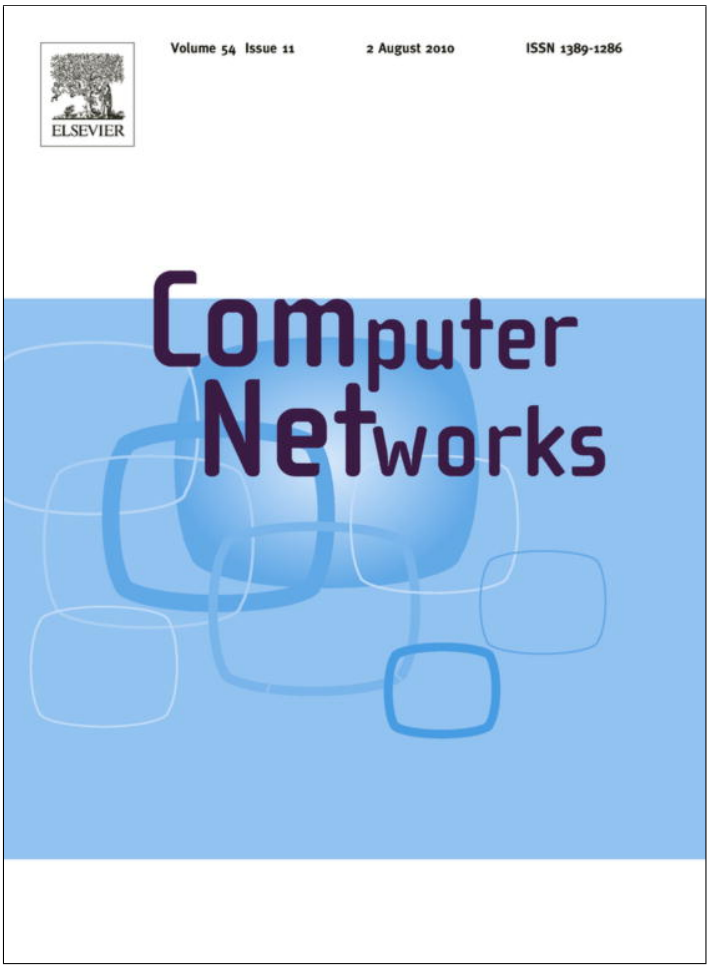

Computer Networks 54 (2010) 1767–1777

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Parallel physics-inspired waterflow particle mechanics algorithm for load rebalancing

Xiang Feng [a,*], Francis C.M. Lau [b]

[a] Department of Computer Science, East China University of Science and Technology, PR China
[b] Department of Computer Science, The University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 26 November 2008
Received in revised form 9 November 2009
Accepted 5 February 2010
Available online 10 February 2010
Responsible editor: C.B. Westphall

Keywords:
Load rebalancing
Approximation algorithm
Nature-inspired algorithm
Waterflow particle mechanics model
Distributed and parallel algorithm

ABSTRACT

The Load Rebalancing Problem (LRP) that reassigns tasks to processors so as to minimize the maximum load arises in the context of dynamic load balancing. Many applications such as on Web based environment, parallel computing on clusters can be stated as LRP. Solving LRP successfully would allow us to utilize resources better and achieve better performance. However LRP has been proven to be NP-hard, thus generating the exact solutions in tractable amount of time becomes infeasible when the problems become large. We present a new nature-inspired approximation algorithm based on the Waterflow Particle Mechanics (W-PM) model to compute in parallel approximate efficient solutions for LRPs. Just like other Nature-inspired Algorithms (NAs) drawing from observations of physical processes that occur in nature, the W-PM algorithm is inspired by kinematics and dynamics of waterflow. The W-PM algorithm maps the classical LRP to the flow of water flows in channels by corresponding mathematical model in which all water flows flow according to certain defined rules until reaching a stable state. By anti-mapping the stable state, the solution to LRP can be obtained.

1. Introduction

With the Internet assuming an ever more central role in the telecommunications infrastructure, Web servers are becoming increasingly important. Applications that handle heavy loads commonly use a cluster-based architecture for Web servers because it combines low cost with good performance.

Recently, several Load Rebalancing assignment policies have been proposed [1–4]. According to two main strategies, these policies attempt to balance the load among back-end servers:

(1) balancing the amount of workload at back-end servers, and
(2) balancing the number of jobs being processed by back-end servers.

The problem studied in this paper focuses on dynamically assigning resources in an ad hoc grid to an application composed of communicating subtasks. We propose the Waterflow Particle Mechanics (W-PM) model and algorithm for balancing the amount of workload.

Well-known policies for balancing the amount of workload include Dynamic [5] and Size-Range [6–8]. Under Dynamic, the dispatcher assigns an incoming job to the back-end server with the smallest amount of residual workload.

The W-PM algorithm is inspired by physical models of waterflow and particle dynamics. The W-PM algorithm is easy to use in spite of its seemingly abstruse theories and sinuate motivation.

2. Problem model for LRP

The dynamic task scheduling considering the load balancing issue is an NP-hard problem [9,10]. The grid has n

* Corresponding author. Present address: Meilong Road 130, Shanghai 200237, PR China, Tel.: +86 21 64253471 103.
E-mail addresses: xfeng@ecust.edu.cn, xfeng@cs.hku.hk (X. Feng), fcmlau@cs.hlu.hk (Francis C.M. Lau).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.02.002

processors and m tasks. Task i (T_i) has n subtasks, $r_{ij}(j = 1, \ldots, n)$. r_{ij} represents the subtask of task T_i that is mapped to processor j. Each subtask r_{ij} can be executed in processor j.

