Nature-Inspired Particle Mechanics Algorithm for Multi-Objective Optimization

Xiang Feng and Francis C.M. Lau

Department of Computer Science, The University of Hong Kong, Hong Kong
{xfeng,fcmlau}@cs.hku.hk

In many real world optimization problems, several optimization goals have to be considered in parallel. For this reason, there has been a growing interest in multi-objective optimization (MOO) in the past many years. Several new approaches have recently been proposed, which produced very good results. However, existing techniques have solved mainly problems of "low dimension", i.e., with less than 10 optimization objectives. This chapter proposes a new computational algorithm whose design is inspired by particle mechanics in physics. The algorithm is capable of solving MOO problems of high dimensions. There is a deep and useful connection between particle mechanics and high dimensional MOO. This connection exposes new information and provides an unfamiliar perspective on traditional optimization problems and approaches. The alternative of *particle mechanics algorithm* (PMA) to traditional approaches can deal with a variety of complicated, large scale, high dimensional MOO problems.

1 Introduction

While single-objective *mimetic algorithms* (MAs) are well established and relatively easy to parallelize, this is not the case for many-objective mimetic algorithms. Until recently, many-objective combinatorial optimization did not receive much attention in spite of its potential application. One of the reasons is due to the difficulty of deriving many-objective combinatorial optimization models that are satisfactory. Nevertheless, the need for parallelizing many-objective mimetic algorithms to solve many-objective combinatorial optimization problems is a real one.

"Distribution problems" is a well-known class of fundamental combinatorial optimization problems, which are much more difficult to solve than assignment problems. Many practical situations can be formulated as a distribution problem. In fact, assignment problems and transportation problems are both sub-problems of the distribution problem.

In [1, 2], some methods were proposed to generate the entire set of exact solutions for simple assignment problems. These methods however seem to be efficient only for small-scale instances. For large-scale instances or more complex problems, their NP-hardness and the multi-objectivity make these problems "intractable" by those methods. For this reason, it makes sense to consider "approximate" methods such as swarm intelligent methods which have been found to be efficient in treating combinatorial optimization problems; these latter methods can function independently of the mathematical structure of the problem, and generate excellent solutions in a very short time [3].

Progress often occurs at the boundaries between disciplines. Throughout history, many artifacts have been built, whose inspiration derives from the natural world. In the field of computer science, especially in artificial intelligence, the need of parallel and distributed intelligent theories and approaches inspired by the principles of nature for difficult problems becomes greater and greater. Bio-inspired approach is probably the most representative one of nature-inspired approaches (NAs) in recent times. On the other hand, physics-inspired approaches did not receive as much attention in spite of their potential advantages. Since simulated annealing algorithm (SAA) was proposed successfully in 1983 [4], no other innovative and significant physics-inspired approach has been suggested. On the contrary, significant bio-inspired approaches such as genetic algorithm (GA) (1975) [5], ant colony optimization (ACO) (1991) [6], particle swarm optimization (PSO) (1995) [7] were emerging one after another. In 1987, the elastic net (EN) approach [8] was presented, which is a physics-inspired approach. Unfortunately, the application and influence of EN were limited by EN's unsubstantial theory foundation. One must not forget although biology is experiencing rapid development, physicists have also made great strides (e.g., particle mechanics) in recent years.

This chapter proposes a new approach—based on what we call the *particle mechanics algorithm* (PMA)—to compute in parallel approximate efficient solutions to the distribution problem with multiple objectives. By proposing this approach, we try to explore a potentially new branch of MA (or NA), which is based on the laws of physical mechanics. Just like other MAs (or NAs) which draw from observations of physical processes that occur in nature, our particle mechanics (PM) based approach is inspired by physical models of particle kinematics and dynamics.

In PMA, we use mathematical formulations to describe or predict the properties and the evolution of the different states of particles. The particles represent distinct parameters in the problem, which follow a path towards a solution. Borrowing from "differential equation theory", we have developed efficient techniques for solving multi-objective optimization problems. The goal is to have all objectives optimized individually and then collectively, and satisfying all the given restrictions.

In the physical world, mutual attraction between particles causes motion. The reaction of a particle to the field of potential would change the particle's coordinates and energies. The change in the state of the particle is a result of the influence of the potential. For PMA, the objectives of individual optimizations are reached by the autonomous self-driving forces of the particles. Global optimization is achieved by the potential of the field, and any restrictions of the problem are satisfied via interaction potential between the particles.

Each particle is described by some differential dynamic equations, and it moves (to a new state in the field) according to the results of these calculations. Specifically, each particle computes the effect of its autonomous self-driving force, the field potential and the interaction potential. If the particles cannot reach an equilibrium, they will proceed to execute a goal-satisfaction process.

Although there are obvious differences between particles in classical mechanics and those in PMA, PMA is by and large inspired by classical mechanics. PMA enables feasible many-objective optimization in very large scales. The approach has a low computational complexity, which is crucial for its functioning in solving large-scale distribution problems.

This chapter is part of the authors' research work on distributed parallel theories and approaches for intelligent processing based on the generalized particle model (GPM). They have proposed the crossbar composite spring net (CCSN) approach [9], from which the GPM approach has evolved. They studied distributed and parallel algorithms for intelligent processing based on GPM, and their application in networks. GPM's application in the bandwidth allocation problem was presented in [10]. Variations of the basic theme then resulted in several extended GPM models, including the "economic generalized particle model" (E-GPM) which draws upon the economics theory of Tatonnement processes; the model has also been applied to the bandwidth allocation problem in communication networks.

All the authors' past methods based on GPM targeted at a specific application to a real-life problem. In this chapter, the PM method is described as a "generic" method meaning potentially it can be applied to a range of different problems.

The structure of this chapter is as follows. In Section 2, we present the multi-objective particle mechanics algorithm (PMA). Section 3 introduces the parallel mimetic PM algorithm for solving multi-objective problems. In Section 4, we discuss the physical meanings of PMA. In Section 5, we give some experimental results. We conclude the chapter in Section 6.

2 The PM Approach for the Multi-Objective Distribution Problem

Definition 1. In a multi-objective framework, the distribution problem can be formulated as

$$\begin{cases} \min : z^q(X) = (C^q)^T X = \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij}^q x_{ij} \quad q = 1,2,\cdots,Q \\ s.t. \quad \sum_{i=1}^{I} x_{ij} = 1 \quad j = 1,2,\cdots,J \\ \sum_{j=1}^{J} x_{ij} = 1 \quad i = 1,2,\cdots,I \end{cases} \tag{1}$$

where X is a two-dimensional distribution vector, C^q a two-dimensional weight vector ($q = 1,2,\cdots,Q$), and Q is the number of multiple objectives.

With this problem model, we can now examine the evolutionary multi-objective model which can mathematically describe PMA for the multi-objective distribution problem. The theory of evolution is a dynamical theory. The evolutionary dynamics will drive PMA to the equilibrium state.

Definition 2. The distribution and weight dynamic equations of PMA are defined, respectively, by

$$x(t+1) = x(t) + \Delta x(t) \tag{2}$$

$$c(t+1) = c(t) + \Delta c(t) \tag{3}$$

The two dynamic equations are seen as the "PMA evolution" by fictitious agents which manipulate the distribution and weight vectors until an equilibrium is reached. In PMA, the rows and columns of distribution vector X are treated as two kinds of fictitious agents (service particles and task particles). In fact, the weight vector is invariable; the evolution of the weight vector only occurs in the computing process in order for us to obtain efficient solutions of the distribution vector.

For fictitious agents—service particles (O) and task particles (S), there are three factors related to the distribution vector (X) and the weight vector (C):

- personal utility (u) (to realize the multiple objectives);
- minimal personal utility (to realize max-min fair distribution and to increase the whole utility) (F);
- interaction among particles (to satisfy the restrictions) (I).

According to "differential equation theory", a variable's increment to make it minimum is equal to the sum of negative items from related factors differentiating the variable. So we have the following definitions.

