Chapter 2

AOC at a Glance

2.1. Introduction

AOC approaches share a basic form with many possible variants. As mentioned in the preceding chapter, the basic form draws on the core notion of autonomy, with such characteristics as multi-entity formulation, local interaction, nonlinear aggregation, and self-organized computation. In order to get a better idea on how these characteristics are reflected in handling some familiar computational or engineering problems, let us now take a look at three illustrative examples: constraint satisfaction, image feature extraction, and robot spatial learning (or world modeling). In our illustrations, we will outline the basic autonomy models implemented and highlight the AOC systems performance.

2.2. Autonomy Oriented Problem Solving

Our first illustrated example deals with distributed problem solving. Liu et al. have developed an AOC-based method for solving CSPs [Liu and Han, 2001, Liu et al., 2002]. This method is intended to provide an alternative, multi-entity formulation that can be used to handle general CSPs and to find approximate solutions without too much computational cost.

2.2.1 Autonomy Oriented Modeling

In the AOC-based method, distributed entities represent variables and a two-dimensional lattice-like environment in which entities inhabit corresponds to the domains of variables. Thus, the positions of entities in such an environment constitute a possible solution to the corresponding CSP. The distributed multi-entity system self-organizes itself, as each entity follows its behavioral rules, and gradually evolves towards a global solution state.

Based on two general principles of 'survival of the fittest' – poor performers will be washed out, and 'law of the jungle' – weak performers will be eliminated by stronger ones, the AOC-by-fabrication approach is applied to solve a benchmark constraint satisfaction problem [Han et al., 1999, Liu and Han, 2001].

2.2.2 N-Queen Problem

The n-queen problem aims to allocate n queens on an $n \times n$ chessboard such that no two queens are placed within the same row, column, and diagonal. Based on the constraints of the problem, a model is formulated in the following manner. Each queen is modeled as an autonomous entity in an AOC system and multiple queens are assigned to each row on the chessboard. This is to allow competition among the queens in the same row such that the queen with the best strategy survives. The system calculates the number of violated constraints (i.e., violations) for each position on the chessboard. This represents the environmental information to all queens in making movement decisions, which are restricted to positions in the same row. Queens are allowed for three types of movement. A 'randomized-move' allows a queen to randomly select a new position. A 'least-move' selects a position with the least number of violations. A 'coop-move' promotes cooperation between queens by excluding positions that will attack those queens with which one wants to cooperate. These types of movement are selected probabilistically.

An initial energy is given to each queen. A queen will 'die' if its energy falls below a predefined threshold. Energy will change in two ways. When a queen moves to a new position that violates the set constraint with m queens, it loses m units of energy. This will also cause those queens that attack this new position to lose one unit of energy. The intention is to encourage a queen to find a position with the least number of violations. The 'law of the jungle' principle is implemented by having two or more queens occupying the same position to compete for the occupancy. The queen with the highest energy will win and eliminate the loser(s) by absorbing all the energy of the loser(s).

The above model is able to efficiently solve n-queen problems with up to 7,000 queens using a moderate hardware configuration. Experimental results show that the 'survival of the fittest' principle helps find an optimal solution much more quickly due to the introduction of competition. The randomized-move is indispensable as it helps an AOC system come out of local minima, although giving a high chance of making a randomized-move will lead to chaotic behavior. The probabilities of selecting 'least-move' and 'coop-move' should be comparable and increased with the size of a problem.

2.3. Autonomy Oriented Search

In our next example, let us consider the following search problem: An environment contains a homogeneous region with the same physical feature. This region is referred to as a goal region. The feature of the goal can be evaluated based on some measurements. Here, the term 'measurement' is taken as a generic notion. The specific quantity that it refers to depends on the nature of applications. For instance, it may refer to the grey level intensity of an image in the case of image processing. The task of autonomous entities 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.

