Multiagent Diffusion and Distributed Optimization

Kwok Ching Tsui and Jiming Liu
Department of Computer Science
Hong Kong Baptist University
Kowloon Tong, Hong Kong
{tsuikc,jiming}@comp.hkbu.edu.hk

ABSTRACT

Distributed problem solving by a multiagent system represents a promising approach to solving complex computational problems. However, many multiagent systems require certain degree of planning, coordination and negotiation to achieve the given goal. This paper presents a multiagent framework for tackling global optimization tasks inspired by diffusion in nature. The framework is designed for situations where agent communication must be kept to a minimal. Hence, complicated coordination and negotiation is not possible. Distributed agents in this framework share the common goal of finding the global optimal solution. They cooperate to achieve this common goal by sharing and updating a common belief that captures their estimation of the whereabouts of the optimal solution. To facilitate this, agents are naturally organized in *families* with a parent and its offsprings as members. This paper also presents an algorithm called *Evolutionary Diffusion Optimization*, which is implemented base on the proposed agent framework. Experimental results on some benchmark problems are presented together with performance comparison with a simulated annealing algorithm.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: multiagent systems; I.2.8 [**Problem Solving, Control Methods, and Search**]: heuristic methods

General Terms

Algorithms, Experimentation, Theory

Keywords

Diffusion model, multiagent system, optimization,autonomy oriented computation

1. INTRODUCTION

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

$$\mathcal{F}(x^*) \leq \mathcal{F}(x) \quad \forall x \tag{1}$$

The search for x^* can be viewed as the minimization of function $\mathcal{F}$. Turning the sign in Equation 1 around makes the search for x^* a maximization task. They can collectively be called global optimization tasks [29].

Many algorithms have been developed over the years to tackle the challenging task of global optimization [14, 15, 23]. In the absence of prior knowledge about the search landscape, stochastic approaches, such as *simulated annealing* [18] and *population-based incremental learning* [3, 4], have been proved 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* [2, 12, 13, 27], *bacterial chemotaxis* [24], *differential evolution* [28], *particle swarm optimization* [17], *cultural algorithm* [26] and *ant colony optimization* [9].

There are several challenges any search algorithm must face. Firstly, the landscape of the function to be optimized is unknown. Unimodal functions can be monotonic in nature and the search is easy once the downhill direction is found. However, finding the direction of the search landscape is not a simple task. Multimodal functions, on the other hand, have many suboptimal solutions where a search algorithm is likely to be trapped.

Secondly, there is usually no linear relationship between changes made to the function variables and the corresponding change in the function value. This assignment of credit problem confuses, if not misleads, the search algorithm.

Thirdly, search algorithms do not normally jump directly to the optimal solution but making incremental changes in small steps instead. Making big steps is not always a better strategy, especially when the optimal solution is close by. In contrast, infinitesimal changes are detrimental to the effort to escape from a local optimum. Therefore, it is crucial to choose a step size appropriate to the situation prevalent during the search.

Fourthly, 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.

Lastly, the advent of grid and cluster computing technology, search and optimization tasks will not be restricted to a single machine or locally connected machines. For example, the aim of the European project Crossgrid [1] is to "develop, implement and exploit new Grid components for interactive compute and data intensive applications". It is essential when tackling large scale optimization problems to have a mechanism for computational entities running on different machines to collaborate without requiring extensive communication or planning.

This paper proposes an agent framework inspired by diffusion in natural and artificial systems. The framework consists of a society of agents where they cooperate to perform an optimization task. The challenge is that the agents have to search for the optimal solution in a distributed manner and in the absence of complete information about the search landscape; only a partial view is available. Moreover, there is no central planner that coordinates the agents during the search. Essentially, the agents have to collaborate in order to complete the task. However, minimal communication bandwidth is available.

There are several special aspects about the optimization agents:

- the agents communicates sparingly and selective based on the organizational structure;
- the agents are homogeneous in the sense that they have the same set of behaviors;
- all the agent's behaviors are local in nature - itself or agents in close neighborhood;
- the agents are rational only to certain degree, which means behavior selection does not strictly follow the same rule every time;
- there is no explicit division of labor for the search;
- the agents cooperative via sharing of belief rather than by negotiation or competition;
- the agents may not survive the whole search process if they cannot provide their worth.

1.1 Organization of This Paper

The next section will describe some observations that motivated the agent diffusion framework. This is followed by a formal description of the agent diffusion framework. An example algorithm that implemented the proposed framework will then be described together with some experimental results. This paper concludes with a summary of the proposed framework and a brief discussion of possible extensions.