The Load Balancing Problem (LRP)

Given:

- A network of processors, A_j; a processor has two attributes: its processing capacity x_j (in task units per second) and its "remaining workload", w_j (in task units).
- The bandwidth of the communication channel C_{jl} between any two processors (A_j and A_l), b_{jl} (in bytes/s) (all pairs are connected).
- A set of tasks, T_i, that are spread across these processors in a given "initial mapping"; a task has an attribute: its "size", s_i (in bytes).

Goal:

To remap the tasks to the processors so that the following are minimized. (During mapping, some tasks will be migrated from their current processors to other processors.)

- The execution time, which is the maximum of the individual execution times after the remapping.
- The migration time, which is the maximum of all individual migration times.

The main notations in any LRP instance are shown as follows.

A_j	jth processor ($j = \overline{1,n}$)
T_i	ith task ($i = \overline{1,m}$)
C_{lj}	communication channel between processors A_l and A_j ($l = \overline{1,n}$)
r_{ij}	subtask of task T_i that is mapped to processor A_j
$\triangle r_{ij}(t)$	increment of subtask r_{ij} at time t
b_{lj}	bandwidth of the communication channel C_{lj}
x_j	processing capacities of processor A_j
w_j	remaining workload of processor $A_j, w_j = \sum_{i=1}^{m} r_{ij}$
s_i	size of task T_i
e_{ij}	execution time of subtask $r_{ij}, e_{ij} = r_{ij}/x_j$
e_j	execution time of processor A_j, $e_j = w_j/x_j = \sum_{i=1}^{m} e_{ij}$
$q_{ilj}(t)$	migration time about task T_i from processor A_l to A_j at time t, $q_{ilj}(t) = \triangle r_{il}(t)/b_{lj}$
$q_{ij}(t)$	migration time about task T_i from processor A_* to A_j at time t, $q_{ij}(t) = \max_l q_{ilj}$ ($l = \overline{1,n}$)
$q_j(t)$	migration times on processor A_j, $q_j(t) = \sum_{i=1}^{m} q_{ij}(t)$

We can formalize LRP using a matrix Λ(see Table 1).

The $m \times n$ computing cells of LRP are shown in Table 2. In Table 2, r_{ij} is the main variable in the computing cells. In our W-PM algorithm, $m \times n$ r_{ij} will evolve in parallel until W-PM algorithm converges ($t = end$). r_{ij}, is the key of LRP. If an algorithm can compute and update r_{ij} in parallel without any information exchange, the algorithm has a change to solve LRP in parallel.

3. The parallel computing architecture of W-PM

The parallel computing architecture of the W-PM, as shown in Fig. 1, is composed of four computing cell arrays, $\mathfrak{C}, \mathfrak{C}_{row}, \mathfrak{C}_{col}$, and $\mathfrak{C}_{globle}$,whose computing cells are denoted by $\mathfrak{C}_{ij}, \mathfrak{C}_{i*}, \mathfrak{C}_{*j}$, and $\mathfrak{C}_{**}$,respectively.

The number of computing cells in each array is equal to: $m \times n$ for $\mathfrak{C}$, m for $\mathfrak{C}_{row}$, n for $\mathfrak{C}_{col}$, and 1 for $\mathfrak{C}_{globle}$, respectively, and hence the total number of computing cells equals $m \times n + m + n + 1$. There is no interconnection among computing cells in the same array, whereas there are local interconnections between the following computing cell pairs: $\mathfrak{C}_{ij}$ and $\mathfrak{C}_{i*}$; $\mathfrak{C}_{ij}$ and $\mathfrak{C}_{*j}$; $\mathfrak{C}_{i*}$ and $\mathfrak{C}_{**}$; $\mathfrak{C}_{*j}$ and $\mathfrak{C}_{**}$. It is obvious that the connection degree of each computing cell in the array $\mathfrak{C}$ of $m \times n$ computing cells is equal to at most 2, and the unique computing cell in $\mathfrak{C}_{globle}$ has connection degree $m + n$, with the total number of interconnections being $2m \times n + m + n$.

At time t in a fixed time slot ϱ, the computing cell $\mathfrak{C}_{ij}$ sends its dynamical state $\langle r_{ij}(t) \rangle$ to computing cells $\mathfrak{C}_{i*}$ and $\mathfrak{C}_{*j}$, and receives the feedback inputs that are generated by computing cells $\mathfrak{C}_{i*}$ and $\mathfrak{C}_{*j}$ at time $(t - \tau)$. By using the received $\langle r_{ij}(t) \rangle$, the computing cell $\mathfrak{C}_{i*}(\mathfrak{C}_{*j}$, resp.) obtains its calculation state $r_{i*}(t)(r_{*j}(t)$,resp.) according to the equation $r_{i*}(t) = \sum_j r_{ij}(t)(r_{*j}(t) = \sum_i r_{ij}(t)$, resp.); which yields its current output to be fed back to the computing cell $\mathfrak{C}_{ij}$. Meanwhile, the computing cell $\mathfrak{C}_{i*}$ and $\mathfrak{C}_{*j}$ receive the feedback from computing cell $\mathfrak{C}_{**}$. The computing cells, $\mathfrak{C}_{i*}, \mathfrak{C}_{*j}$, and $\mathfrak{C}_{**}$, will change their calculation states respectively. The computing cell $\mathfrak{C}_{ij}$ will change its dynamical state according to Eq. (13) (to be given in Section 5).

The implementation of W-PM algorithm can enjoy a high degree of parallelism and good scalability. All the computations of cellular dynamics both in the same array and in the different arrays are concurrently carried out. The computing cellular structure, computing cellular dynamics and algorithm are all independent of the problem scale. Moreover, there is no direct interconnection among computing cells in the same array, so it is relatively easier to implement the proposed structure in VLSI technology.