Definition 3. The increments of distribution and weight are defined, respectively, by

$$\Delta x \approx \frac{dx}{dt} = -\sum_{q=1}^{Q}(\lambda_1^q \frac{\partial u_O^q}{\partial x} + \lambda_2^q \frac{\partial F_O^q}{\partial x}) - \lambda_3 \frac{\partial I_O}{\partial x} \tag{4}$$

$$\Delta c^q \approx \frac{dp}{dt} = -(\gamma_1^q \frac{\partial u_S^q}{\partial c} + \gamma_2^q \frac{\partial F_S^q}{\partial c}) - \gamma_3 \frac{\partial I_S}{\partial c} \quad q = 1,2,\cdots,Q \tag{5}$$

where $\lambda_1^q, \lambda_2^q, \lambda_3, \gamma_1^q, \gamma_2^q, \gamma_3$ are coefficients ($q = 1,2,\cdots,Q$).

Definition 4. Three kinds of factor functions for service particles and task particles are defined, respectively, by

$$u_{Oi}^q = 1 - \exp\left(-\sum_{j=1}^{J} c_{ij}^q \cdot x_{ij}\right) \quad q = 1,2,\cdots,Q \tag{6}$$

$$F_O^q = (k_O^q)^2 \text{In} \sum_{i=1}^{I} \exp[(u_{Oi}^q)^2 \big/ 2(k_O^q)^2] \quad q = 1,2,\cdots,Q \tag{7}$$

$$I_O = a_1 \sum_{i=1}^{I}\left(\sum_{j=1}^{J} x_{ij} - 1\right)^2 + a_2 \sum_{j=1}^{J}\left(\sum_{i=1}^{I} x_{ij} - 1\right)^2 \tag{8}$$

$$u_{Sj}^q = 1 - \exp\left(-\sum_{i=1}^{I} c_{ij}^q \cdot x_{ij}\right) \quad q = 1,2,\cdots,Q \tag{9}$$

$$F_S^q = (k_S^q)^2 \text{In} \sum_{i=1}^{J} \exp[(u_{Sj}^q)^2 \big/ 2(k_S^q)^2] \quad q = 1,2,\cdots,Q \tag{10}$$

$$I_S = I_O \tag{11}$$

where k_O^q, a_1, a_2, k_S^q are coefficients.

Now, we explain why the three kinds of functions are chosen.

1. The smaller the value of the summation in Eq. 6, the more profit the i-th service particle would obtain. The optimization problem here is posed as a minimization problem. And we use the exponential function in order that $u^q_{Oi}(t)$ would be between 0 and 1. $u^q_{Oi}(t)$ can be regarded as the q-th dimensional utility of service particle. The smaller $u^q_{Oi}(t)$ is, the more profit service particle O_i would get. Schematically, the q-th dimensional utility function u^q_{Oi} of a particle corresponds to the q-th dimensional coordinate of the q-th dimensional force-field. We define the distance from the bottom boundary to the upper boundary of all q dimensional force-fields to be 1. The physical meaning of PMA is discussed in Section IV. A graphical presentation of $u^q_{Oi}(t)$ is shown in Fig. 1. Obviously, the smaller $u^q_{Oi}(t)$ is the better. The presentation of $u^q_{Sj}(t)$ in Eq. 9 is similar.
2. For Eq. 7, $0 < k^q_O < 1$ is a parameter to be tuned in the implementation. The smaller F^q_O is, the better. With Eq. 7, we attempt to construct a potential energy function, F^q_O, such that the decrease of its value would imply the decrease of the maximal utility of all the service particles. We prove that in Theorem 1. This way we can optimize the distribution problem in the sense that we consider not only the individual personal utility, but also the aggregate utilities, by decreasing the maximum utility of all the service particles again and again. In fact, k^q_O represents the strength of the downward gravitational force in the q-th dimensional force-field. The bigger k^q_O is, the faster the particles would move down; hence, k^q_O influences the convergence speed of the distribution problem. k^q_O needs to be carefully adjusted in order to minimize the q-th objective. The explanation of F^q_S in Eq. 10 is likewise.
3. For Eq. 8, $0 < a_1, a_2 < 1$. The smaller I_O is, the better. a_1, a_2 are weights applied to the distribution availability of service and the satisfactory ratio of the demands, respectively. Eq. 8 describes the effect of interactions among service particles during the distribution process. The first term and the second term of $I_O(t)$ perform penalty functions with respect to the constraints on the utilization of service (or resources) (i.e., service particles) and the degree of satisfaction of the demands (i.e., task particles) respectively. Therefore, distribution utilization and demands' satisfaction can be explicitly included as optimization objectives through some appropriate choices

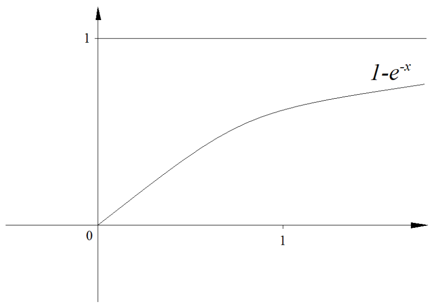

Fig. 1. Graphical presentation of $u^q_{Oi}(t)$

of the coefficients a_1 and a_2 respectively. We presume that there are specific interactive forces among particles, and these forces may cause the potential energy components represented by the first and second term of $I_O(t)$ to decrease. In Eq. 11, I_S is defined as the same as I_O.

We can therefore obtain the iteration velocity of service particles and task particles by the following equations, respectively.

$$v^q_{Oi} = du^q_{Oi}\big/dt = \frac{\partial u^q_{Oi}}{\partial x_{ij}}\frac{dx_{ij}}{dt} \tag{12}$$

$$v^q_{Sj} = du^q_{Sj}\Big/dt = \frac{\partial u^q_{Sj}}{\partial c^q_{ij}}\frac{dc^q_{ij}}{dt} \tag{13}$$

v^q_{Oi} represents the iteration velocity of the q-th objective by service particle O_i. Meanwhile, v^q_{Oi} represents the velocity of the downward movement of service particle O_i in the q-th dimensional force-field. The meaning of v^q_{Sj} is similar; it represents the iteration velocity of the q-th objective by task particle S_j and the velocity of the downward movement of task particle S_j in the q-th dimensional force-field.

Proving the PM model

In order to justify our choice of Eqs. 1-13 for the PM mathematical model, we give the following theorems.

Theorem 1. If k^q_O is very small, the decrease of F^q_O will cause a decrease of the service particles' maximal utility. (Likewise, if k^q_S is very small, a decrease of F^q_S will cause a decrease of the task particles' maximal utility.)

Proof. Supposing that
$M(t) = \max\limits_i[(u^q_{Oi})^2(t)]$. Because

$M(t) = \max\limits_i (u^q_{Oi})^2(t) \le \sum\limits_{i=1}^{I} (u^q_{Oi})^2(t) \le I \cdot \max\limits_i (u^q_{Oi})^2(t) = I \cdot M(t)$,

we have

$$\left[e^{\frac{M(t)}{2(k^q_O)^2}}\right]^{2(k^q_O)^2} \le \left[\sum_{i=1}^{I} e^{\frac{(u^q_{Oi})^2(t)}{2(k^q_O)^2}}\right]^{2(k^q_O)^2} \le \left[I \cdot e^{\frac{M(t)}{2(k^q_O)^2}}\right]^{2(k^q_O)^2}.$$

Simultaneously taking the logarithm of each side of the equation above leads to

$M(t) \ge 2(k^q_O)^2 \mathrm{In} \sum\limits_{i=1}^{I} e^{\frac{(u^q_{Oi})^2(t)}{2(k^q_O)^2}} \ge M(t) + 2(k^q_O)^2 \mathrm{In} I$,

$2(k^q_O)^2 \mathrm{In} \sum\limits_{i=1}^{I} e^{\frac{(u^q_{Oi})^2(t)}{2(k^q_O)^2}} \le M(t) \le 2(k^q_O)^2 \mathrm{In} \sum\limits_{i=1}^{I} e^{\frac{(u^q_{Oi})^2(t)}{2(k^q_O)^2}} - 2(k^q_O)^2 \mathrm{In} I$,

$2F_O(t) \le \max\limits_i u^q_{Oi}(t) \le 2F_O(t) - 2(k^q_O)^2 \mathrm{In} I$.

Since I is the number of service particles (the number of rows of the distribution vector X), $2(k^q_O)^2 \mathrm{In} I$ is constant.