2. BACKGROUND AND RELATED WORK

Diffusion in nature and the successful application of diffusion model to image segmentation [21] have inspired the multiagent diffusion framework, which attempts to tackle the task of optimizing multi-dimensional functions. This section describes in more detail a diffusion model observable from nature. Some research works on agent model and distributed problem solving will then be discussed in order to highlight the major innovations in the proposed framework. Lastly, a computational paradigm called *autonomy oriented computation* (AOC) [22] that is relevant to the current work will be introduced.

2.1 Human Migration Study

The study of human geography categorized reasons for human migration into *push* factors and *pull* factors [5, 7]. Push factors relate to undesirable conditions such as poor living conditions, lack of job and overcrowding. Pull factors are those positive factors that attracts people to relocate, such as jobs and better living conditions. Two forms of migration can also be identified. *Step migration* refers to a series of local movements, such as moving from village to town, then to city. *Chain migration* refers to a more drastic change beyond the local region and is usually assisted by people who have already emigrated. The availability of information seems to be an important factor that helps people to decide when and where to migrate.

The important lesson to learn for optimization is that once the landscape of the target function is known, finding the optimal solution becomes trivial. The question is how to capture the *trend*. A search algorithm will need past experience to inform it of the possible successful moves (push factor) and unsuccessful moves (pull factor). In addition, whoever has captured the trend of the search space can help others to make better moves by *sharing their experience*.

2.2 Agent Models and Distributed Optimization

Rao and Georgeff have proposed the most widely-used *Belief-Desire-Intention* (BDI) agent model [25]. *Belief* represents the state of the environment that an BDI agents roams. *Desire* represents the motivational aspect of an agent-oriented system. *Intention* represents the course of action taken by an agent. While the general principles for constructing an agent system have been explained with examples, the BDI architecture does not give any indication for multiagent interaction.

Chainbi *et. al.* picked up the interaction issue and proposed the *Belief-Goal-Role* (BGR) agent model [6], which stresses the fact that agents can have different local goals and roles in achieving a common global goal. In the context of optimization, we argue that agents have the common goal of finding the global solution. Hence, global and local goals are indistinguishable. However, this does not preclude the agents from playing different roles (two in our case) during the search. In fact, having agents to assume one of the two roles is a crucial feature our framework that facilitates cooperative collaboration.

Liu *et. al.* [20] proposed an *Environment-Reactive rules-Agents* (ERA) agent model for tackling constraint satisfaction problems. The ERA model describes competitive collaboration among agents, highlights the 'survival-of-the-fittest' principle and demonstrated the importance of having randomized strategies. In the proposed framework, we model agent interactions and the formation of agent society as a cooperative collaboration process.

Durfee [11] has outlined the major steps in distributed search by a multiagent system as: *task decomposition*, *task allocation*, *accomplishment* and *result synthesis*. *Negotiation* and *contract* are two of the important mechanisms to achieve coordination. Our agent framework views an optimization task completely from the bottom up and the solution emerges as a result of agent interaction. In other words, no central coordination and planning (both implicit and explicit) is in the system. This is the design of a new computational paradigm called *autonomy oriented computation* (AOC), which is outlined below.

Applications running on the grid have emerged in recent years. A software called *DisOpt* [8] for solving large scale optimization problems in distributed environments. However, it relies on task decomposition and coordination, which requires in-depth knowledge on the problem nature. We propose that the agents under our formulation are able to organize themselves to tackle the given problem.

2.3 Autonomy Oriented Computation

This paper will describe an algorithm called *Evolutionary Diffusion Optimization* (EDO) based on the proposed multiagent diffusion framework. In computational system term, EDO belongs to a class of computational methods called *Autonomy Oriented Computation* (AOC) [22]. There are three main classes of AOC methods:

- AOC-by-fabrication methods aim at replicating certain self-organized behavior observable in the real-world to form a general-purpose problem solver. The operating mechanism is more or less known and may be simplified during the modeling process. Research in Artificial Life is related to this AOC approach up to the behavior replication stage. Nature-inspired techniques such as *genetic algorithm* and *ant colony optimization* are typical examples of such an extension.

- AOC-by-prototyping methods aim at understanding the operating mechanism underlying a complex system to be modeled by simulating the observed behavior, through characterizing a society of autonomous entities. Examples of this approach include the study of Internet ecology, traffic jams and Web log analysis. This AOC approach relates to multi-agent approaches to complex systems in *distributed artificial intelligence*.

