex systems phenomena are abstracted and replicated in problem solving or system modeling.
2. AOC-by-prototyping: a trial-and-error approach to finding explanations to some complex behavior observations via autonomy oriented systems prototyping.
3. AOC-by-self-discovery: an autonomous problem solving approach that can fine-tune its own settings to suit the problem at hand.

Acknowledgments

We want to acknowledge the support of the following research grants:

1. Hong Kong Research Grant Council (RGC) Central Allocation Grant (HKBU 2/03/C) and Earmarked Research Grants (HKBU 2121/03E)(HKBU 2040/02E),
2. Hong Kong Baptist University Faculty Research Grants (FRG),
3. National Grand Fundamental Research 973 Program of China (2003CB317001), and
4. Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology (KP0705200379).

- J. Liu, X. Jin, and K. C. Tsui. *Autonomy Oriented Computing (AOC): From Problem Solving to Complex Systems Modeling.* Springer, 2005.
- 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: Systems and Humans* (in press).
- J. Liu, S. Zhang, and J. Yang. Characterizing Web Usage Regularities with Information Foraging Agents. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):566-584, 2004.
- J. Liu, J. Han, and Y. Y. Tang. Multi-Agent Oriented Constraint Satisfaction *Artificial Intelligence*, 136(1):101-144, 2002.
- K. C. Tsui and J. Liu. Evolutionary Multi-Agent Diffusion Approach to Optimization. *International Journal of Pattern Recognition and Artificial Intelligence*, World Scientific Publishing, 16(6):715-733, 2002.

- J. Liu and Y. Y. Tang. Adaptive Segmentation with Distributed Behavior Based Agents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):544-551, 1999.

References I


Adamic, L. A. (1999).
The small world Web.
In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries (ECDL'99)*, volume 1696 of *LNCS*, pages 443–452. Springer.


Adamic, L. A. and Huberman, B. A. (1999).
Technical comment to 'Emergence of scaling in random networks'.
Science, 286:509–512.


Adamic, L. A. and Huberman, B. A. (2000).
The nature of markets in the World Wide Web.
Quarterly Journal of Electronic Commerce, 1:5–12.


Adar, E. and Huberman, B. A. (2000).
The economics of surfing.
http://www.hpl.hp.com/research/idl/abstracts/ECommerce/econsurf.html.


Albert, R., Jeong, H., and Barabasi, A. L. (1999).
Diameter of World Wide Web.
Nature, 410:130–131.

References II

Arlitt, M. F. and Williamson, C. L. (1996).
Web server workload characterization: The search for invariants.
In Gaither, B. D., editor, *Proceedings of the ACM SIGMETRICS'96 Conference on Measurement and Modeling of Computer Systems*, pages 126–137. ACM Press.

Ashby, W. R. (1966).
Design for a Brain.
Chapman & Hall, 2nd edition edition.

Ashby, W. Ross (1947).
Principles of the self-organizing dynamic system.
Journal of General Psychology, 37:125–128.

Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (1997).
Handbook of Evolutionary Computation.
Oxford University Press.

Bak, P. (1996).
How Nature Works: The Science of Self-Organized Criticality.
Copernicus Books.

References III

Baluja, S. (1994).
Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning.
Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University.

Baluja, S. and Caruana, R. (1995).
Removing the genetics from the standard genetic algorithm.
In Prieditis, A. and Russel, S., editors, *Proceedings of the Twelfth International Conference on Machine Learning (ICML'95)*, pages 38–46. Morgan Kaufmann Publishers.

Barabasi, A. L. and Albert, R. (1999).
Emergence of scaling in random networks.
Science, 286:509–512.

Barabasi, A. L., Albert, R., and Jeong, H. (2000).
Scale-free characteristics of random networks: The topology of the World Wide Web.
Physica A, 281:69–77.

Barford, P., Bestavros, A., Bradley, A., and Crovella, M. (1999).
Changes in Web client access patterns: Characteristics and caching implications.
World Wide Web, 2:15–28.

References IV


Barford, P. and Crovella, M. (1998).
Generating representative Web workloads for network and server performance evaluation.
In Leutenegger, S., editor, *Proceedings of the ACM SIGMETRICS'98 Conference on Measurement and Modeling of Computer Systems*, pages 151–160. ACM Press.


Barták, R. (1998).
On-line guide to constraint programming.
http://kti.mff.cuni.cz/~bartak/constraints/stochastic.html.


Bitner, J. R. and Reingold, E. M. (1975).
Backtrack programming techniques.
Communications of the ACM, 18(11):651–656.


Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999).
Swarm Intelligence: From Natural to Artificial Systems.
Oxford University Press.


Bonabeau, E., Dorigo, M., and Theraulaz, G. (2000).
Inspiration for optimization from social insect behavior.
Nature, 406:39–42.

References V


Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1998).
Web caching and Zipf-like distributions: Evidence and implications.
Technical Report 1371, Computer Sciences Department, University of Wisconsin-Madison.


Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000).
Graph structure in the Web.
In Bulterman, D., editor, *Proceedings of the Ninth World Wide Web Conference (WWW9)*, pages 247–256.


Brooks, R. A. (1991).
Intelligence without representation.
Artificial Intelligence, 47:139–159.


Bruynooghe, M. (1981).
Solving combinatorial search problems by intelligent backtracking.
Information Processing Letters, 12(91):36–39.


Casti, J. (1997).
Would-Be Worlds: How Simulation is Changing the Frontiers of Science.
John Wiley & Son.

References VI


Cooper, M. C. (1989).
An optimal k-consistency algorithm.
Artificial Intelligence, 41:89–95.


Crovella, M. E. and Taqqu, M. S. (1999).
Estimating the heavy tail index from scaling properties.
Methodology and Computing in Applied Probability, 1(1):55–79.


Cuhna, C. R., Bestavros, A., and Crovella, M. E. (1995).
Characteristics of WWW client based traces.
Technical Report BU-CS-95-010, Computer Science Department, Boston University.


Davis, M., Logemann, G., and Loveland, D. (1962).
A machine program for theorem proofing.
Communications of the ACM, 5:394–397.


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

References VII

Durfee, E. H. (1999).
Distributed problem solving and planning.
In Weiss, G., editor, *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 121–164. MIT Press.

Ferber, J. (1999).
Multi-agent Systems: An Introduction to Distributed Artificial Intelligence, chapter 1, pages 31–35.
Addison-Wesley.

Gaschnig, J. (1979).
Performance Measurement and Analysis of Certain Search Algorithm.
PhD thesis, Department of Computer Science, Carnegie-Mellon University.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994).
PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing.
MIT Press.

Gent, I. P. and Walsh, T. (1993).
Towards an understanding of hill-climbing procedures for SAT.
In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 28–33. AAAI Press/MIT Press.

References VIII


Gent, I. P. and Walsh, T. (1995).
Unsatisfied variables in local search.
In Hallam, J., editor, *Hybrid Problems, Hybrid Solutions*, pages 73–85. IOS Press.


Glassman, S. (1994).
A caching relay for the World Wide Web.
Computer Networks and ISDN Systems, 27(2):165–173.


Goss, S., Beckers, R., Deneubourg, J. L., Aron, S., and Pasteels, J. M. (1990).
How trail laying and trail following can solve foraging problems for ant colonies.
In Hughes, R. N., editor, *Behavioral Mechanisms of Food Selection*, volume G20 of *NATO-ASI*, pages 661–678. Springer.


Gu, J. (1992).
Efficient local search for very large-scale satisfiability problem.
SIGART Bulletin, 3:8–12.


Gutowitz, H. (1991).
Cellular Automata: Theory and Experiment.
MIT Press.

References IX


Haken, H. (1983a).
Advanced Synergetics : Instability Hierarchies of Self-Organizing Systems and Devices.
Springer.


Haken, H. (1983b).
Synergetics: An Introduction. Nonequilibrium Phase Transition and Self-Organization in Physics, Chemistry, and Biology.
Springer, third revised and enlarged edition edition.


Haken, H. (1988).
Information and Self-Organization : A Macroscopic Approach to Complex Systems.
Springer.


Han, C. C. and Lee, C. H. (1988).
Comments on Mohr and Henderson's path consistency algorithm.
Artificial Intelligence, 36:125–130.


Helbing, D., Huberman, B. A., and Maurer, S. M. (2000).
Optimizing traffic in virtual and real space.
In Helbing, D., Herrmann, H. J., Schreckenberg, M., and Wolf, D. E., editors, *Traffic and Granular Flow '99: Social, Traffic, and Granular Dynamics*. Springer.

References X

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

Hoos, H. H. and Stützle, T. (1999).
Systematic vs. local search for SAT.
In *Proceedings of KI-99*, volume 1701 of *LNAI*, pages 289–293. Springer.

Horst, R. and Pardalos, P. M., editors (1995).
Handbook of Global Optimization.
Kluwer Academic Publishers.

Horst, R. and Tuy, H. (1990).
Global Optimization: Deterministic Approaches.
Springer.

Huberman, B. A., editor (1988).
The Ecology of Computation.
North-Holland.