It turns out that $F_O^q(t)$ at time t represents the maximum among $u_{Oi}^q(t)$ obtained by the service particle O_i, namely, the minimum of the personal profit obtained by a service particle at time t. Hence decreasing $F_O^q(t)$ implies the decrease of the maximal utility of the service particles. $\square$

Based on Theorem 1, we can construct the potential energy functions, F_O^q and F_S^q (see Eqs. 7 and 10), such that the decrease of their values would imply the decrease of the maximal utilities of the two kinds of particles. By decreasing the maximum utilities of the two kinds of particles again and again, the aggregate utilities of whole problem will decrease, which is the basis that PM algorithm can obtain the Max-min fair solution.

Firstly, we give the definition of Max-min Fairness. Then, Theorem 2-5 will explain why the Max-min fair solution can be obtained by the PM mathematical model as defined in Eqs. 1-13.

Definition 5. (Max-min Fairness) [11] A feasible distribution X is max-min fair if and only if a decrease of any distribution x within the domain of feasible distributions must be at the cost of an increase of some already larger distribution x. Formally, for any other feasible distribution Y, if $y_{ij} < x_{ij}$ then there must exist some i' such that $x_{i'j} \geq x_{ij}$ and $y_{i'j} > x_{i'j}$.

Theorem 2. The behavior of the service particle O_i that is related to the term of Eq. 4, $-\lambda_2^q \frac{\partial F_O^q}{\partial x}$, will always bring about the decrease of the maximal utility of all service particles, and the decrement of the maximal utility is directly proportional to the coefficient vector λ_2^q. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_2^q \frac{\partial F_S^q}{\partial c}$, will always bring about the decrease of the maximal utility of all task particles, and the decrement of the maximal utility is directly proportional to the coefficient vector γ_2^q.)

Theorem 3. The behavior of the service particle O_i that is related to the term of the Eq. 4, $-\lambda_1^q \frac{\partial u_O^q}{\partial x}$, will always result in the decrease of the personal utility of service particle O_i, and the decrement of its personal utility is related to coefficient vectors λ_1^q. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_1^q \frac{\partial u_S^q}{\partial c}$, will always result in the decrease of the personal utility of task particle S_j, and the decrement of its personal utility is related to coefficient vectors γ_1^q.

Theorem 4. The behavior of the service particle O_i that is related to the term of the Eq. 4, $-\lambda_3 \frac{\partial I_O}{\partial x}$, will decrease the potential interaction energy function I_O, with the intensity of the decrease being proportional to coefficient vector λ_3. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_3 \frac{\partial I_S}{\partial c}$, will decrease the potential interaction energy function I_S, with the intensity of the decrease being proportional to coefficient vector γ_3.

Theorem 5. (Max-min fair allocation) Max-min fair allocation can be obtained by the mathematical model for the distribution problem with multi-objectives as defined in Eqs. 1–13.

The proofs of Theorems 2–5 are omitted.

3 The Parallel PM Algorithm

The results given in the previous sections suggest that we may use a parallel implementation of the evolutionary particle mechanics approach to solve the multi-objective distribution problem. We consider such an algorithm in Table 1.

The algorithm PMA has in general a complexity of $O(I+J)$, where $I+J$ is the number of particles (the sum of the number of rows and columns of X). The time complexity of the algorithm is $O(I_1)$, where I_1 is the number of iterations for Costep 2 (the while loop).

4 Physical Meaning of PMA

PMA puts emphasis on

- providing a view of individual and global optimization (with one to two objectives);
- parallelization with reasonably low time complexity;
- all objectives being optimized individually as well as collectively;
- the ability to deal with social interactions;
- the physical meaning of the model.

The mathematical model of PMA has its physical meaning.

In PMA, the rows and columns of the distribution vector X are treated as two kinds of generalized particles (service particles and task particles) that are located in two groups of force-fields, respectively, hence transforming the distribution problem into the kinematics and dynamics of the particles in the two groups of force-fields.

The two groups of force-fields are a group of service (or resource) force-fields and a group of task force-fields. Every force-field in a group of service force-fields or in a group of task force-fields is a Q-dimensional space where coordinates in the space are in $[0,1]$.

Table 1. The PM algorithm

```
1. Initialization:
      $t \leftarrow 0$
      $x_{ij}(t)$ , $c^q_{ij}(t)$                                  ——Initialize in parallel
2. **While** ($v^q_{Oi} \neq 0$ or $v^q_{Sj} \neq 0$ )   **do**
      $t \leftarrow t+1$
      $u^q_{Oi}(t)$                                   ——Compute in parallel according to Eq. 6
      $v^q_{Oi}$                                      ——Compute in parallel according to Eq. 12
      $u^q_{Sj}(t)$                                   ——Compute in parallel according to Eq. 9
      $v^q_{Sj}$                                      ——Compute in parallel according to Eq. 13
      $dx_{ij}(t)/dt$                                 ——Compute in parallel according to Eq. 4
      $x_{ij}(t) \leftarrow x_{ij}(t-1) + dx_{ij}(t)/dt$          ——Compute in parallel according to Eq. 2
      $dc^q_{ij}(t)/dt$                               ——Compute in parallel according to Eq. 5
      $c^q_{ij}(t) \leftarrow c^q_{ij}(t-1) + dc^q_{ij}(t)/dt$    ——Compute in parallel according to Eq. 3
```

If the number of minimum objectives is 1, the particles will move downwards on a 1-dimensional space (a line) ($x_1 \in [0,1]$) during the optimization process. If the number of minimum objectives is 2, the particles will move towards the origin in a 2-dimensional space (a plane) ($x_1 \in [0,1], x_2 \in [0,1]$) during the optimization process. Analogously, if the number of minimum objectives is Q, the particles will move towards the origin on a Q-dimensional space ($x_1 \in [0,1], \cdots, x_q \in [0,1], \cdots, x_Q \in [0,1]$) during the optimization process, where x_q is a coordinate of the q-dimensional space.

Particles in PMA move not only under outside forces, but also under their internal forces; hence they are different from particles in classical physics. The kinds of force-fields (resource force-field F_R and demands force-field F_D) are geometrically independent, without any forces directly exerted from each other; they are mutually influenced and conditioned by each other through a reciprocal procedure whereby the distribution policy of the distributions x and the weight policy of the weights c change alternatively. In this way, the two groups of force-fields form a pair of reciprocal dual force-field groups.

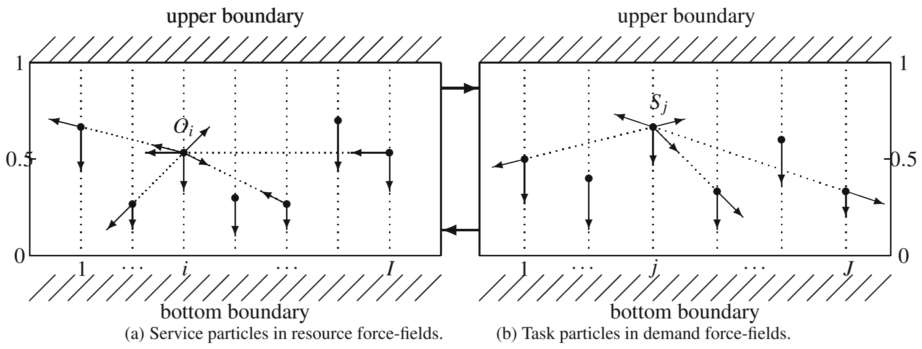

Fig. 2. The physical model of PMA for the distribution problem with one objective

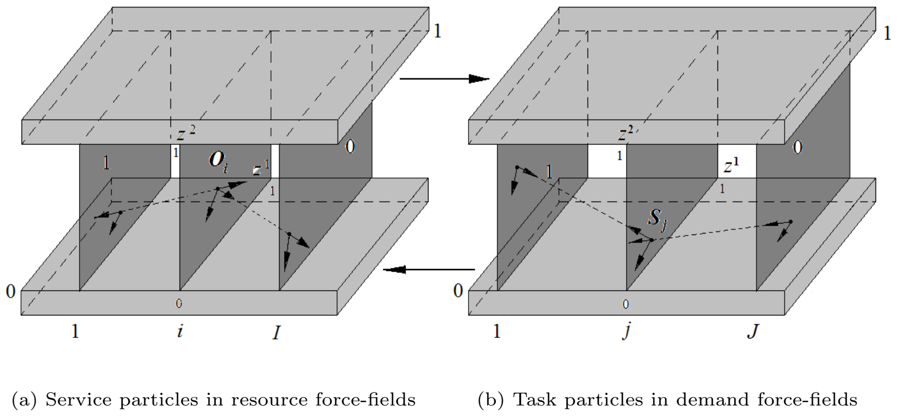
1
0
z^2
O_i
z^1
1 i I
(a) Service particles in resource force-fields
1
0
z^2
z^1
S_j
1 j J
(b) Task particles in demand force-fields

Fig. 3. The physical model of PMA for the distribution problem with two objectives

In a resource force-field F_R, the coordinates of the Q-dimensional space of service particles represent the utilities of the rows of the distribution vector X that are described by the service particles. A particle will be influenced simultaneously by several kinds of forces in the Q-dimensional space, which include the gravitational force of the Q-dimensional space force-field where the particle is located, the pulling or pushing forces stemming from the interactions with other particles in the same force-field, and the particle's own autonomous driving force.