4. The motivation and architecture of W-PM model

As described in Section 2, LRP has two main characteristics.

(1) LRP aims to minimize two variables: the maximal execution times and the maximal migration times. The two goals are conflicting. It is very difficult to realize the two goal in parallel and in real-time.
(2) The LRP has been proved to be NP-hard [10]. Meanwhile The number of processors and tasks in LRP is very large. Thus we are unable to deliver an exact solution to LRP in a reasonable amount of time. For instance, if we have 5000 tasks and 10 processors that we could use, we have 10^{5000} configurations to enumerate and compare for the Load Balancing Problem (LBP). For LRP, if we restrict the number of moves to 8, then we would end up with approxi-

Table 1
The representation matrix Λ of LRP, with processors (A) and tasks (T).

	A_1	$\cdots$	A_j	$\cdots$	A_n	s_i
T_1	r_{11}, e_{11}, q_{11}	$\cdots$	r_{1j}, e_{1j}, q_{1j}	$\cdots$	r_{1n}, e_{1n}, q_{1n}	$s_1 = \sum_{j=1}^{n} r_{1j}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
T_i	r_{i1}, e_{i1}, q_{i1}	$\cdots$	r_{ij}, e_{ij}, q_{ij}	$\cdots$	r_{in}, e_{in}, q_{in}	$s_i = \sum_{j=1}^{n} r_{ij}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
T_m	r_{m1}, e_{m1}, q_{m1}	$\cdots$	r_{mj}, e_{mj}, q_{mj}	$\cdots$	r_{mn}, e_{mn}, q_{mn}	$s_m = \sum_{j=1}^{n} r_{mj}$
w_j	$w_1 = \sum_{i=1}^{m} r_{i1}$	$\cdots$	$w_j = \sum_{i=1}^{m} r_{ij}$	$\cdots$	$w_n = \sum_{i=1}^{m} r_{in}$	$\sum_{j=1}^{n} w_j = \sum_{i=1}^{m} s_i$

Table 2
The $m \times n$ computing cells of LRP.

r_{11}, e_{11}, q_{11}	$\cdots$	r_{1j}, e_{1j}, q_{1j}	$\cdots$	r_{1n}, e_{1n}, q_{1n}
$\vdots$		$\vdots$		$\vdots$
r_{i1}, e_{i1}, q_{i1}	$\cdots$	r_{ij}, e_{ij}, q_{ij}	$\cdots$	r_{in}, e_{i1}, q_{i1}
$\vdots$		$\vdots$		$\vdots$
r_{m1}, e_{m1}, q_{m1}	$\cdots$	r_{mj}, e_{mj}, q_{mj}	$\cdots$	r_{mn}, e_{mn}, q_{mn}

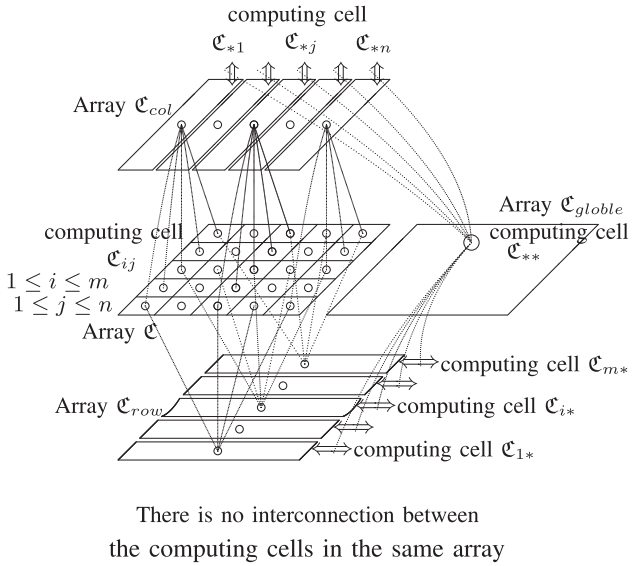

Fig. 1. The parallel computing architecture of W-PM algorithm.

mately 5000^8 configurations. While solving these problems is not feasible under such a situation, approximation and parallel algorithms could possibly be implemented to achieve a reasonable estimate within a fixed error ratio in a reasonable amount of time.

We will introduce two models: Waterflow model and Particle Mechanics model. By transforming LRP to the two models respectively, the two problems mentioned above will be solved.

4.1. Waterflow model

By analyzing the relation between the two conflict goals of LRP and observing the flow of water in some vessels which are connected by channels, we find that they have much in common. Water in vessels always flows from higher water level vessels to lower water level vessels through channels. Similarly, tasks waiting on processors should remap to the processors whose remaining workload is less. We transform LRP to Waterflow model (see Fig. 2).

We summarize the analogical relation between LRP and Waterflow model as follows.

LRP $\Longleftrightarrow$ Waterflow model
Processor $\Longleftrightarrow$ Vessel
Task $\Longleftrightarrow$ Water
Communication channel $\Longleftrightarrow$ Channel
Max–min fairness balancing of LRP $\Longleftrightarrow$ Potential energy of water
The flow of water among vessels $\Longleftrightarrow$ LRP optimization

The corresponding meanings in Waterflow model of main notations in LRP are shown as follows.