References XI


Huberman, B. A. and Adamic, L. A. (1999).
Evolutionary dynamics of the World Wide Web.
Nature, 399:131.


Huberman, B. A., Pirolli, P. L. T., Pitkow, J. E., and Lukose, R. M. (1997).
Strong regularities in World Wide Web surfing.
Science, 280:96–97.


IEEE (2001).
Draft standard for information technology, learning technology glossary.
Technical Report P1484.3/D3, IEEE.


IEEE (2002).
IEEE standard for learning object metadata.
Technical Report P1484.12.1, IEEE.


Jensen, H. J. (1998).
Self-Organized Criticality: Emergent Complex Behavior in Physical and Biological Systems.
Cambridge University Press.


Johansen, A. and Sornette, D. (2000).
Download relaxation dynamics on the WWW following newspapers publication of URL.
Physica A, 276:338–345.

References XII


Kauffman, S. (1993).
Origins of Order: Self-Organization and Selection in Evolution.
Oxford University Press.


Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983).
Optimization by simulated annealing.
Science, 220:671–680.


Kuhnel, R. (1997).
Agent oriented programming with Java.
In Plander, I., editor, *Proceedings of the Seventh International Conference on Artificial Intelligence and Information – Control Systems of Robots (AIICSR'97)*. World Scientific Publishing.


Kumar, V. (1992).
Algorithm for constraint satisfaction problem: A survey.
AI Magazine, 13(1):32–44.


Levene, M., Borges, J., and Loizou, G. (2001).
Zipf's law for Web surfers.
Knowledge and Information Systems, 3:120–129.

References XIII


Levene, M. and Loizou, G. (1999).
Computing the entropy of user navigation in the Web.
Technical Report RN/99/42, Department of Computer Science, University College London.


Liu, J. (2001).
Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation.
World Scientific Publishing.


Liu, J. and Han, J. (2001).
ALIFE: A multi-agent computing paradigm for constraint satisfaction problems.
International Journal of Pattern Recognition and Artificial Intelligence, 15(3):475–491.


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


Liu, J. and Wu, J. (2001).
Multi-Agent Robotic Systems.
CRC Press.

References XIV

Loser, A., Grune, C., and Hoffmann, M. (2002).
A didactic model, definition of learning objects and selection of metadata for an online curriculum.
http://www.ibi.tu-berlin.de/diskurs/onlineduca/onleduc02/Talk_Online_Educa__02_Loeser_TU_berlin.pdf.

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

Lucas, C. (1997).
Self-organizing systems (SOS).
http://www.calresco.org/sos/sosfaq.htm.

Lukose, R. M. and Huberman, B. A. (1998).
Surfing as a real option.
In *Proceedings of the First International Conference on Information and Computation Economics (ICE'98)*, pages 45–51.

References XV


Mackworth, A. K. (1977).
Consistency in networks of relations.
Artificial Intelligence, 8(1):99–118.


Maurer, S. M. and Huberman, B. A. (2000).
The competitive dynamics of Web sites.
http://ideas.repec.org/p/sce/scecf0/357.html.


Mazure, B., Sais, L., and Grégoire, É. (1997).
Tabu search for SAT.
In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 281–285.


McAllester, D., Selman, B., and Kautz, H. (1997).
Evidence for invariants in local search.
In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)*, pages 321–326.


Milgram, S. (1967).
The small world problem.
Psychology Today, 2:60–67.

References XVI


Mockus, J. (1989).
Bayesian Approach to Global Optimization : Theory and Applications.
Kluwer Academic Publishers.


Mohr, R. and Henderson, T. C. (1986).
Arc and path consistency revisited.
Artificial Intelligence, 28:225–233.


Montoya, J. M. and Sole, R. V. (2000).
Small world patterns in food webs.
http://arxiv.org/abs/cond-mat/0011195.


MPI (1996).
http://www-unix.mcs.anl.gov/mpi/.


Müller, S. D., Marchetto, J., Airaghi, S., and Koumoustsakos, P. (2002).
Optimization based on bacterial chemotaxis.
IEEE Transactions on Evolutionary Computation, 6(1):16–29.


Nehaniv, C. L. (2000).
Preface.
In *Nehaniv [?]*, pages iii–iv.

References XVII

Nicolis, G. and Prigogine, I. (1977).
Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order through Fluctuations.
John Wiley.

Prigogine, I. (1980).
From Being to Becoming: Time and Complexity in the Physical Sciences.
W. H. Freeman and Company.

PVM (1989).
http://www.epm.ornl.gov/pvm/.