When the number of minimum objectives is 1, all the above-mentioned forces that are exerted on a particle are dealt with as forces along a vertical direction (along a line). Thus a particle will be driven by the resultant force of all the forces that act on it upwards or downwards, and moves along a vertical direction. The larger the downward resultant force on a particle, the faster the downward movement of the particle. When the downward resultant force on a particle is equal to zero, the particle will stop moving, being at an equilibrium status. As shown in Fig. 2, the service particles that have service or resource move in the resource force-fields F_R, and the task particles that require distribution move in the demand force-fields F_D.

The downward gravitational force of a force-field on a particle causes a downward component of the motion of the particle, which represents the tendency that the particle pursues the common benefit of the whole group. The downward or upward component of the motion of a particle, which is related to the interactions with other particles, depends upon the strengths and categories of the interactions. The particle's own autonomous driving force is proportional to the degree the particle tries to move downwards in the force-field where it is located, i.e., the particle (service particle or task particle) tries to acquire its own minimum utility.

When the number of minimum objectives is two, each service particle and task particle move towards the origin in a unit plane, as shown in Fig. 3.

When the number of minimum objectives is Q, each service particle and task particle move towards the origin in a Q-dimensional space.

One major difference between the particle of the proposed generalized particle model and the particle of a classical physical model is that the generalized particle has its own driving force which depends upon the autonomy of the particle. All the generalized particles, both in different Q-dimensional spaces of the same force-field and in different force-fields simultaneously, evolve under their exerted forces; as long as they gradually reach their equilibrium positions from their initial positions which are set at random, we can obtain a feasible solution to the multi-objective distribution problem.

5 Simulations

Here, we give the experimental results which serve six purposes. First, we use a simple example (a small-scale problem) to explain how our PM algorithm is used. Secondly, the effectiveness of PM algorithm is tested on many-objective large-scale problems. Thirdly, we show the actual times and iterations used to solve multi-objective optimization problems on a cluster, which can verify the efficiency and parallelism of our PM algorithm. Fourthly, we test the effectiveness of our PM algorithm to solve optimization problems with different number of objectives (from single objective to many-objective problems). Fifthly, the performance of the PM algorithm is compared against NSGA-II [12] for two-objective distribution problems. Finally, we make a general comparison between the PM algorithm and other benchmark MAs.

All the experiments presented in this section are completed on a cluster. Each of the machines of the cluster has a Pentium 4 2.0 GHz CPU with 512 Kbytes of L2 cache and 512 Mbytes of DDR SDRAM, and they are interconnected via Fast Ethernet.

5.1 How to Use the PM Algorithm

Here we give a simple distribution problem, and then use our method to find the solution.

$$C^1 = \begin{pmatrix} 8 & 7 & 10 & 1 & 4 \\ 7 & 11 & 16 & 0 & 5 \\ 2 & 7 & 6 & 19 & 15 \\ 3 & 6 & 4 & 7 & 11 \\ 14 & 5 & 7 & 3 & 2 \end{pmatrix} \quad C^2 = \begin{pmatrix} 10 & 7 & 10 & 5 & 11 \\ 3 & 19 & 5 & 3 & 13 \\ 2 & 18 & 9 & 0 & 1 \\ 13 & 3 & 7 & 5 & 12 \\ 7 & 4 & 6 & 15 & 3 \end{pmatrix} \qquad q = 1,2.$$

Find an X satisfying

$$\begin{cases} \min : z^q(X) = (C^q)^T X = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij}^q x_{ij} & q = 1,2 \\ s.t. \quad \sum_{i=1}^{5} x_{ij} = 1 & j = 1,2,\cdots,5 \\ \quad \sum_{j=1}^{5} x_{ij} = 1 & i = 1,2,\cdots,5 \end{cases}$$

We use the PM algorithm to solve this distribution problem.

Step 1. Initialization: $(t = 0)$

$$X = \begin{pmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{pmatrix}$$

x_{ij} is also initialized as a random number between 0 and 1. Based on some experiments we have done, we found that the results are not affected by the initialization of X.

Standardization of C^1 and C^2:

$$C^1 = \begin{pmatrix} 0.80 & 0.70 & 1.00 & 0.10 & 0.40 \\ 0.44 & 0.69 & 1.00 & 0 & 0.31 \\ 0.11 & 0.37 & 0.32 & 1.00 & 0.79 \\ 0.27 & 0.55 & 0.36 & 0.64 & 1.00 \\ 1.00 & 0.36 & 0.50 & 0.21 & 0.14 \end{pmatrix} \quad C^2 = \begin{pmatrix} 0.91 & 0.64 & 0.91 & 0.45 & 1.00 \\ 0.16 & 1.00 & 0.26 & 0.16 & 0.68 \\ 0.11 & 1.00 & 0.50 & 0 & 0.06 \\ 1.00 & 0.23 & 0.54 & 0.38 & 0.92 \\ 0.47 & 0.27 & 0.40 & 1.00 & 0.20 \end{pmatrix}$$

1. According to Eq. 1,$z^q(X) = (C^q)^T X = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij}^q x_{ij}$,

we get

$$z^1 = 2.6120, \quad z^2 = 2.6500$$

2. According to Eq. 6, $u_{Oi}^{q} = 1 - \exp\left(-\sum_{j=1}^{5} c_{ij}^{q} \cdot x_{ij}\right)$, we compute in parallel and get

$$u_{O1}^{1} = 0.4512 \quad u_{O2}^{1} = 0.3861 \quad u_{O3}^{1} = 0.4043 \quad u_{O4}^{1} = 0.4311 \quad u_{O5}^{1} = 0.3573$$
$$u_{O1}^{2} = 0.5425 \quad u_{O2}^{2} = 0.3636 \quad u_{O3}^{2} = 0.2839 \quad u_{O4}^{2} = 0.4588 \quad u_{O5}^{2} = 0.3737$$

3. According to Eq. 9, $u_{Sj}^{q} = 1 - \exp\left(-\sum_{i=1}^{5} c_{ij}^{q} \cdot x_{ij}\right)$, we compute in parallel and get

$$u_{S1}^{1} = 0.4079 \quad u_{S2}^{1} = 0.4137 \quad u_{S3}^{1} = 0.4706 \quad u_{S4}^{1} = 0.3229 \quad u_{S5}^{1} = 0.4102$$
$$u_{S1}^{2} = 0.4114 \quad u_{S2}^{2} = 0.4663 \quad u_{S3}^{2} = 0.4067 \quad u_{S4}^{2} = 0.3283 \quad u_{S5}^{2} = 0.4356$$

Step 2. Compute in parallel:
The first evolutionary iteration $(t = 1)$:

1. According to Eq. (4), we have

$$\Delta x_{ij} \approx \frac{dx_{ij}}{dt} = -\sum_{q=1}^{2}\left(\lambda_1^q \frac{\partial u_{Oi}^q}{\partial x_{ij}} + \lambda_2^q \frac{\partial F_O^q}{\partial x_{ij}}\right) - \lambda_3 \frac{\partial I_O}{\partial x_{ij}}$$
$$= -\lambda_1^1 \frac{\partial u_{Oi}^1}{\partial x_{ij}} - \lambda_2^1 \frac{\partial F_O^1}{\partial x_{ij}} - \lambda_1^2 \frac{\partial u_{Oi}^2}{\partial x_{ij}} - \lambda_2^2 \frac{\partial F_O^2}{\partial x_{ij}} - \lambda_3 \frac{\partial I_O}{\partial x_{ij}}$$

where $\frac{\partial u_{Oi}^q}{\partial x_{ij}} = c_{ij}^q \cdot \exp\left(-\sum_{j=1}^{J} c_{ij}^q \cdot x_{ij}\right)$

$$\frac{\partial F_O^q}{\partial x_{ij}} = \frac{\partial F_O^q}{\partial u_{Oi}^q} \cdot \frac{\partial u_{Oi}^q}{\partial x_{ij}} = (k_O^q)^2 \cdot \frac{\exp\left\{(u_{Oi}^q)^2/[2(k_O^q)^2]\right\} \cdot [(u_{Oi}^q)/(k_O^q)^2]}{\sum_{i=1}^{5} \exp\left\{(u_{Oi}^q)^2/[2(k_O^q)^2]\right\}} \cdot \frac{\partial u_{Oi}^q}{\partial x_{ij}}$$

$$\frac{\partial I_O}{\partial x_{ij}} = 2Ja_1 \sum_{i=1}^{I}\left(\sum_{j=1}^{J} x_{ij} - 1\right) + 2Ia_2 \sum_{j=1}^{J}\left(\sum_{i=1}^{I} x_{ij} - 1\right)$$

2. According to Eq. (5), we have

$$\Delta c_{ij}^q \approx \frac{dc_{ij}^q}{dt} = -\left(\gamma_1^q \frac{\partial u_{Sj}^q}{\partial c_{ij}^q} + \gamma_2^q \frac{\partial F_S^q}{\partial c_{ij}^q}\right) - \gamma_3 \frac{\partial I_S}{\partial c_{ij}^q}$$

where $\frac{\partial u_{Sj}^q}{\partial c_{ij}^q} = x_{ij} \cdot \exp\left(-\sum_{i=1}^{I} c_{ij}^q \cdot x_{ij}\right)$

$$\frac{\partial F_S^q}{\partial c_{ij}^q} = \frac{\partial F_S^q}{\partial u_{Sj}^q} \cdot \frac{\partial u_{Sj}^q}{\partial c_{ij}^q} = (k_S^q)^2 \cdot \frac{\exp\left\{(u_{Sj}^q)^2/[2(k_S^q)^2]\right\} \cdot [(u_{Sj}^q)^2/(k_S^q)^2]}{\sum_{j=1}^{5} \exp\left\{(u_{Sj}^q)^2/[2(k_S^q)^2]\right\}} \cdot \frac{\partial u_{Sj}^q}{\partial c_{ij}^q}$$

$$\frac{\partial I_S}{\partial c_{ij}^q} = 0$$

3. In addition,

$$x_{ij}(t=1) = x_{ij}(t=0) + \Delta x_{ij}(t=1)$$
$$c_{ij}^1(t=1) = c_{ij}^1(t=0) + \Delta c_{ij}^1(t=1)$$
$$c_{ij}^2(t=1) = c_{ij}^2(t=0) + \Delta c_{ij}^2(t=1)$$
$$\lambda_1^1 = 0.05 \quad \lambda_2^1 = 0.05 \quad \lambda_1^2 = 0.05 \quad \lambda_2^2 = 0.05 \quad \lambda_3 = 0.01$$
$$\gamma_1^1 = 0.05 \quad \gamma_2^1 = 0.05 \quad \gamma_1^2 = 0.05 \quad \gamma_2^2 = 0.05 \quad \gamma_3 = 0.01$$
$$a_1 = 0.5 \quad a_2 = 0.5 \quad k_O^1 = k_O^2 = k_S^1 = k_S^2 = 0.8$$

As for these coefficients, we can draw the following conclusions from the experiments we have done.

- When k_O^q (or k_S^q) is larger, the corresponding convergence speed is faster.
- If the values of λ and γ change in direct proportion, the experimental results will hardly be influenced.
- If we increase λ_1^q, λ_2^q and do not touch the other coefficients, the q-th objective will take precedence over all the other objectives.

We compute in parallel and get

$$X(t=1) = \begin{pmatrix} 0.1890 & 0.2007 & 0.1819 & 0.2277 & 0.2006 \\ 0.2133 & 0.1708 & 0.1881 & 0.2301 & 0.1978 \\ 0.2246 & 0.1766 & 0.2001 & 0.1968 & 0.2019 \\ 0.1973 & 0.2138 & 0.2100 & 0.2054 & 0.1736 \\ 0.1778 & 0.2109 & 0.2003 & 0.1886 & 0.2223 \end{pmatrix}$$

$$C^1(t=1) = \begin{pmatrix} 0.7751 & 0.6784 & 0.9719 & 0.0965 & 0.3876 \\ 0.4263 & 0.6687 & 0.9719 & 0 & 0.3004 \\ 0.1066 & 0.3586 & 0.3110 & 0.9648 & 0.7655 \\ 0.2616 & 0.5330 & 0.3499 & 0.6175 & 0.9690 \\ 0.9688 & 0.3489 & 0.4860 & 0.2026 & 0.1357 \end{pmatrix}$$

$$C^2(t=1) = \begin{pmatrix} 0.8818 & 0.6219 & 0.8816 & 0.4343 & 0.9702 \\ 0.1550 & 0.9717 & 0.2519 & 0.1544 & 0.6597 \\ 0.1066 & 0.9717 & 0.4844 & 0 & 0.0582 \\ 0.9690 & 0.2235 & 0.5231 & 0.3667 & 0.8926 \\ 0.4554 & 0.2624 & 0.3875 & 0.9650 & 0.1940 \end{pmatrix}$$

$$z^1(t=1) = 2.5243, \qquad z^2(t=1) = 2.5590$$

Obviously, $z^1(t=1) < z^1(t=0)$ and $z^2(t=1) < z^2(t=0)$, and the distribution problem is optimized.

The evolutionary experimental results and the optimization trend from $t=0$ to $t=18$ are shown in Fig. 4 and Fig. 5.

As shown in Fig. 4 and Fig. 5, the two-objective distribution problem is optimized by the evolution of the PM algorithm. The convergence speed is faster at the beginning of evolution. The optimization trend of z^1 and z^2 reflects exactly the optimization of the problem, that is, the distribution problem is optimized step by step.

5.2 Effectiveness of the PM Algorithm

We pick 320×200-scale combinatorial optimization problems with five objectives to test our PM algorithm. Of course, for larger scales and problems of more objectives, our PM algorithm can work out the optimum solutions quickly too.

The problem-related matrixes C^1, C^2, C^3, C^4, C^5 are randomly generated. We let

(1) $\lambda_1^q, \lambda_2^q (q = \overline{1,5})$ be 0.05;
(2) $\lambda_3 = 0.01$;
(3) $\gamma_1^q, \gamma_2^q (q = \overline{1,5})$ be 0.05;
(4) $\gamma_3 = 0.01$;

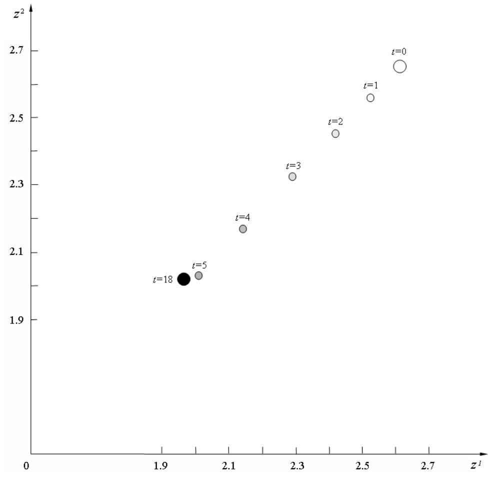

Fig. 4. Optimization from $t = 0$ to $t = 18$

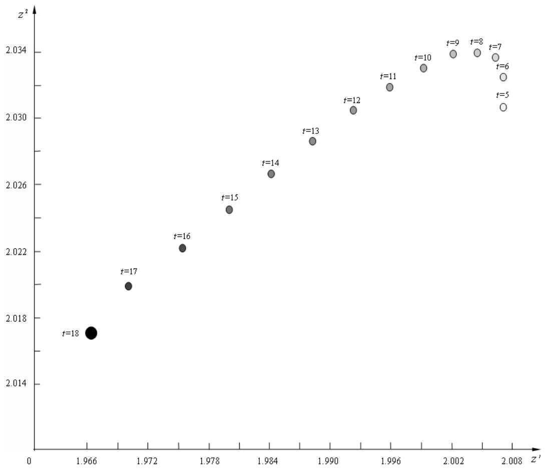

Fig. 5. Optimization from $t = 5$ to $t = 18$

(5) a_1=0.5 and a_1=0.5;

(6) $k_O^q, k_S^q (q = \overline{1,5})$ be 0.8.