- AOC-by-self-discovery methods aim at the automatic discovery of a solution. The trial-and-error process of an AOC-by-prototyping algorithm is replaced by autonomy in the system. In other words, the distance measure between the desired emergent behavior and the current emergent behavior of the system in question becomes part of the environmental information that affects the local behavior of an entity. Some evolutionary algorithms that exhibit self-adaptive capability are examples of this approach.

EDO belongs to the class of AOC-by-self-discovery methods as it has mechanisms to automatically fine-tune its parameters when finding solutions to hard computational problems.

3. AGENT DIFFUSION

The multiagent diffusion framework describes how a population of autonomous agents decide for themselves what to do regarding a search task. Following the formulation in the ERA model [20], three key concepts forms the core of the proposed framework, namely, *environment*, *reactive rules* and *agents*. To reflect the nature of the reactive rules, they are called local behaviors hereafter. The following section describes these three key concepts. It is followed by detailed descriptions of the framework and local behaviors of an optimization agent.

3.1 The Fundamentals

Definition 1 (Environment)
The environment $\mathcal{E}$ of an optimization task is the task-dependent computational system on which a group of agents work. It can be static or dynamic in nature, depending on the task. It is capable of evaluating the set of numbers that an agent owns, via a functional form $\mathcal{F}$ of the optimization task such as Equation 1, and provide feedback in the form of a scalar value. The environment also acts as the placeholder for global information $\mathcal{I}_g$ that every agent requires. Formally, $\mathcal{E} =< \mathcal{F}, \mathcal{I}_g >$.

Definition 2 (Local Behaviors)
Every agent in a multiagent optimization system is capable of exhibiting a set of local behaviors $\mathcal{L}$. In the current framework, a common set of behaviors is defined for all agents in the system. These behaviors can be categorized into three subsets that will either: (i) enact upon an agent itself to change its internal state ($\mathcal{L}_i$), (ii) affect the organizational structure of the multiagent system ($\mathcal{L}_o$), or (iii) cause direct communication with another agent to occur ($\mathcal{L}_c$). However, none of them will enact upon other agents directly. Formally, $\mathcal{L} = \{\mathcal{L}_i, \mathcal{L}_o, \mathcal{L}_c\}$.

Definition 3 (Agents)
The basic functional entity in a multiagent optimization system is an agent x. It is capable of selecting one or more of the reactive rules based on information it receives alone.

The above definitions are suffice to define an optimization system with only one agent and one with many agents.

Definition 4 (A single-agent optimization system)
A single-agent optimization system $\mathcal{AS}_1$ is the tuple $< \mathcal{E}, \mathcal{L}_i, x >$. There is no organizational changes and inter-agent communication for the obvious reason.

Example 1
The classical simulated annealing algorithm maintains a single solution at all times. A single new test case is generated by making some changes to the set of numbers maintained by the agent. The new case will be adopted by the agent according to a probability distribution. There is no other agents in the system.

Definition 5 (A multiagent optimization system)
A multiagent optimization system $\mathcal{AS}_m$ is the triplet either of the form $< \mathcal{E}, \mathcal{L}, \mathcal{A} >$, $< \mathcal{E}, \mathcal{L} \setminus \mathcal{L}_o, \mathcal{A} >$, or $< \mathcal{E}, \mathcal{L} \setminus \mathcal{L}_c, \mathcal{A} >$, where $\mathcal{A}$ is the set of agents $\mathcal{A} = \{x_1, x_2, \ldots, x_n\}$.

Example 2
Genetic algorithm (GA) is an example of a multiagent optimization system. It has a population of agents representing a group of candidate solutions. The mutation operator in GA corresponds to the local behavior $\mathcal{L}_i$. The recombination operation corresponds to the local behavior $\mathcal{L}_o$. There is no inter-agent communication and there are behaviors that are not local to a particular agent, e.g. the selection process.

3.2 An Optimization Agent

An agent x can be described by the tuple $< \mathcal{B}, \mathcal{S} >$, where $\mathcal{B}$ is the belief of an agent and $\mathcal{S}$ is the state of an agent. The set of agents $\mathcal{A}$ are structurally organized into *families* to facilitate inter-agent collaboration and communication. It should be pointed out that all agents share the same goal, *i.e.* find the optimal solution to the optimization task.