2.3.1 Autonomy Oriented Modeling

In the AOC-based method, an entity checks its neighboring environment, i.e., small circles as in Figure 2.1(a), and selects its behavior according to the concentration of elements in the neighboring region. If the concentration is within a certain range, the current location satisfies a triggering condition. This activates the reproduction mechanism of the entity.

Taking a border tracing entity for example (see Figure 2.1(a)), if an entity of the border sensitive class reaches a border position, this entity will inhabit at the border and proceed to reproduce both within its immediate neighboring region and inside a large region, as illustrated in Figures 2.1(b) and (c).

2.3.2 Image Segmentation Problem

Image segmentation requires to identify homogeneous regions within an image. However, homogeneity can be at varying degrees at different parts of the image. This presents problems to conventional methods, such as split-and-merge that segments an image by iteratively partitioning heterogeneous regions and simultaneously merging homogeneous ones [Pavlidis, 1992, Pitas, 1993]. An autonomy oriented method has been developed to tackle the same task [Liu et al., 1997]. Autonomous entities are deployed to the two-dimensional representation of an image, which is considered as the search space of entities. Each entity is equipped with an ability to assess the homogeneity of a region within a predefined locality. Specifically, homogeneity is defined by the relative contrast, regional mean, and region standard deviation of the grey level intensity. When 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.

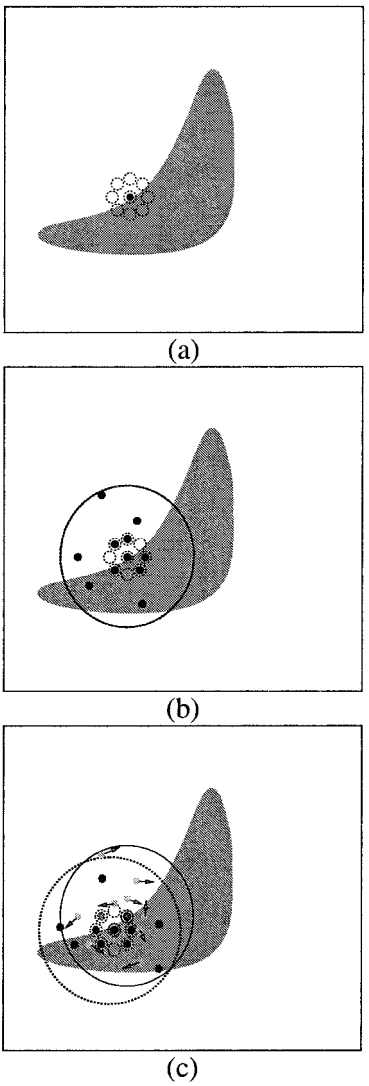

Figure 2.1. An illustration of the behavior of autonomous entities. (a) As an entity, which is marked as a solid circle, moves to a new location, it senses its neighboring locations, marked by dotted circles in this example. Specifically, it counts the number of locations at which the grey level intensity is close to that of the entity's current location. (b) 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. (c) 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].

Through breeding behavior, an entity distributes its newly created offspring into the region that is found to be homogeneous, so that the offspring entities are more likely to find extensions to the current homogeneous region. Apart from breeding, an entity will also label an pixel that is found to be in a homogeneous region. If an autonomous entity fails to find a homogeneous region during its lifespan (a predefined number of steps) or wanders off the search space during diffusion, it will be marked as an inactive entity.

In summary, the stimulus from pixels will direct autonomous entities to two different behavioral tracts: breeding and pixel labeling, or diffusion and decay. The directions of breeding and diffusion are determined according to their respective behavioral vectors, which contain weights (between 0 and 1) of all possible directions. The weights are updated by considering the number of successful siblings in the respective directions. An entity is considered to be successful if it has found one or more pixels that are within a homogeneous region. This method of direction selection is somewhat similar to herding behavior that only considers local information. A similar technique has been applied to feature extraction tasks, such as border tracing and edge detection [Liu and Tang, 1999]. A more difficult task where an image contains different homogeneous regions has been successfully handled by deploying autonomous entities with different homogeneity criteria.