Because the problem-related matrices are too large to list here, we give the results of the problem directly. The evolutionary experimental results for z^1, z^2, z^3, z^4, z^5 are depicted in Fig. 6.

We use 16 computing modes of the cluster (mentioned at the beginning of this section) to compute the solution. More data about the solution are shown in Table 2.

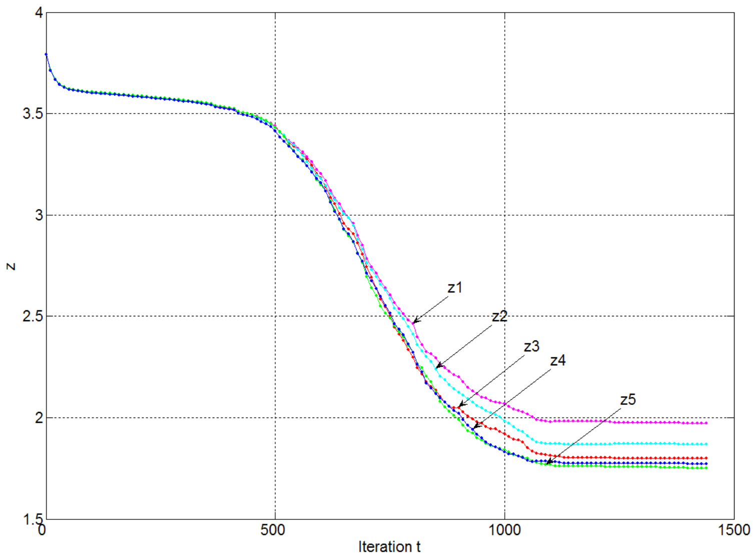

Fig. 6. The optimization trend of five objectives in a large-scale problem

Table 2. Other solution-related data

Objectives	5
Processors	16
Scale	320×200
Max iterations	50000
Convergence iterations	1447
Convergence time (second)	49.934
$z(t = 1447)$	9.16125010
$z(t = 1448)$	9.16125010

As shown in Fig. 6, the five optimization curves of z^1, z^2, z^3, z^4, z^5 reflect exactly the optimization of the problem and five objectives. The convergence speeds of five objectives are fastest at the beginning (about from $t = 0$ to $t = 20$) and faster within the medium range. At $t = 1447$, $z(= z^1 + z^2 + z^3 + z^4 + z^5)$ reaches its minimum (9.16125010), and stays unchanged in the remainder of the iterations. For this 320×200-scale problem, PM algorithm converges to a stable equilibrium state when $t = 1447$. Undoubtedly this solution is the optimum solution as it minimizes $z(z^1, z^2, z^3, z^4, z^5)$, which verifies the approach's convergence and its ability to arrive at the optimum for large-scale many-objective problems.

In Table 2, "Processors" represents the number of computing nodes in the cluster we used to compute the solution. We use the number of rows and columns of distribution matrix (X) to represent the "scale" of the problem. "Convergence time and iterations" represent the actual time and iterations used to compute the solution (a stable equilibrium) on the cluster. Considering the situation that, after many iterations, stable equilibrium is still not reached by the PM algorithm, we should end the computation at this time. When $t =$"Max iterations", the PM algorithm ends. z is aggregated by z^1, z^2, z^3, z^4, z^5, representing the optimization of the whole problem.

5.3 Efficiency and Parallelism of the PM Algorithm

Mimetic algorithms (MAs) provide a valuable alternative to traditional methods because of their inherent parallelism and their ability to deal with difficult problems. Our PM approach as a new branch of MA, has these two main advantages too. The distribution and weight variables, x_{ij} and c_{ij}^q, can be computed and updated in parallel without any information exchange, which is the foundation of PMA's parallelism.

In order to test the high parallelism and good scalability of the PM algorithm, we compute multi-objective problems of different scales both in the parallelized way and the non-parallelized way. The hardware and software environment in which these simulations are run have been mentioned at the beginning of this section.

Convergence time and iterations for different multi-objective problems using PMA are shown in Table 3 and Table 4. In the two Tables, the data can be categorized into two

Table 3. Convergence time and iterations of PMA for multi-objective medium-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
5	32×4	0.085	101	0.296	101	0.516	101	0.705	99
5	32×8	0.167	102	0.295	100	0.509	95	0.747	105
5	32×12	0.272	101	0.289	99	0.505	101	0.729	100
5	32×16	0.365	102	0.337	98	0.516	102	0.738	100
5	32×20	0.436	105	0.394	108	0.63	104	0.77	105
5	32×24	0.611	113	0.453	110	0.607	111	0.786	108
5	32×28	0.672	115	0.473	116	0.645	114	0.864	123
5	32×32	0.842	121	0.51	124	0.744	120	0.881	125
5	32×36	1.059	140	0.568	132	0.737	139	0.878	126
5	32×40	1.087	133	0.664	139	0.819	145	1.16	153
5	32×44	1.333	147	0.807	159	0.853	154	1.01	144
5	32×48	1.866	183	0.805	161	0.916	161	1.047	149
5	32×52	1.74	161	0.87	171	0.962	174	1.268	189
5	32×56	2.367	201	0.995	189	1.135	195	1.285	190
5	32×60	2.753	217	1.111	189	1.292	222	1.242	183
5	32×64	2.851	222	1.516	256	1.239	213	1.401	218
5	32×68	2.915	206	1.386	229	1.325	226	1.601	239
5	32×72	3.329	220	1.342	217	1.465	250	2.085	267
5	32×76	3.737	238	1.701	268	1.667	283	1.865	286
5	32×80	5.304	318	2.005	302	1.613	263	1.897	287

Table 4. Convergence time and iterations of PMA for multi-objective large-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
5	320×40	12.81	153	3.702	159	2.39	160	1.90	166
5	320×80	55.26	324	14.161	316	7.92	316	5.74	348
5	320×120	136.11	537	39.516	580	19.65	554	12.62	602
5	320×160	289.79	851	79.355	907	43.05	933	25.87	973