Definition 6 (Agent Belief)
The belief $\mathcal{B}$ of an optimization agent is its set of estimates of the likelihood of finding the optimal solution in a certain direction with respect to a particular position on the search landscape. This is shared among members of the same agent family. Agent belief should not be treated as global information as it is sensitive to the context (i.e. position in the solution space).

Definition 7 (Agent State)
The state $\mathcal{S}$ of an optimization agent can be represented by the tuple $< a, s, f, \mathcal{V}, \mathcal{M} >$ where a is the age of an agent since it was created, s is its status (active, inactive, or dead), f is its fitness with respect to the optimization task, $\mathcal{V}$ is a set of values used to obtain the fitness and $\mathcal{M}$ is a set of modifications made to $\mathcal{V}$ since last communication was performed.

Definition 8 (Agent Family)
An agent family $\mathcal{A}_f$ consists of a parent agent and its offspring agents. All the agents of the same family share the same belief and the offspring agents provide information for belief revision to the parent agent. Therefore, the set of agents $\mathcal{A}$ can be represented by the set of agent families $\{\mathcal{A}_{f1}, \mathcal{A}_{f2}, \ldots, \mathcal{A}_{fx}\}$.

3.3 Agent Behaviors

The set of behaviors associated with the agents in the multiagent system consists of three categories. The internal state modifying behaviors $\mathcal{L}_i$ concern the set of values $\mathcal{V}$, the belief $\mathcal{B}$ and the age a of an agent. The organizational modifying behaviors $\mathcal{L}_o$ change the role of an agent under certain situation and creates a new family of agents as a result. This framework also allows the removal of an agent from the system when its lifespan expires. The inter-agent communication behaviors $\mathcal{L}_c$ concern feeding information from an offspring agent to its parent agent for the purpose of updating the family belief. We will describe the behaviors below and present them formally using the following notation:

$$var \xrightarrow{behavior} var' \quad \text{(new var)}$$

Definition 9 (Modify Value Set)
The value set maintained by an agent is a candidate solution to the optimization task. To explore a different candidate solution, an agent modifies one or more of these values to reach a different point in the solution space. This process is called diffusion as it involves minute changes to the value set and is analogous to a local search strategy. The direction of change is selected after consulting the belief of the family to which the agent belongs. The diffusion process τ, which will only be performed by offspring agents, can be represented as:

$$\mathcal{V} \xrightarrow{\tau(\mathcal{B})} \mathcal{V}'$$

A special case of diffusion is when the belief is not consulted. In this case, directions of change are selected randomly. The random diffusion process, or random walk, τ_{rw} can be represented as:

$$\mathcal{V} \xrightarrow{\tau_{rw}(\{\})} \mathcal{V}'$$

One of these two strategies is selected every time diffusion is to be enacted. The selection process σ will base it decision on some of the state information of an agent, such as age and/or fitness.

$$\{\tau, \tau_{rw}\} \xrightarrow{\sigma(\mathcal{S}_{x_i})} \tau_j : \tau_j \in \{\tau, \tau_{rw}\}$$

No matter what kind of diffusion has taken place, the changes are recorded in the modification set $\mathcal{M}$.

$$\mathcal{M} \xrightarrow{\tau'(\mathcal{V},\mathcal{V}')} \mathcal{M}'$$

Definition 10 (Revise Agent Belief)
The belief of an agent (or rather an agent family) is an estimation of the whereabouts of the optimal solution based on past experience. It has to be updated continuously and using the current context. The belief revision process ψ relies on the change information, as well as an indication whether the changes have been successful or not, feed back to the parent agent from its offspring agents. The result is a set of new estimates, which will be used by all offspring agents in the family to perform future diffusion. It should be noted that only a parent agent will perform this behavior.

$$\mathcal{B} \xrightarrow{\psi(\mathcal{M}_{x_i},+|-)} \mathcal{B}'$$

where $\mathcal{M}_{x_i}$ is the modification set of agent x_i.

Definition 11 (Aging)
The age, a, of an agent keeps track of the number of iteration since being created. The aging process ρ increases the age by one.

$$a \xrightarrow{\rho} a + 1$$

Definition 12 (Agent Death)
An agent will remove itself from the system (or die χ) once its age has reached a certain lifespan limit Θ unless its fitness f is above the population average fitness $\overline{f}$. In addition, a parent agent who has become the only member of the family when all members have out-lived their lifespan will also be removed.

$$\mathcal{A} \xrightarrow{\chi(a,f,\overline{f},\Theta)} \mathcal{A} \setminus x_i$$

where x_i is the agent concerned.