2.3.3 An Illustrative Example

In order to examine the above autonomy oriented method in the simultaneous detection of significant image segments, Liu et al. [Liu, 2001, Liu et al., 1997, Liu and Tang, 1999] have conducted several experiments in which various classes of entities are defined and employed to extract different homogeneous regions from an image, such as the example given in Figure 2.2 ($t = 0$). For this image segmentation task, 1,500 entities, evenly divided into three classes, are randomly distributed over the given image. Figure 2.2 presents a series of intermediate steps during the collective image segmentation. Figure 2.2 ($t = 50$) gives the resultant markers as produced by the different classes of entities.

2.3.4 Computational Steps

In the AOC-based image segmentation, the computational steps required can be estimated by counting how many active entities are being used over time (i.e., the entities whose ages do not exceed a given life span). For the above mentioned collective image segmentation task, we have calculated the number of active entities in each class that have been involved over a period of 50 steps, as given in Table 2.1. It can readily be noted that the total number of

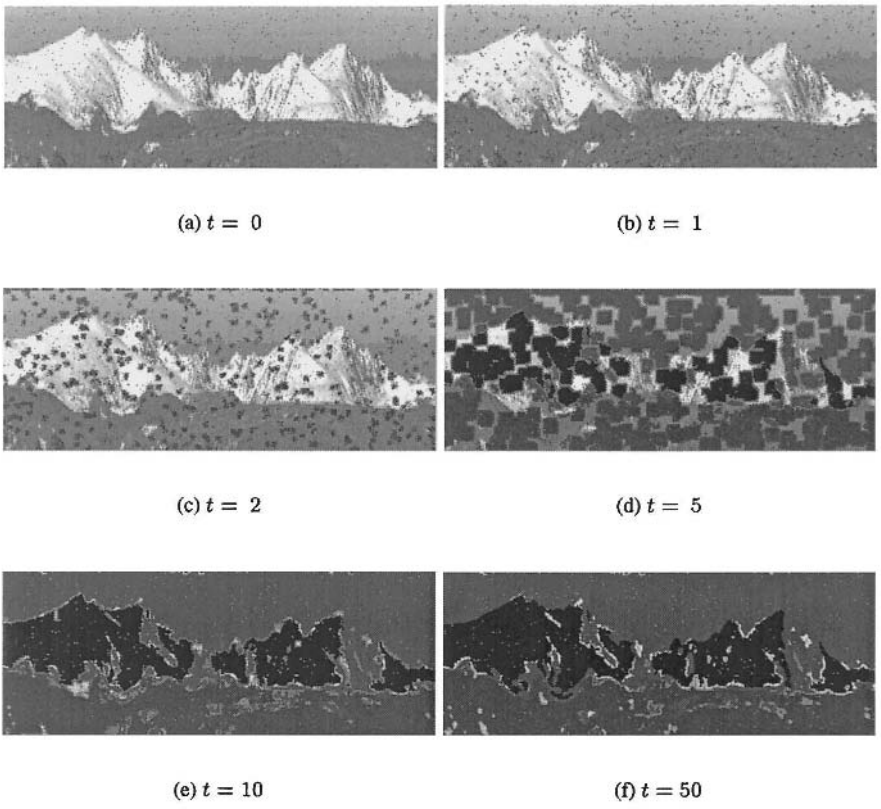

Figure 2.2. Segmenting a landscape image that contains three complex-shaped homogeneous regions [Liu, 2001].

active entities (i.e., computational steps) involved in extracting a homogeneous region is less than the size of the given image, $526 \times 197 = 103,622$.

Table 2.1. The number of active entities involved in extracting the homogeneous regions of a landscape image.