A_j	jth vessel ($j = \overline{1,n}$)
T_i	ith waterflow ($i = \overline{1,m}$)
C_{lj}	channel between vessels A_l and A_j ($l = \overline{1,n}$)
r_{ij}	subwaterflow of T_i in vessel A_j
$\triangle r_{ij}(t)$	the increment of water r_{ij} at time t
b_{lj}	width of the channel C_{lj}
x_j	size of the downspout on the bottom of vessel A_j
w_j	remaining water in vessel A_j
s_i	amount of waterflow T_i
e_{ij}	time that subwaterflow r_{ij} in vessel
A_j	flows down through the downspout

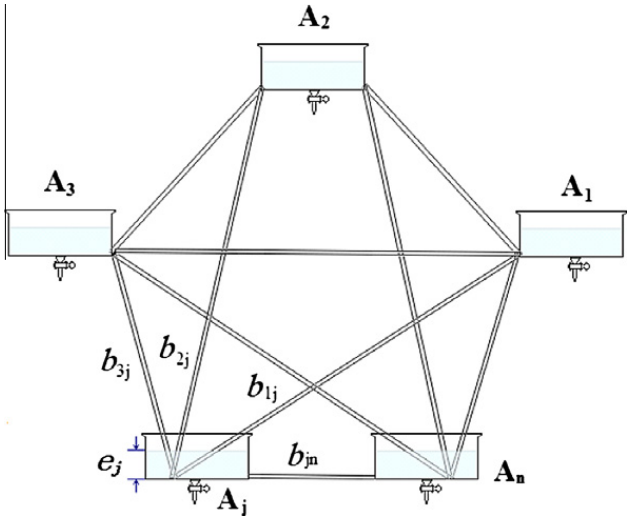

Fig. 2. The transformation of LRP to Waterflow model.

e_j time that all water in vessel A_j flows down through the downspout

$q_{ilj}(t)$ time that waterflow T_i flows through the channel from vessel A_l to A_j at time t

$q_{ij}(t)$ time that waterflow T_i flows through the channel from vessel A_* to A_j at time t

$q_j(t)$ migration time on vessel A_j

In Waterflow model, all vessels are evenly distributed. All pairs of vessels are connected by channels, which have different width. There is a downspout on the bottom of each vessel. Because the water level in vessels dynamically changes, the potential energy of water in vessels dynamically changes correspondingly. Water is being exerted upon simultaneously by potential energy to (1) flow down through downspouts on the bottom of vessels; (2) flow through channels between vessels. The time that water in a vessel flows down through downspout is in inverse proportion to the size of the downspout. The time that water flows through the channel between two vessels is in inverse proportion to the width of the channel. Water in vessels continuously flows down through downspouts, at the same time, potential energy makes water flow from higher water level vessels to lower water level vessels through channels. Obviously, when the maximum of e_j is equal to the maximum of q_j, the time that all water flow away vessels will be minimum.

Based on Waterflow model, we find a way to optimize the two conflict goals of LRP. As shown in Fig. 3, the decrease of $\max e_j$ is at the cost of the increase of $\max q_j$. At $t = 0, \max e_j$ is maximum and $\max q_j = 0$. When the two curves intersect ($\max e_j = \max q_j$), we can obtain the minimum of $\max e_j + \max q_j$, which is the optimal solution of LRP.

4.2. PM model

By transforming LRP to Waterflow model, we find the optimization relation between the maximum of execution time and the maximum of migration time, which is the criterion of LRP optimization. Here we will introduce the other model-Particle Mechanics model. By transforming LRP to PM model, we can deduce a parallel algorithm for LRP. Therefore, the second problem above can be solved.

Before describing PM model, we introduce two important variables E_j and Q_j that are related to the execution time of processor e_j and the migration time of processor q_j, respectively.

We can obtain the execution time of every r_{ij} at time t as

$$e_{ij}(t) = \frac{r_{ij}(t)}{x_j(t)} \tag{1}$$

and the execution time of processor A_j at time t as

$$e_j(t) = \sum_{i=1}^{m} \frac{r_{ij}(t)}{x_j(t)}. \tag{2}$$

We can obtain the migration time about task T_i from processor A_l to A_j at time t as

$$q_{ilj}(t) = \triangle r_{il}(t)/b_{lj}, \tag{3}$$

the migration time about task T_i from processor A_* to A_j at time t as

$$q_{ij}(t) = \max_l q_{ilj} \quad (l = \overline{1,n}) \tag{4}$$

and the migration time on processor A_j at time t as

$$q_j(t) = \sum_{i=1}^{m} q_{ij}(t). \tag{5}$$

Because $1 - e^{-x}$ is a monotone increasing function and between 0 and 1, we choose the function to standardize the execution times and the migration time as

$$E_j(t) = 1 - \exp[-e_j(t)] = 1 - \exp\left[-\sum_{i=1}^{m} \frac{r_{ij}(t)}{x_j(t)}\right], \tag{6}$$

$$Q_j(t) = 1 - \exp[-q_j(t)] = 1 - \exp\left[-\sum_{i=1}^{m} (\max_{l=1}^{n} \frac{\triangle r_{il}(t)}{b_{lj}})\right]. \tag{7}$$

By introducing E_j and Q_j, the goal of LRP to minimize–maximize e_j and q_j transforms to maximize–minimize E_j and Q_j.

Based on Fig. 3, we can draw Fig. 4. When the two curves $\min E_j$ and $\min Q_j$ intersect ($\min E_j = \min Q_j$), we can obtain the maximum of $\min E_j + \min Q_j$, that is the minimum of $\max e_j + \max q_j$, which is the optimal solution of LRP.