Definition 13 (Reproduce)
When an agent has a fitness higher than its parent, it will replicate itself a number of times up to certain limit Ω. Each offspring agent will then be subjected to the diffusion process. In this way, a new agent family is born and the reproducing agent will sever linkage with its own parent agent once the parent's belief has been updated and copied. This action is necessary to make sure the belief remain relevant to the current context of the agent concerned and its offsprings.

$$\mathcal{A} \xrightarrow{\pi(x_i,\Omega)} \mathcal{A} \cup \mathcal{A}_\pi$$

where $\mathcal{A}_\pi$ is the set of new agents created during the reproduction process.

Definition 14 (Inter-agent Communication)
For the purpose of updating the belief of the agent family, all offspring agents in the family will communicate with the parent agent about the result of their diffusion actions. An agent will pass to the parent agent the changes it has made so far since the last communication and indicates whether the changes has been successful or not. The timing of the communication is dictated by the availability of a communication channel. However, in the event of a successful diffusion, the agent can request the parent to communicate since the agent needs to reproduce.

4. THE EDO ALGORITHM

This section details an implementation of the multiagent diffusion framework in *Evolutionary Diffusion Optimization* (EDO). Figure 1 shows the pseudocode of the major steps. A more detailed analysis of the behavior and performance of EDO can be found in [30].

4.1 An EDO Agent

Agents in EDO are divided into two categories. *Active* agents are those that are performing the search for the optimal solution. They will engage in the behaviors of *diffuse* and *revise* their belief *negatively*. In contrast, *inactive* agents are those that have found a position that is better than their parents, though not necessarily a sub-optimal or global optimal solution. These agents will not diffuse any more but will *reproduce* and revise their belief *positively*.

An agent e in the population E maintained by EDO is the tuple (P, S) where S is the agent state and P is the *probability matrix* representing the agent's belief of the optimal solution's location. The agent state S is a triple (a, f, V), where V is the *object vector*, a and f are scalars representing the *age* and *fitness* of the agent respectively. While V contains the values of the potential solution, P, a and f are crucial to the search process of EDO.

4.1.1 The Object Vector

The object vector V can be represented as:

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

where LB and UB are the lower bound and upper bound respectively. All function variables can take a value within these bounds.

```
WHILE (number of agents > 0) and
        (number of iterations < limit)
    evaluate agent
    if (performance better than parent)
        positive feedback
        reproduce
        parent becomes inactive
    else
        negative feedback
        diffuse
        aging
    end-if
END-WHILE

Procedure Diffuse
    if rand() > $P_{rand\_walk}$
        Random Walk
    else
        for each variable
            select step direction and size
        end-for
    end-if

Procedure Reproduce
    quota ← $f(fitness)$
    create new probability matrix
    for each offspring
        copy variables and point to new prob. matrix
        Diffuse
    end-for

Procedure Aging
    age ← age +1
    if fitness < parent's fitness * threshold or
        (age > lifespan and fitness < average)
            Remove
    end-if
```

Figure 1: Pseudocode for EDO

4.1.2 *Fitness*

Fitness, f, in EDO measures an agent's degree of success in the course of the search for the optimal solution. For simplicity, the objective function value is used as fitness in EDO if the task is minimization.

4.1.3 *Age*

The age, a, of an agent in EDO means the number of iterations this agent has survived since birth. Once an agent becomes a parent, the age does not need to be updated any more.

4.1.4 *Probability Matrix*

The probability matrix, P, implements the agent belief of the proposed multiagent optimization system. Specifically, the probability matrix is an $n \times m$ matrix representing the n function variables to be optimized and m possible steps of motion, i.e. $y = (m-1)/2$ steps towards the upper bound and the lower bound respectively, and remain stationary. A global step size parameter, Δ, governs the unit of change to all function variables and the product of Δ and the number of steps becomes the final modification to be effected on V. Formally,

$$P = \{P_1, P_2, \ldots, P_n\}$$

$$P_i = \{p_{i,-y}, \cdots, p_{i,0}, \cdots, p_{i,y}\}, 0 \leq p_{i,j} \leq 1$$

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

At the beginning of the search, all entries are initialized to $1/m$, which means it is equally likely to make any of the possible moves.

4.2 Diffusion

Two diffusion strategies are implemented in EDO, namely *rational move* and *random walk*. They operate as described below.