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

2.4. Autonomy Oriented Learning

Our final example demonstrates how AOC can be embodied in a group of distributed autonomous robots to perform a task of collective spatial learning (or world modeling).

Ant colonies are able to collect objects, such as food or dead ants, and place them in particular places. Collective behavior in a complex system offers the possibilities of enhanced task performance, increased task reliability, and decreased computational cost over traditional complex systems. Much work to date in collective robotics focuses on limited cases, such as flocking and foraging. Typical entities in those studies either use manually built (non-learning) controllers [Balch and Arkin, 1995], or perform a learning task in simulated [Balch, 1997] or relatively simple physical environments [Mataric, 1994]. One way to generate robust collective behavior is to apply biologically inspired adaptive algorithms at a team level. In such a case, the environment plays a central role in triggering a certain basic behavior at any given time. It draws on the idea of providing robots with a range of primitive behaviors and letting the environment determine which behavior is more suitable as a response to a certain stimulus. The integration of learning methods can significantly contribute to the performance of a team of self-programming robots for some predefined tasks. These individual robots can automatically program their task-handling behavior to adapt to dynamical changes in their task environment in a collective manner.

2.4.1 World Modeling

Liu and Wu have developed an AOC-based method for 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, that is, to construct a potential field map as efficiently as possible. The following issues are addressed in developing the proposed world modeling method:

- How to formally define and represent the reactive behavior of mobile robots and the underlying adaptation mechanisms to enable the dynamical acquisition of collective behavior?
- How to solve the problem of collective world modeling (i.e., potential field map building) in an unknown robot environment based on self-organization principles?

The artificial potential field (APF) theory states that for any goal directed robot in an environment that contains stationary or dynamically moving obstacles, an APF can be formulated and computed by taking into account an attractive pole at the goal position of the robot and repulsive surfaces of the

obstacles. Using APF, any dynamical changes in the environment can be modeled by updating the original artificial potential field. With APF, a robot can reach a stable configuration in its environment by following the negative gradient of its potential field.

An important challenge in the practical applications of the APF methodology is that evolving a stable APF is a time consuming learning process, which requires a large amount of input data coming from the robot-environment interaction. The distributed self-organization method for collective APF modeling with a group of mobile robots begins with the modeling of local interactions between the robots and their environment, and then applies a global optimization method for selecting the reactive motion behavior of individual robots in an attempt to maximize the overall effectiveness of collectively accomplishing a task.

The main idea behind self-organization based collective task handling is that multiple robots are equipped with a repository of behavioral responses in such a way as to create some desirable global order, e.g., the fulfillment of a given task. For instance, mobile robots may independently interact with their local environment. Based on their performance (e.g., distributed proximity sensory measurements), some global world models of an unknown environment (i.e., global order) can be dynamically and incrementally self-organized.

2.4.2 Self-Organization

In the case of collective world modeling, self-organization is carried out as follows: Suppose that a robot moves to position p_0 and measures its distances to the surrounding obstacles of its environment in several directions (n). These measurements are recorded 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 denotes the distance between position p_0 and an obstacle sensed in the ith direction. The robot will then associate this information to its adjacent positions in the environment by estimating the proximity values in the neighboring positions. The estimated proximity of any position p_j inside the neighboring region of p_0 to a sensed obstacle will be calculated as follows: $\hat{d}_i^j = d_i^0 - \rho_j \cdot cos\beta \quad (i = 1, 2, \cdots, n)$, where $\beta = \alpha_0^{(i)} - \alpha_j$. $\alpha_0^{(i)}$ and α_j denote the polar angle of the sensing direction and that of position p_j, respectively. $\hat{d}_i^j$ is an estimate for p_j based on the ith direction sensing value. d_i^0 is the current measurement taken from p_0 in the ith direction. Thus, the estimated proximity values for position p_j can be written as: $\hat{\mathcal{S}}_j = \left[\hat{d}_1^j, \hat{d}_2^j, \cdots, \hat{d}_i^j, \cdots, \hat{d}_n^j\right]$. Figure 2.3 illustrates the distance association scheme.