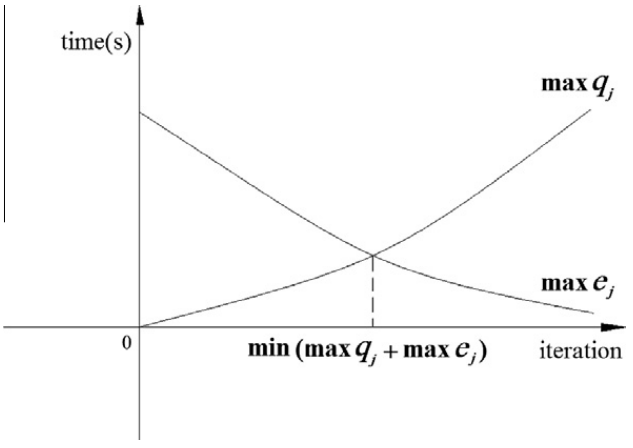

Fig. 3. The optimization relation between the maximum of execution time and the maximum of migration time.

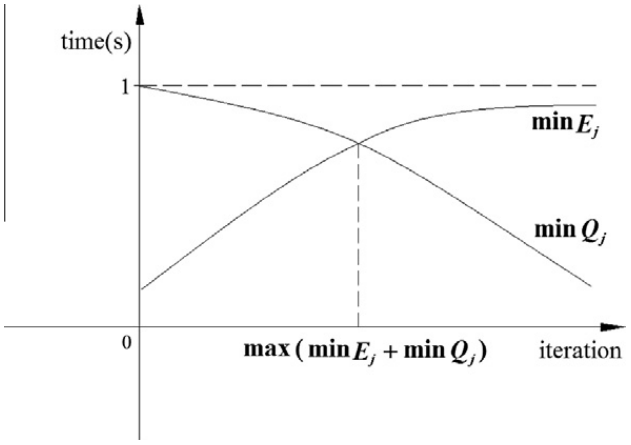

Fig. 4. When $\min E_j = \min Q_j$, we obtain the optimal solution of LRP.

From Fig. 4, it can be known that

- at the beginning of optimization process, the optimization velocity is faster;
- the optimization velocity will slow down, when the optimization value is close to the optimal solution.

Fig. 4 illustrates the physical PM model. Here we introduce two key concepts (particles and force-field) of our PM model to describe and model the optimization of LRP. We treat E_j and Q_j as two kinds of particles, which are evenly distributed at an even radian surrounded by a circumferential force-field. The values of E_j and Q_j represent the coordinates of the two kinds of particles E_j and Q_j. The coordinate of a particle is the radial distance between the origin and the particle. The force-field can make the E_j particles' minimal value increase (which will be proved in next section). E_j particles and Q_j particles move only along a radial orbit. To minimize the maximum of the individual execution times, the corresponding E_j particle in force-field will try to move as close as possible to the circumference surrounding it. At the same time, the increase of corresponding E_j will cause a decrease of corresponding Q_j based on Fig. 4 (that is, the decrease of corresponding e_j will cause an increase of corresponding q_j). Therefore, in force-field, when the corresponding E_j particle moves away from the origin, the corresponding Q_j particle moves forwards the origin. In Fig. 5, the inner loop made of E_j particles will spread out and the outer loop made of Q_j particles will shrink towards the origin. When the two loops intersect, we obtain the optimal solution of LRP.

5. The mathematical model and corresponding algorithm of W-PM

We now examine the evolutionary model that can mathematically describe the W-PM's physical models for LRP.

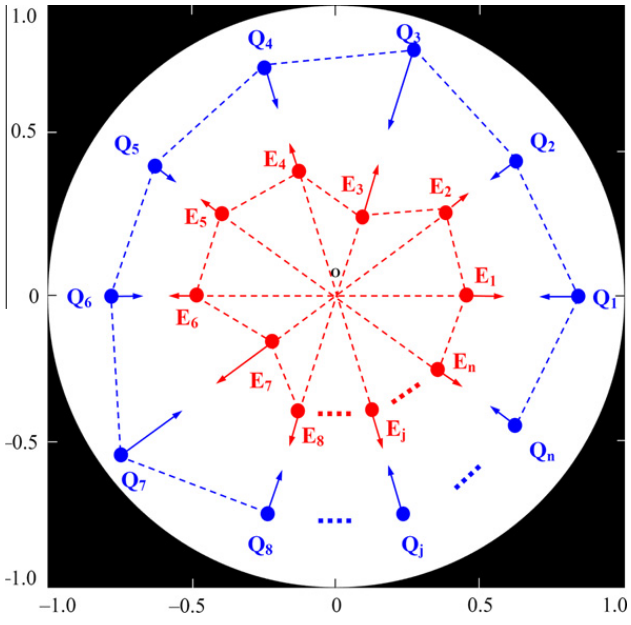

Fig. 5. The transformation of LRP to Particle Mechanics model.

We define the potential energy function of force-field as

$$P(t) = k^2 \ln \sum_{j=1}^{n} \exp\left[\frac{E_j^2(t)}{2k^2}\right]. \tag{8}$$

Theorem 1. *In Eq. (8), if k is very small, the decrease of the potential energy P(t) of force-field will cause a decrease of the processors' maximal execution times.*

Proof. Supposing that $H(t) = \max_j E_j^2(t)$, we have

$$[e^{\frac{H(t)}{2k^2}}]^{2k^2} \leqslant \left[\sum_{j=1}^{n} e^{\frac{E_j^2(t)}{2k^2}}\right]^{2k^2} \leqslant \left[n e^{\frac{H(t)}{2k^2}}\right]^{2k^2}. \tag{9}$$

Taking the logarithm for both sides of Eq. (9) leads to

$$H(t) \leqslant 2k^2 \ln\left[\sum_{j=1}^{n} e^{\frac{E_j^2(t)}{2k^2}}\right] \leqslant H(t) + 2k^2 \ln n.$$

Because n is constant and k is very small, we have

$$H(t) = \max_j E_j^2(t) \approx 2k^2 \ln\left[\sum_{j=1}^{n} e^{\frac{E_j^2(t)}{2k^2}}\right] = 2P(t).$$