4.2.1 *Rational Move*

In the majority of time, an agent modifies its object variables by choosing randomly the number of steps to take according to the probability matrix. The actual amount of change is the product of current step size and the number of steps chosen:

$$v_i = v_i + \delta v_i \cdot \Delta, \forall\, i$$

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

where v_i is the ith function variable, δv_i is the amount of change made to v_i, 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).

4.2.2 *Random Walk*

As an agent becomes older and still has not located a place better than that of its parent, it will decide to take a random walk with increasing probability. The probability to take random walk, P_{rw}, is given by:

$$P_{rw} = exp\,[-\frac{\Theta - a}{\alpha}]$$

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

The direction of movement is first chosen uniformly between towards the upper bound, towards the lower bound and no move. In case a move is to be made, a new value between the chosen bound and the current value is then selected randomly.

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

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

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

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

4.3 Reproduce

A reproducing agent in EDO will replicate itself a number of times according to a quota system. The new agents are then sent off to a new location by *rational move*. The number of offspring to reproduce is determined according to the following two rules.

4.3.1 *Differentiation Rule*

An agent 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 individual x having fitness f is:

$$q_x = \begin{cases} \Omega, & \text{if } \frac{f_x}{\overline{f}} \leq 0.25 \\ \Omega - 1, & \text{if } 0.25 < \frac{f_x}{\overline{f}} \leq 0.5 \\ \Omega - 2, & \text{otherwise} \end{cases}$$

where $\overline{f}$ is the population average fitness.

4.3.2 Population Size Rule

Further to the above differential scheme, the reproduction quota is subjected to further restriction to avoid overcrowding:

$$q_x = \lfloor \frac{q_x * (\Pi - |E|)}{\Pi} \rfloor$$

where $|E|$ is the size of the population of agents, and Π is the maximum population size.

4.4 Aging

Aging is the process to help EDO to keep track of the unproductive moves throughout the search. By limiting the number of unsuccessful moves, EDO can properly channel resource to explore the search space. However, sufficient time should be put aside to allow each agent to survey its neighborhood.

At the end of each iteration, the age of all agents is increased by one. Agents whose age is greater than the lifespan limit are eliminated from the system. However, it is necessary to provide exceptions to the rule to either retain the exceptionally good performer or eliminate prematurely the exceptionally bad ones.

4.4.1 Extended Life

Lifespan limit of an agent is extended by one iteration when it expires if its fitness is higher than the population average.

4.4.2 Sudden Death

An unsuccessful agent is eliminated if the agent's fitness is less than a certain percentage of the fitness of the parent. The threshold is set at 80% in the experiments reported later.

4.4.3 Rejuvenate

An inactive agent is allowed to spawn new offspring if all its offspring agents are dead but its fitness is better than the population average. The main idea behind *rejuvenate* is that the inactive agent has been receiving reinforcement signals from its offspring, its probability matrix contains the latest information regarding the neighborhood landscape.

4.5 Feedback

All agents in EDO pass information back to their parents every time it has taken a good move to a better location or bad move. This information allows the parent to update its probability matrix, which is shared among its offspring.

4.5.1 Positive feedback

A successful move, which may happen after taking many moves, is defined as a gain in fitness. In order to bias the future moves of an agent's siblings to its own successful move, we update the probabilities in the parent's probability matrix that corresponds to the changes made in the successful agent's object vector. The update rule is:

$$p_{i,j} = \frac{(p_{i,j} + \beta)}{(1 + \beta)}$$

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

4.5.2 Negative feedback

In order to steer an agent's siblings away from non-optimal area in the search space, agents will update the parent's probability matrix after each unsuccessful move using the following rule:

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

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

Negative feedback is a finer grain update than the positive reinforcement signal as it happens after each step. Moreover, the use of a multiplicative scaling factor ensures that the probability remains bigger than zero at all times. The whole probability matrix is normalized after updating within the set of probabilities for each dimension.

4.6 The Environment in EDO

Various system-wide information has been mentioned in the previous section. They either controls the flow of the search or act as parameters to some of the features in EDO. Below is a summary of the global information:

- Step Size (Δ): The size of a step to be taken during *rational move*. It is used in combination with the probability matrix to decide what should the actual change be. For example, if the two steps toward the upper limit were chosen based on the probability distribution in the probability matrix, a value equal to two times of Δ will be added to the function variable in question.