Next, we define a confidence weight for each element of $\hat{\mathcal{S}}_j$, that is, a function of the distance between a robot and position p_j, or specifically, $w_j =$

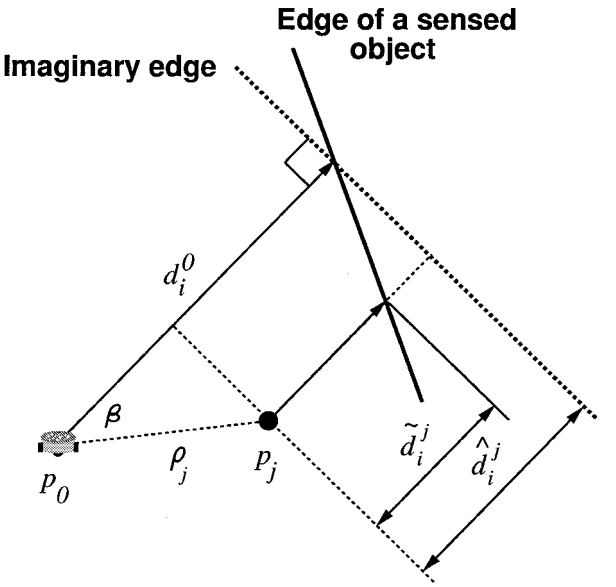

Figure 2.3. An illustration of the distance association scheme [Liu, 2001].

$e^{-\eta\rho_j^2}$, where η is a positive constant; ρ_j is the distance between the robot and position p_j.

The potential field estimate at position p_j is then computed as follows:

$$\hat{U}_j^t = \sum_{i=1}^{n} e^{-\lambda \hat{d}_i^j}, \tag{2.1}$$

where λ is a positive constant. Thus, 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, that is,

$$\Omega_t^j \longleftarrow \Omega_{t-1}^j \bigcup \mathcal{Q}, \tag{2.2}$$

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: $W_t^j = \{w_j^{t_1}, w_j^{t_2}, \cdots, w_j^{t_i}, \cdots, w_j^{t_k}\}$.

Hence at time t, an acceptable potential field value can readily be calculated as follows:

$$U_j^t = \begin{cases} \hat{U}_j^{t_i}, & \exists i \in [1,k], w_j^{t_i} = 1, \\ \sum_{i=1}^{k} \hat{U}_j^{t_i} \cdot \bar{w}_j^{t_i}, & \text{otherwise}, \end{cases} \tag{2.3}$$

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

$$\bar{w}_j^{t_i} = \frac{w_j^{t_i}}{\sum_{n=1}^{k} w_j^{t_n}}. \tag{2.4}$$

2.4.3 Adaptation

In order to optimize the efficiency of the above self-organization based collective world modeling, we need an adaptation mechanism for distributed autonomous robots to dynamically generate and 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.

In order to evaluate the group fitness, we identify two situations involved in the evolution: One is spatial diffusion when the inter-distance between robots i and j, σ_{ij}, is less than or equal to a threshold, η, and the other is area coverage when $\sigma_{ij} > \eta$. In either situation, we can use a unified direction representation of robot proximity, denoted by θ_i to indicate a significant proximity direction of all proximity stimuli to robot i. Having identified these two situations in group robots, we can reduce the problem of behavior evolution into that of acquiring two types of individual reactive motion behavior: One for spatial diffusion and the other for area coverage, respectively. Both types of reactive behavior respond to proximity stimuli as defined in terms of a unified significant proximity direction.