The potential energy function $P(t)$ of force-field at time t turns out to represent the maximal value among $E_j(t), j = 1, \ldots, n$. So, decreasing $P(t)$ amounts to decreasing the processors' maximal execution times. □

We define the dynamic equation of subtask $r_{ij}(t)$ in W-PM model as

$$r_{ij}(t+1) = r_{ij}(t) + \triangle r_{ij}(t). \tag{10}$$

The dynamic equation is seen as the "W-PM evolution", which manipulate the update and iteration of r_{ij} until the inner loop made of E_j and the outer loop made of Q_j intersect ($E_j \geqslant Q_j$). By W-PM algorithm, r_{ij} can be computed and updated in parallel without any information exchange, which is the foundation of W-PM algorithm's parallelism.

The most important related factor that influences the update and iteration of r_{ij} is the potential energy function $P(t)$. 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. Thus we define the first item of $\triangle r_{ij}(t)$ as

$$\triangle r_{ij}(t+1) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)}, \tag{11}$$

where $0 < \lambda_1 < 1$.

Theorem 2. *The update and iteration of r_{ij} according to Eq. (11) will always cause a decrease of the processors' maximal execution times.*

Proof. Consider the effect of only the potential energy $P(t)$ on $r_{ij}(t+1)$; namely, let $\triangle r_{ij}(t+1)$ be $-\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)}$ (Eq. (11)). We determine the increment of the potential energy $P(t)$ in the unit time period as follows:

$$\triangle P(t) = \frac{\partial P(t)}{\partial r_{ij}} \frac{dr_{ij}}{dt} \approx \frac{\partial P(t)}{\partial r_{ij}} \triangle r_{ij} = -\lambda_1 \| \frac{\partial P(t)}{\partial r_{ij}} \|^2 \leqslant 0.$$

So, the update and iteration of $r_{ij}(t)$ according to Eq. (11) will make the potential energy $P(t)$ reduce, with the intension strength being λ_1. By Theorem 1, the conclusion thus is straightforward. □

The other important related factor that influences the update and iteration of r_{ij} is $E_j(t)$ because $E_j(t)$ needs to maximize. We define the second item of $\triangle r_{ij}(t+1)$ as

$$\triangle r_{ij}(t+1) = -\lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)}. \tag{12}$$

Theorem 3. *The update and iteration of r_{ij} according to Eq. (12) will always cause a decrease of the processors' execution times.*

Proof. Consider the effect of only the function $E_j(t)$ on $r_{ij}(t+1)$; namely, let $\triangle r_{ij}(t+1)$ be $-\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} - \lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)}$ (Eqs. (11), (12)). Then, the changing rate of the personal execution times of processor A_j is equal to

$$\begin{aligned}\triangle E_j(t) &= \frac{dE_j}{dt} = \frac{\partial E_j}{\partial r_{ij}}\frac{dr_{ij}}{dt} \approx \frac{\partial E_j}{\partial r_{ij}}\triangle r_{ij} \\ &= \frac{\partial E_j}{\partial r_{ij}}[-\lambda_1 \frac{\partial P(t)}{\partial r_{ij}} - \lambda_2 \frac{\partial E_j}{\partial r_{ij}}] \\ &= \frac{\partial E_j}{\partial r_{ij}}[-\lambda_1 \frac{\partial P(t)}{\partial E_j}\frac{\partial E_j}{\partial r_{ij}} - \lambda_2 \frac{\partial E_j}{\partial r_{ij}}] \\ &= -[\lambda_1 \frac{\partial P(t)}{\partial E_j} + \lambda_2]\| \frac{\partial E_j}{\partial r_{ij}}\|^2,\end{aligned}$$

where

$$\frac{\partial P(t)}{\partial E_j} = E_j \frac{\exp\left(\frac{E_j^2}{2k^2}\right)}{\sum_{j=1}^n \exp\left(\frac{E_j^2}{2k^2}\right)} \geqslant 0.$$

Thus

$$\triangle E_j(t) \leqslant 0.$$

So, the personal execution times of processor A_j will decrease. In W-PM physical model, by the theorem, we can conclude that all E_j particles will try to move as close as possible to the circumference along their own radial orbits. □

Combining Eqs. (11) and (12), we have

$$\triangle r_{ij}(t+1) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} - \lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)}, \tag{13}$$

$$\because \frac{dE_j(t)}{dr_{ij}(t)} = \frac{u_j(t)}{x_j}$$

$$\frac{dP(t)}{dr_{ij}(t)} = \frac{\partial P(t)}{\partial E_j(t)}\frac{dE_j(t)}{dr_{ij}(t)} = \frac{E_j(t)\cdot u_j(t)}{x_j}\cdot\frac{\exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^n \exp\left(\frac{E_j^2(t)}{2k^2}\right)},$$

$$\therefore \Delta r_{ij}(t) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} - \lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)} \tag{14}$$

$$= -\frac{u_j(t)}{x_j}\cdot\left[\lambda_1 \frac{E_j(t)\exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^n \exp\left(\frac{E_j^2(t)}{2k^2}\right)} + \lambda_2\right].$$

In order to satisfy the constraints of LRP, r_{ij} will be dealt with using the following two steps in parallel computing process.

- Nonnegativity: If $\min_{i,j} < 0$, then let $r_{ij} = r_{ij} - \min_{i,j} r_{ij}$.
- Normalization: Let $r_{ij} = \frac{r_{ij}}{\sum_{j=1}^n r_{ij}}$, in order to map all tasks to processors; that is, $\sum_{j=1}^n r_{ij} = 1, i = 1, \ldots, m$.