- Lifespan (Θ): This is the duration, in iterations, given to an agent to perform search. It aims at limiting the amount of unsuccessful exploration any agent can perform before it is eliminated.

- Maximum Offspring (Ω): This is the maximum number of offspring any reproducing agent is allowed to spawn at one time. It represents the amount of local exploration an agent is allowed to perform in the neighborhood of a good solution.

- Maximum Population Size (Π): This is the upper limit of the number of agents, both active and inactive together, an EDO session can keep.

The *step size* parameter, Δ, is reduced according to the *golden ratio*[1] if the current best has not been renewed for half of *Lifespan*, Θ. The rationale behind this reduction is that the agents may be in the neighborhood of a minimum. Therefore, finer steps are required for careful exploitation. Conversely, if the population has been improving continuously for some time (in number of iterations, again equal to half of *Lifespan*), the step size is increased according to the golden ratio (division):

$$\Delta = \begin{cases} \Delta * \phi, & \text{if} \quad u < \Theta/2, \phi = \frac{1+\sqrt{5}}{2} \\ \Delta/\phi, & \text{otherwise} \end{cases}$$

where u is the number of times the current best solution has been renewed since the step size parameter was last updated, and ϕ is the golden ratio.

4.7 Experimental Results

Four benchmark functions are chosen to test the EDO algorithm (see references within [31]). F1 and F2 are unimodal functions

[1]the golden ratio is one of the roots to the equation $x^2 - x - 1 = 0$ and many visually appealing geometry shapes in nature are having a golden ratio dimension [10].

Table 1: Average number of function evaluations (over 50 runs) required by EDO and ASA to optimize the benchmark functions of various complexity and the success rate of both methods

	Target	EDO			ASA		
		Average	Std Dev.	Success	Average	Std. Dev.	Success
F1_5	1.0×10^{-6}	2,000	1,168	96%	5,235	2,828	100%
F1_10		3,240	5,017	94%	26,355	12,144	100%
F1_15		6,816	10,234	88%	61,209	28,316	100%
F1_20		17,277	31,174	84%	112,261	58,460	100%
F1_25		26,683	24,031	84%	178,062	82,119	92%
F1_30		167,094	878,892	84%	235,071	103,478	100%
F2_5	2.0×10^{-3}	1,705	1,472	90%	13,701	72,864	100%
F2_10		7,710	11,875	96%	9,302	27,483	96%
F2_15		29,328	38,789	90%	10,725	16,801	90%
F2_20		103,910	141,580	94%	68,189	173,688	82%
F2_25		227,959	490,607	90%	35,689	50877	92%
F2_30		2,224,903	4,558,546	94%	91,098	181,442	82%
F3_5	1.0×10^{-6}	15,687,788	5,812,865	74%	101,468	61,699	94%
F4	0.998004	12,172	48,182	90%	78,441	251,730	58%

while F3 and F4 are multimodal functions. The unimodal functions are used to demonstrate the hill climbing capability of EDO and the multimodal functions are used to test EDO's ability to escape from local minima.

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

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

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

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

$$a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$$

Several forms of the functions, except function F4 (which has only two free parameters), are tested by increasing the number of free parameters. The intention is to test the performance and scalability of EDO.

In order to benchmark the performance of EDO with some existing algorithms, some experiments were also conducted using adaptive simulated annealing (ASA) [16][2]. The average number of steps (function evaluations) required to reach the target over 50 runs using different random seed is shown in the Table 1 along with the success rate, which measures the percentage of times when the method can reach the target. Both EDO and ASA are considered failing to reach the target by comparing the function value achieved when they stopped, with the target. EDO stops either when the maximum number of iterations is reached, which is set at 5,000, or when the number of agents reaches zero. ASA stops when the maximum number of steps is reached, which is set at 2,000,000. The targets shown in Table 1 are extremely close to the optimal solutions.

[2]source code obtained from http://www.taygeta.com/annealing

The general trend is that more complex problems requires more time (iterations) to solve. However, multimodal functions tend to be more difficult to solve than unimodal ones. It can be observed from Table 1 that EDO performs better than ASA in F1, F2 with lower dimensions and F4. As the complexity of the problem increases, EDO tends to fall behind. Moreover, EDO tends to be less robust than ASA as it fails to reach the target in more cases than ASA. While one has to wait until the maximum number of steps is reached to decide whether ASA has reached the target, EDO generally stops within few tens of hundreds of iterations. Therefore, EDO allows a better chance to restart the search using another random seed.