The fitness function will consist of two terms: One is called general fitness, denoted by f_g, and the other is called special fitness, denoted by f_s. The general fitness term encourages group robots to explore the potential field in new, less confident regions, and at the same time, avoid repeating the work of other robots. It is defined as follows:

$$f_g = \prod_{i=1}^{m} \left\{ (1 - \max\{w_i^{t_k}\}) \prod_{j=1}^{m_e} \sqrt[4]{\sigma_{ij} - \delta} \right\}, \tag{2.5}$$

where $\max\{w_i^{t_k}\}$ denotes the maximal confidence weight corresponding to the position of robot i. m denotes the number of robots that are grouped together during one evolutionary movement step (of several generations). m_e denotes the number of robots that do not belong to m and have just selected and executed their next behavior. σ_{ij} denotes the distance between robots i and j, which is greater than a predefined distance threshold, δ.

Two special fitness terms will be defined corresponding to the performance of spatial diffusion and area coverage:

$$\text{spatial\_diffusion: } f_{s1} = \prod_{i=1}^{m_d-1} \prod_{j=i+1}^{m_d} \sqrt{\sigma_{ij} - \eta}, \tag{2.6}$$

and

$$\text{area\_coverage: } f_{s2} = \frac{\sqrt{\Delta \mathcal{V}}}{\prod_{i=1}^{m_c} \zeta_i}, \tag{2.7}$$

where m_d denotes the number of spatially diffusing robots whose inter-distances σ_{ij} have become greater than the distance threshold, η. $\Delta \mathcal{V}$ denotes the total number of positions visited by a group of m_c area-covering robots based on their selected motion directions. ζ_i denotes a significant proximity distance between robot i and other robots in the environment.

2.5. Summary

So far, we have provided three illustrative examples: constraint problem solving, distributed search, and spatial learning. We have stated the basic problem requirements and showed the ideas behind the AOC solutions, ranging from their formulations to the emergence of collective solutions through self-organization.

From the illustrations, we can note that using an AOC-based method to solve a problem is essentially to build an autonomous system, which usually involves a group of autonomous entities residing in an environment. Entities are equipped with some simple behaviors, such as move, diffuse, breed, and decay, and one or more goals (e.g., to locate a pixel in a homogeneous region). In order to achieve their goals, entities either directly interact with each other or indirectly interact via their environment. Through interactions, entities accumulate their behavioral outcomes and some collective behaviors or patterns emerge. Ideally, these collective behaviors or patterns are what we are expecting, i.e., solutions to our problems at hand.

Exercises

2.1 Provide a conceptual blueprint for a potential AOC programming language that can support the applications as mentioned in this chapter. What will be its constructs? What will be the key characteristics and requirements of its operations?

2.2 In order to evaluate computing languages or environments for AOC, what will be your suggested criteria? What will be your suggested benchmark problems?

2.3 AOC offers a new way of tackling complexity, whether in problem solving or in complex systems modeling, by utilizing localized, autonomous and yet low-cost (computationally and physically speaking), and self-organized entities. Based on the illustrative examples given in this chapter, try to suggest and develop some other alternative models of AOC, as inspired by nature, for solving the same or different problems.

2.4 Identify from the above solved problems the real benefits of taking this route to complexity.

2.5 The example on world modeling in this chapter utilizes self-organized collective behavior in a multi-robot system to model an unknown environment. Think and propose a similar solution to the problem of multi-robot navigation.

2.6 Can you summarize the similarity in computational ideas between the image segmentation example and the world modeling example? (for instance, both have been treated as distributed problem solving).

2.7 The chapter illustrates a search (optimization based) strategy for image segmentation and feature detection. From a computer vision point of view, compare this strategy with some traditional search based segmentation algorithms, and evaluate their performances with other computer vision segmentation benchmarks.

http://www.springer.com/978-1-4020-8121-7

Autonomy Oriented Computing
From Problem Solving to Complex Systems Modeling
Liu, J.; Jin, X.; Tsui, K.C.
2005, XXXII, 216 p. 57 illus., Hardcover
ISBN: 978-1-4020-8121-7