Rossi, F., Petrie, C., and Dhar, V. (1990).
On the equivalence of constraint satisfaction problem.
In *Proceedings of the Ninth European Conference on Artificial Intelligence (ECAI-90)*, pages 550–556.

Sandholm, T. W. (1999).
Distributed rational decision making.
In Weiss, G., editor, *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. MIT Press.

References XVIII

Selman, B., Kautz, H., and Cohen, B. (1994).
Noise strategies for improving local search.
In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94)*, pages 337–343.

Selman, B., Levesque, H., and Mitchell, D. (1992).
A new method of solving local search.
In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446.

Shanahan, M. (1994).
Evolutionary automata.
In *Artificial Life IV: Proceedings of the Fourth International Workshop Synthesis and Simulation of Living Systems*, pages 387–393. MIT Press.

Shoham, Y. (1993).
Agent oriented programming.
Artificial Intelligence, 60(1):51–92.

Silaghi, M., Haroud, D., and Faltings, B. (2001a).
ABT with asynchronous reordering.
In *Proceedings of the International Conference on Intelligent Agent Technology (IAT'01)*, pages 54–63.

References XIX


Silaghi, M., Haroud, D., and Faltings, B. (2001b).
Asynchronous consistency maintenance.
In *Proceedings of the International Conference on Intelligent Agent Technology (IAT'01)*, pages 98–102.


Silaghi, M., Haroud, D., and Faltings, B. (2001c).
Secure asynchronous search.
In *Proceedings of the International Conference on Intelligent Agent Technology (IAT'01)*, pages 400–404.


Smith, R. E. and Taylor, N. (1998).
A framework for evolutionary computation in agent based systems.
In *Proceedings of the 1998 International Conference on Intelligent Systems*, pages 221–224. ISCA Press.


Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. (1996).
MPI: The Complete Reference.
MIT Press.


Sosic, R. and Gu, J. (1994).
Efficient local search with conflict minimization: A case study of the n-queen problem.
IEEE Transactions on Knowledge and Data Engineering, 6(5):661–668.

References XX

Stallman, R. and Sussman, G. J. (1977).
Forward reasoning and dependency directed backtracking.
Artificial Intelligence, 9(2):135–196.

Steinmann, O., Strohmaier, A., and Stützle, T. (1997).
Tabu search vs. random walk.
In *Advances in Artificial Intelligence (KI97)*, volume 1303 of *LNCS*, pages 337–348. Springer.

Storn, R. and Price, K. (1997).
Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces.
Journal of Global Optimization, 11(4):341–359.

Swarm (1994).
An overview of the Swarm simulation system.
http://www.santafe.edu/projects/swarm/swarm-blurb/swarm-blurb.html.

Torn, A. and Zilinskas, A. (1989).
Global Optimization.
Springer.

References XXI


Ünsal, C. (1993).
Self-organization in large populations of mobile robots.
Master's thesis, Department of Electrical Engineering, Virginia Polytechnic Institute and State University.
http://www-2.cs.cmu.edu/~unsal/thesis/cemsthesis.html.


Walsh, T. (1999).
Search in a small world.
In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1172–1177.


Watts, D. J. and Strogatz, S. H. (1998).
Collective dynamics of small world networks.
Nature, 393:440–442.


Yan, T. W., Jacobsen, M., Garcia-Molina, H., and Dayal, U. (1996).
From user access patterns to dynamic hypertext linking.
In *Proceedings of the Fifth World Wide Web Conference (WWW5)*, pages 1007–1014.


Yao, X. and Liu, Y. (1997).
Fast evolution strategies.
Control and Cybernetics, 26(3):467–496.

References XXII


Yao, X., Liu, Y., and Lin, G. (1999).
Evolutionary programming made faster.
IEEE Transaction on Evolutionary Computation, 3(2):82–102.


Yokoo, M. (1995).
Asynchronous weak-commitment search for solving large-scale distributed CSPs.
In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 467–518.


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


Yokoo, M., Etzioni, O., Ishida, T., Jennings, N., and Sycara, K. (2001).
Distributed Constraint Satisfaction Foundations of Cooperation in Multi-Agent Systems.
Springer.


Yokoo, M. and Hirayama, K. (1998).
Distributed constraint satisfaction algorithm for complex local problems.
In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 372–379.

References XXIII

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

Yokoo, M. and Kitamura, Y. (1996).
Multi-agent real-time-A* with selection: Introducing competition in cooperative search.
In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS'96)*, pages 409–416.

Zipf, G. K. (1949).
Human Behavior and the Principle of Least Effort.
Addison-Wesley.