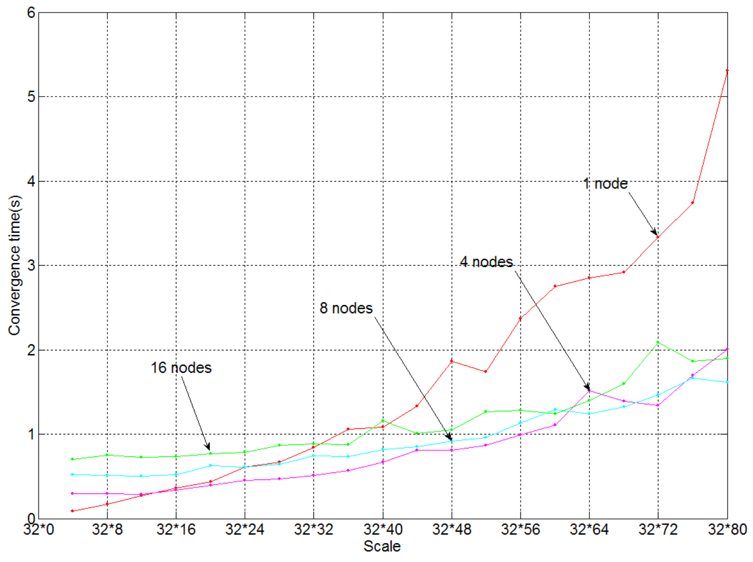

Fig. 7. Convergence time of PMA for multi-objective medium-scale problems

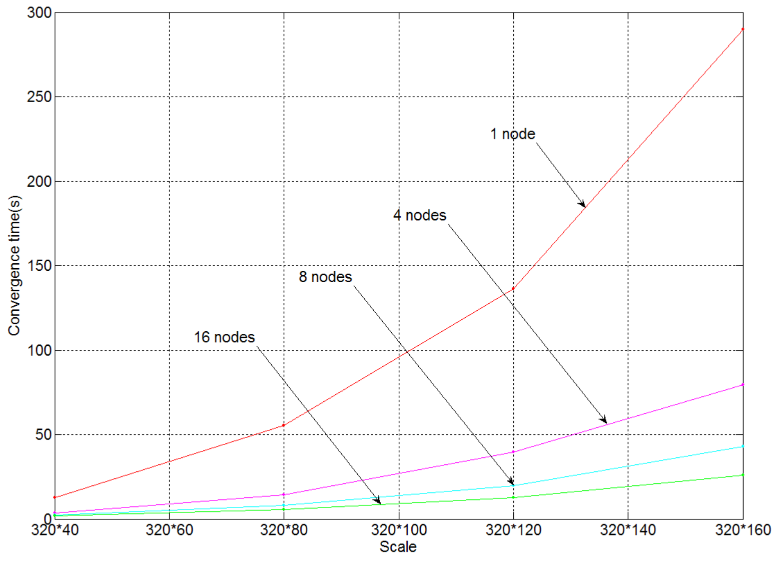

Fig. 8. Convergence time of PMA for multi-objective large-scale problems

parts. One part is related to the sequential version, which comes from our experimental results using one computing node of the cluster. The other part is related to the parallel version, which comes from the results using 4, 8 and 16 computing codes of the cluster. “Iterations” and “time” are the iterations and time PMA takes to converge.

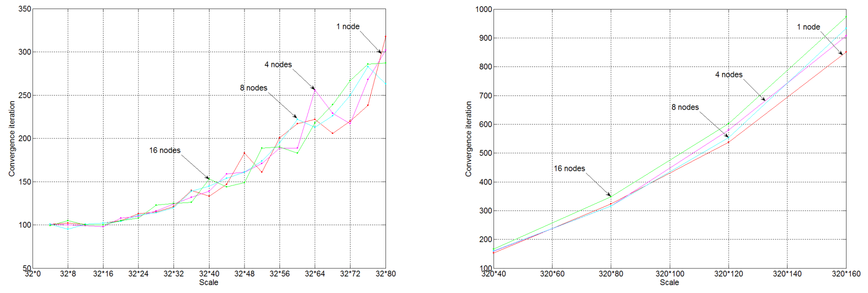

Fig. 9. Convergence speeds of PMA for multi-objective problems

As shown in Fig. 7 and Fig. 8, the convergence time of the sequential version increases exponentially with the scale, which is similar to all the exact methods. When parallelized, the convergence time drops significantly across the larger scale problems. The times for the smaller scale problems are dominated by the message exchange time, and the sequential version appears to be more efficient in that range. For the large-scale problems (see Fig. 8), the more computing nodes of the cluster we used, the less actual times used to compute the solutions.

As shown in Fig. 9, the convergence speeds (in terms of number of iterations to convergence) of the sequential version and parallel version increase steadily with the scale. The convergence speed of PMA is almost not related to the computing version (sequential version or parallel version).

5.4 Problems with Different Numbers of Objectives by PMA

Here we test the effectiveness of our PM algorithm to solve problems with different numbers of objectives (from single objective to many-objective problems). Convergence time and speeds using PMA for different numbers of objectives are shown in Table 5 and Table 6. Table 5 and Fig. 10 show the relation between the convergence time, the number of objectives and computing version (sequential or parallel) when the scale of the problems is fixed (at 80×80). Table 6 and Fig. 11 show the relation between the convergence time, the number of objectives and the scale when the computing version is fixed (parallel version using 16 computing nodes of the cluster).

As shown in Fig. 10, the convergence time increases steadily when the number of objectives of the problem increases. The more computing nodes of the cluster are used to compute the solutions, the slower the convergence time's increase with the number of objectives. The experimental results verify the good parallelism of our PM approach for multi-objective optimization problems.

As shown in Fig. 11, the more the objectives, the faster the convergence time would increase with the scale. The experimental results verify that our PM approach can deal with many-objective, large-scale optimization problems.

Table 5. Convergence time and speeds of PMA for different numbers of objectives, medium-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
1	80×80	4.69	539	1.57	526	1.35	536	1.47	531
5	80×80	13.00	308	3.92	305	2.77	297	2.40	280
10	80×80	25.25	298	7.80	310	4.80	283	4.49	288
15	80×80	36.69	292	11.28	297	7.53	296	6.92	313
20	80×80	51.90	308	16.02	319	11.22	331	10.37	335
25	80×80	70.70	336	21.07	326	14.10	324	12.47	344
30	80×80	81.58	323	24.19	319	17.42	342	12.79	308
35	80×80	101.45	338	31.46	355	19.85	338	17.39	361
40	80×80	123.20	367	36.58	362	24.09	360	19.55	355
45	80×80	141.15	373	40.62	354	26.76	347	21.91	343
50	80×80	157.11	373	49.10	382	29.26	342	25.41	353
55	80×80	181.71	386	52.29	372	34.65	376	28.76	379
60	80×80	186.66	369	58.20	383	37.07	364	31.30	380
65	80×80	221.14	399	67.81	381	41.55	376	36.08	397
70	80×80	237.41	404	69.53	416	46.24	391	39.32	402
75	80×80	251.73	396	75.90	402	50.80	380	41.70	409
80	80×80	278.09	409	83.24	405	52.46	416	43.63	389
85	80×80	313.88	440	85.04	397	56.10	387	47.35	406
90	80×80	345.14	456	86.90	375	63.68	422	52.20	426

Table 6. Convergence time and speeds of PMA for different numbers of objectives, large-scale problems using 16 computing nodes of the cluster

Scale	1 objective		5 objectives		50 objectives	
	time(s)	iterations	time(s)	iterations	time(s)	iterations
320×40	0.983	242	1.90	166	16.767	176
320×80	2.583	600	5.74	348	58.827	393
320×120	5.453	1046	12.62	602	169.793	816
320×160	10.8	1759	25.87	973	343.651	1312
320×200	15.868	2227	42.034	1293	569.017	1832
320×240	29.419	3327	63.617	1722	1044.953	2842
320×280	38.901	4159	99.039	2276	1571.968	3711
320×320	63.718	5711	130.258	2679	2113.836	4351
320×360	84.762	6806	183.987	3410	3358.083	5839
320×400	118.984	8404	247.028	4122	4276.371	6758
320×440	155.671	9816	336.842	4988	5839.789	8245
320×480	262.693	12544	446.898	5834	8130.714	9963
320×520	307.017	13909	578.865	6845	9729.199	11014
320×560	374.298	15629	699.978	7718	13528.261	13064
320×600	425.01	17180	861.57	8609	15294.553	14319
320×640	594.781	19854	997.091	9528	18108.043	15921
320×680	698.079	21922	1641.718	11881	25035.577	18459
320×720	774.948	23927	1418.417	11523	27951.384	20024
320×760	987.354	26780	2122.651	13738	37178.713	22961
320×800	1332.217	30418	2196.459	14275	38786.457	24314

5.5 Performance Comparison between PMA and NSGA-II

In this subsection, we compare the proposed algorithm, PMA, with NSGA-II [12] for a number of two-objective distribution problems which are similar to the example problem mentioned in subsection 5.1. For the NSGA-II, we use a standard real-parameter SBX and polynomial mutation operator with $\eta_c = 10$ and $\eta_m = 10$, respectively. We run