After $\triangle r_{ij}(t)$ is updated in parallel, we can compute and update $Q_j(t)$ in parallel according to Eq. (7). Meanwhile, we can compute and update $E_j(t)$ in parallel according to Eq. (6). When all $E_j \geqslant Q_j$ $(j = 1, \ldots, n)$, the optimal solution to LRP can be obtained.

Motion equations for particle E_j algorithm are defined by

$$\begin{cases} dE_j(t)/dt = \Psi_1(t) + \Psi_2(t), \\ \Psi_1(t) = -E_j(t) + \gamma_1 v_j(t), \\ \Psi_2(t) = -\left[\lambda_1 + \lambda_2 \frac{\partial P(t)}{\partial r_{ij}(t)}\right]\left\{\sum_{j=1}^m \left[\frac{\partial E_j(t)}{\partial r_{ij}(t)}\right]^2\right\}, \end{cases} \tag{15}$$

where $\gamma_1 > 1$. And $v_j(t)$ is a piecewise linear function of $v_j(t)$ defined by

$$v_j(t) = \begin{cases} 0 & \text{if } E_j(t) < 0, \\ E_j(t) & \text{if } 0 \leqslant E_j(t) \leqslant 1, \\ 1 & \text{if } E_j(t) > 1. \end{cases} \tag{16}$$

The definitions of Eqs. (15) and (16) are for the convergence proofs of W-PM algorithm (see Section 6.3).

The parallel W-PM algorithm for LRP	
0.	**Input:**
	s_i, x_j
1.	**Initialization:**
	$t \leftarrow 0$
	$\triangle t, \lambda_1, \lambda_2, r_{ij}(t)$
2.	**while** $(E_j(t) < Q_j(t))$ **do**
	$t \leftarrow t+1$
	Compute $\triangle r_{ij}(t)$ according to Eq. (13)
	$r_{ij}(t) \leftarrow r_{ij}(t-1) + \triangle r_{ij}(t)$
	If $\min_{i,j}(t) < 0$, **then**
	$r_{ij}(t) \leftarrow r_{ij}(t) - \min_{i,j} r_{ij}(t)$
	$r_{ij}(t) \leftarrow \frac{r_{ij}(t)}{\sum_{j=1}^n r_{ij}(t)}$
	Compute $E_j(t)$ according to Eq. (6)
	Compute $Q_j(t)$ according to Eq. (7)

At the end, when $E_j \geqslant Q_j (j = 1, \ldots, n), r_{ij} = r_{ij} \cdot s_i$ is the solution to LRP.

6. Convergence and parameters analysis

6.1. Convergence analysis

In this section, we construct a Lyapunov function, which is energy-related positive definite function. We can judge

the stability of the W-PM model by analyzing if the Lyapunov function monotonically decrease with the elapsing time.

Lyapunov second theorem on stability. *Consider a function $L(X)$ such that*

- $L(X) > 0$ (*positive definite*);
- $dL(X(t))/dt < 0$ (*negative definite*).

Then$L(X(t))$ is called a Lyapunov function candidate and X is asymptotically stable in the sense of Lyapunov.

It is easier to visualize this method of analysis by considering the energy of W-PM model. If W-PM model loses energy over time, then eventually W-PM model must grind to a stop and reach some final resting state. This final state is called the stable equilibrium state.

Theorem 4. *If the condition (Eq. (17)) remain valid, then W-PM model will converge to a stable equilibrium state.*

Proof. For the physical W-PM model, we define a Lyapunov function $L(r_{ij}(t))$ as

$$L(r_{ij}(t)) \triangleq \sum_{i,j} r_{ij}(t) + 2\lambda_2 \sum_{j=1}^{n} \int_0^t \frac{u_j(y)}{x_j} dy.$$

Obviously, $L(r_{ij}(t)) > 0$.

$$\because \frac{dE_j(t)}{dr_{ij}(t)} = \frac{u_j(t)}{x_j}$$

$$\frac{dP(t)}{dr_{ij}(t)} = \frac{\partial P(t)}{\partial E_j(t)} \frac{dE_j(t)}{dr_{ij}(t)} = \frac{E_j \cdot u_j}{x_j} \cdot \frac{\exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)},$$

$$\triangle r_{ij}(t) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} - \lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)}$$

$$= -\frac{u_j(t)}{x_j} [\lambda_1 \frac{E_j(t) \exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)} + \lambda_2],$$

$$\therefore \frac{dL(t)}{dt} = \frac{dr_{ij}(t)}{dt} + 2\lambda_2 \frac{u_j(t)}{x_j} = \triangle r_{ij}(t) + 2\lambda_2 \frac{u_j(t)}{x_j}$$

$$= \frac{u_j(t)}{x_j} [-\lambda_1 \frac{E_j(t) \exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)} + \lambda_2].$$

Because $\frac{u_j(t)}{x_j} > 0$, if the following Eq. (15) remain valid, then $dL(t)/dt < 0$.

$$\lambda_2 < \lambda_1 \frac{E_j(t) \exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)}. \quad (17)$$

Based on Lyapunov second theorem on stability, as long as we properly select the parameters λ_1, λ_2 according to Eq. (15), the convergence and stability can be guaranteed. That is, when $t \to \infty$, then all $r_{ij}(t) \to r_{ij}$ (constants). □

6.2. Parameters analysis

There are only three parameters in W-PM model and algorithm, k in Eq. (8) and λ_1, λ_2 in Eq. (13).

k represents the strength of the gravitational force in force-field. The larger k is, the faster the particles would move away from the origin along their radial orbits; hence, k influences the convergence speed of W-PM algorithm. As required in Theorem 1, k must be small. Usually, $0 < k < 1$.

$$\because 0 < \frac{E_j(t) \exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)} \leqslant \frac{\exp\left(\frac{E_j^2(t)}{2k^2}\right)}{\sum_{j=1}^{n} \exp\left(\frac{E_j^2(t)}{2k^2}\right)} \approx \frac{1}{n} < 1.$$

According to Eq. (15), λ_2 should be much smaller that λ_1— that is,

$$\lambda_2 < \frac{\lambda_1}{n},$$

where n is the number of processors in LRP.