5. CONCLUSIONS

This paper has described a multiagent diffusion framework for optimization based on diffusion. The agents in the framework cooperative by forming families of agents. Through this organizational structure, agents share the same belief regarding the search space as well as their exploration results with their parent. As a result, they share these crucial information implicitly with their siblings. Moreover, the proposed framework restricted communication requirement to a minimum by elimination negotiation. In other words, the agents collaborate by self-organization.

An implemented example of this agent framework is the new search algorithm called *evolutionary diffusion optimization* (EDO). Experiments reveal that EDO is able to automatically maintain a good balance between exploration and exploitation by maintaining high population diversity. On the other hand, the ability to automatically adapt the search step size is proved to be very useful. Results from the experiments on four benchmark functions show that EDO performs better than adaptive simulated annealing (ASA) in two of them while ASA excels in the rest two.

Despite the limited success of the EDO algorithm, the proposed agent framework need to be verified through more extensive experimental work. We are currently working on another implementation that works on combinatorial optimization problems. One of the areas for future study is on the collaboration between families of agents. As they are exploring the search space in close proximity, certain niche may be identify by the families separately. It would be useful to extend the inter-agent communication to inter-family communication for exchange family belief. Another area worth pursuing is to make as much global information into local information as possible so that each agent family can perform the search more effective. The obvious candidate for such change is the step size parameter [19]. As agents are performing search in parallel, they may reach a different stage during search - some may be far away from an (sub-)optimal solution while others may be far away. A single step size may not be useful to all.

6. REFERENCES

[1] Crossgrid website. http://ups.savba.sk/parcom/crossgrid/main.html.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[3] S. Baluja. 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, Pittsburgh, PA, 1994.

[4] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The International Conference on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[5] S. W. Bednarz. Mission geography. http://geog.tamu.edu/sarah/humangeog/migration8.html, 2000.

[6] W. Chainbi, A. Ben-Hamadou, and M. Jmaiel. A belief-goal-role theory for multiagent systems. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(3):435–450, 2001.

[7] W. A. V. Clark. *Human Migration*. Sage Publications, 1986.

[8] J. Contreras, A. Losi, M. Russo, and F. F. Wu. DistOpt: A distributed optimization software modeling and evaluation framework. *Journal of Parallel and Distributed Computing*, 60(6):741–763, 2000.

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

[10] R. A. Dunlap. *Golden Ratio and the Fibonnaci Numbers*. World Scientific, Singapore ; River Edge, NJ, 1997.

[11] E. H. Durfee. Distributed problem solving and planning. In G. Weiß, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, volume 2086, chapter 3, pages 111–164. MIT Press, Cambridge, MA, 2001.

[12] L. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulation Evolution*. Wiley, NY, 1966.

[13] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, 1992.

[14] R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.

[15] R. Horst and H. Tuy. *Global Optimization : Deterministic Approaches*. Springer-Verlag, 1990.

[16] L. Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Journal of Control and Cybernetics*, 25:33–54, 1996.

[17] J. Kennedy. The particle swarm: Social adaptation of knowledge. In *Proceedings of the 1997 International Conference on Evolutionary Computation*, pages 303–308, Piscataway, NJ, 1997. IEEE Service Center.

[18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671 – 680, 1983.

[19] Y. Li. *Directed Annealing Search in Constraint Satisfaction and Optimization*. PhD thesis, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, October 1997.

[20] J. Liu, J. Han, and Y. Y. Tang. Multi-agent oriented constraint satisfaction. *Artificial Intelligence*, 136(1):101–144, 2002.

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

[22] J. Liu and K. C. Tsui. Introduction to autonomy oriented computation. In *Proceedings of 1st International Workshop on Autonomy Oriented Computation*, pages 1–11, 2001.

[23] J. Mockus. *Bayesian Approach to Global Optimization : Theory and Applications*. Kluwer Academic, 1989.

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

[25] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, pages 312–317, San Francisco, 1995. AAAI Press.

[26] R. G. Reynolds. An introduction to cultural algorithms. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, NJ, 1994.

[27] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley Inter-Science, 1995.

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

[29] A. Torn and A. Zilinskas. *Global Optimization*. Springer-Verlag, 1989.

[30] K. C. Tsui and J. Liu. An evolutionary multiagent diffusion approach to optimization. *International Journal of Artificial Intelligence and Pattern Recognition*, 16(6):715–733, September 2002.

[31] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.