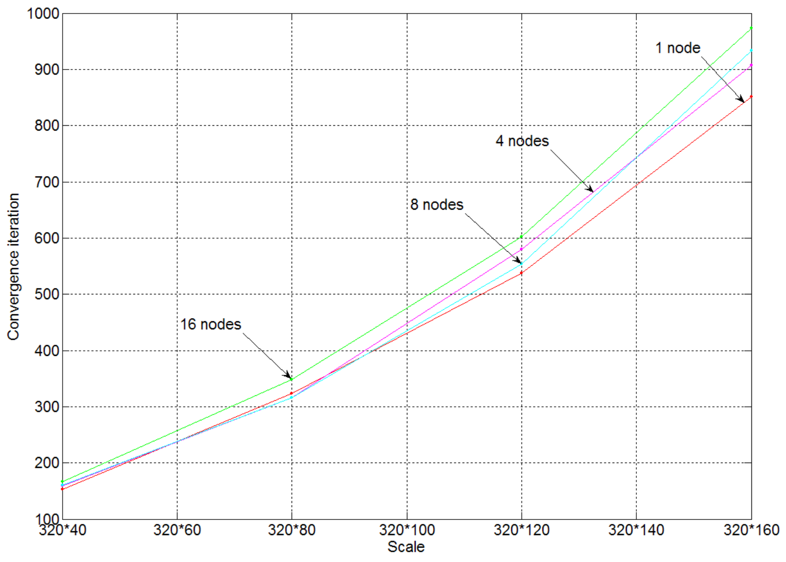

Fig. 10. Convergence time of PMA for different numbers of objectives, medium-scale problems

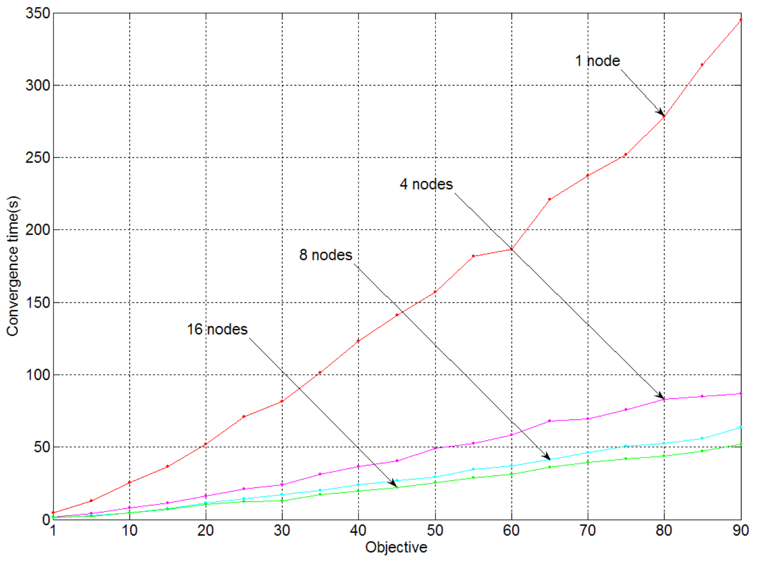

Fig. 11. Convergence time of PMA for different numbers of objectives, large-scale problems using 16 computing nodes of the cluster

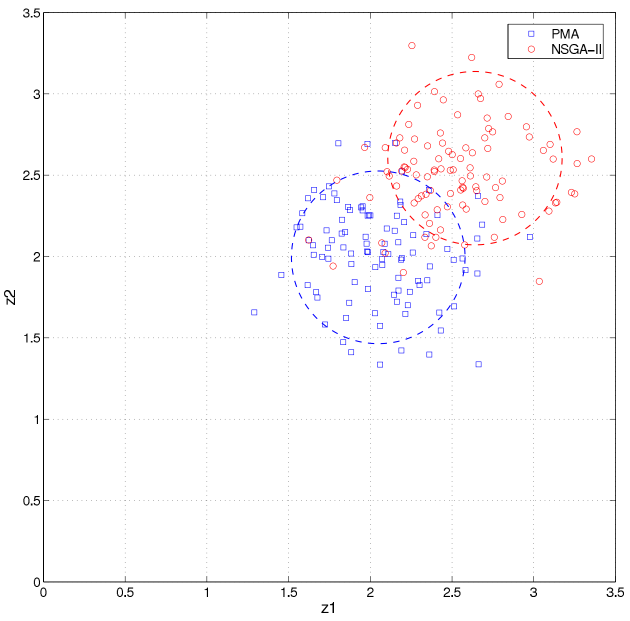

Fig. 12. The two solution spaces of PMA and NSGA-II respectively for 100 random two-objective problems

each algorithm for 100 random two-objective problems (using the same 100 problems for both); the results are presented in Fig. 12.

As marked by the two rings in Fig. 12, the solution space of PMA is closer to the origin than that of NSGA-II. From the experimental results, it can be seen that PMA has better performance than the normal NSGA-II for two-objective distribution problems.

5.6 Comparison between PMA and Other Benchmark MAs (NAs)

As mentioned in Section 1, popular mimetic algorithms (MAs) (or nature-inspired approaches (NAs)) include genetic algorithm (GA), simulated annealing algorithm (SA), ant colony optimization (ACO), particle swarm optimization (PSO), etc. The proposed

Table 7. Common features between PMA and other benchmark MAs

Aspects	Common features
Drawn from	Observations of physical processes that occur in nature
Belong to	The class of meta-heuristics, approximate algorithms
Parallelism	Have inherent parallelism
Performance	Consistently perform well
Fields of application	Artificial intelligence including multi-objective optimization
Solvable problems	All kinds of difficult problems

Table 8. Relative differences between PMA and other benchmark MAs

	PMA	GA	SA	ACO	PSO
Inspired by	Physical models of particle dynamics	Natural evolution	Thermodynamics	Behaviors of real ants	Biological swarm (e.g., swarm of bees)
Key components	Energy function; differential dynamic equations	Chromosomes	Energy function	Pheromone laid	Velocity-coordinate model
Exploration	Both Macro-evolutionary and Micro-evolutionary processes	Macro-evolutionary processes	Micro-evolutionary processes	Macro-evolutionary processes	Macro-evolutionary processes
Dynamics	Can capture the entire dynamics inherent in the problem	Cannot capture	Can capture partly	Cannot capture	Cannot capture
High-dimensional, highly nonlinear, random behaviors and dynamics	Can describe	Cannot describe	Can describe partly	Cannot describe	Cannot describe
Adaptive to problem changes	Fast	Middle	Fast	Low	Middle
Exchange overhead	Low	Middle	Low	Low	Middle

PMA and these benchmark MAs share some common features, which are listed in Table 7.

We also summarize the relative differences between our PMA and the benchmark MAs in Table 8.

6 Conclusion

In this chapter, we propose a novel mimetic approach—particle mechanics algorithm (PMA) for solving multi-objective distribution problems, which is based on the theory of particle mechanics. The approach maps a given distribution problem to the movement of particles in a multi-dimensional space in a pair of (dual) groups of force-fields. The particles move according to certain rules defined by a mathematical model until arriving at a stable state; subsequently, the solution of the multi-objective distribution problem is obtained by anti-mapping the stable state.

Although there are many differences between particles in classical mechanics and those in PMA, we have shown that being inspired by classical mechanics, PMA enables feasible many-objective optimization of problems of very large scale. The PM approach has a low computational complexity, which is crucial for the functioning of large-scale distribution problems.

References

1. Ulungu, E.L., Teghem, J.: The two phases method: an efficient procedure to solve bi-objective combinatorial optimization problems. Journal Foundations of Computing & Decision Sciences 20(2), 149–165 (1995)
2. Tuyttens, D., Teghem, J., Fortemps, P., Van Nieuwenhuyse, K.: Performance of the MOSA method for the bicriteria assignment problem. Journal of Heuristics 6, 295–310 (2000)
3. Pirlot, M.: General local search methods. Europe Journal of Operational Research 92, 493–511 (1996)
4. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
5. Forrest, S.: Genetic algorithms—principles of natural-selection applied to computation. Science 261(5123), 872–878 (1993)
6. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature 406(6791), 39–42 (2000)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Conf. Neural Networks, Piscataway, NJ, vol. IV, 1942–1948 (1995)
8. Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method. Nature 326(6114), 689–691 (1987)
9. Shuai, D., Feng, X.: Distributed Problem Solving in Multi-Agent Systems: A Spring Net Approach. IEEE Intelligent Systems 20(4), 66–74 (2005)
10. Shuai, D., Feng, X.: The Parallel Optimization of Network Bandwidth Allocation Based on Generalized Particle Model. Computer Networks 50(9), 1219–1246 (2006)
11. Le Boudec, J.-Y.: Rate adaptation, Congestion Control and Fairness: A Tutorial (November 22, 2005)
12. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: Afast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)