6.3. Convergence proofs

Theorem 5 indicates that W-PM algorithm converges to the stable equilibrium point.

Lemma 1. *If $1 - \gamma_1 < \Psi_2(t) < 0$, then the dynamic states of particle E_j will eventually converge to a stable equilibrium states, $v_j(t) \in \{0, 1\}$.*

Proof. For $\gamma_1(t) > 1$, the $\Psi_1(t)$ of particle E_j is a piecewise linear function of the stimulus $r_{ij}(t)$, as shown by three segments: Segment I, Segment II, and Segment III in Fig. 6. By Eq. (15),$dE_j(t)/dt = 0$ holds true iff $-\Psi_2(t) = \Psi_1(t)$, which means that an intersection point between $-\Psi_2(t)$ and $\Psi_1(t)$ as the functions of $r_{ij}(t)$ in Fig. 6 is an equilibrium point. We see that, for the case of $1 - \gamma_1 < \Psi_2(t) < 0$, there are three intersection points between $-\Psi_2(t)$ and $\Psi_1(t)$, among which only the intersection points on the Segment I and Segment III, e.g. p_3 and p_4, are stable, that correspond to $v_j(t) = 0, E_j(t) < 0$ and $v_j(t) = 1, E_j(t) > 1$, respectively. □

Lemma 2. *If $\gamma_1 > 1$, $\Psi_2(t) > 0$, then the dynamic states of particle E_j will eventually converge to the stable equilibrium states, $v_j(t) = +1$.*

Proof. If $\gamma_1(t) > 1$ and $\Psi_2(t) > 0$, then the intersection points between $-\Psi_2(t)$ and $\Psi_1(t)$ of particle E_j are all located on Segment III, e.g. p_6. Therefore E_j has the stable equilibrium points with $v_j(t) = 1$, $E_j(t) > 1$. □

Lemma 3. *If $\Psi_2(t) < 1 - \gamma_1 < 0$, then the dynamic states of particle E_j will eventually converge to the stable equilibrium states, $v_j(t) = 0$.*

Proof. If $\Psi_2(t) < 1 - \gamma_1 < 0$, then $-\Psi_2(t)$ and $\Psi_1(t)$ of particle E_j only has the intersection points on Segment I, e.g.

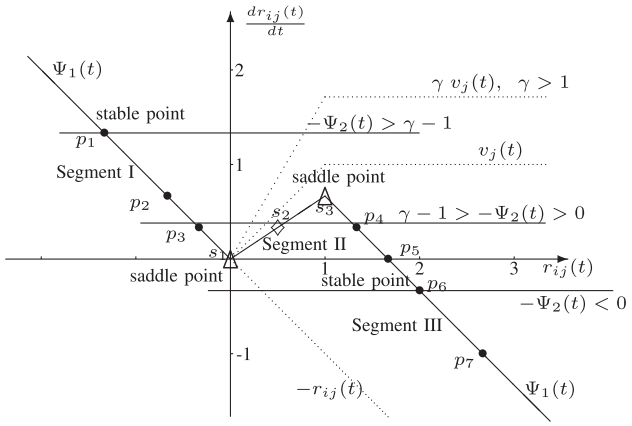

Fig. 6. When $\gamma_1 > 1$, the reachable equilibrium points of the dynamic status $v_j(t)$ of particle E_j. The point where $-\Psi_2(t)$ equals $\Psi_1(t)$ is an equilibrium point. •, △ and ♢ denote a stable equilibrium point, saddle point and unstable equilibrium point, respectively.

p_1. Therefore E_j has the stable equilibrium points with $v_j(t) = 0,\ E_j(t) < 0$. □

Theorem 5. *In the W-PM model, the dynamic Eq. (15) has the stable equilibrium points iff the right side of Eq. (15) is larger than 0 for* $E_j(t) = 1$ and $v_j(t) = 1$;.

Proof. By Eq. (6), we have $E_j(t) \geqslant 0$. Thereby, we only consider the equilibrium points with $E_j(t) \geqslant 0$. The right side of Eq. (15) is denoted by RHS. Sufficiency. Assume that, for Eq. (15), $RHS > 0$ holds for $E_j(t) = 1,\ v_j(t) = 1$. It follows that $-\Psi_2(t) \neq \Psi_1(t)$ for $E_j(t) = 1,\ v_j(t) = 1$, namely, it is impossible that the equilibrium point is the intersection point s_3 between Segment II and Segment III. Note that the saddle point s_3 isn't stable equilibrium point. Thus $RHS = \frac{dE_j(t)}{dt} > 0$ leads to the stable equilibrium points of E_j on Segment III. Necessity. Suppose that Eq. (15) has a stable equilibrium point. we need to prove that $RHS > 0$ holds for $E_j(t) = 1$ and $v_j(t) = 1$. By contrary, if there is $RHS \leqslant 0$, then the equilibrium point must be either at the point s_3 for the case of $RHS = 0$, or on Segment II for the case of $RHS < 0$. Since the point s_3 and the points on Segment II are all not stable, a contradiction happens. □

7. Simulations

We give the experimental results in this section. First, we use a simple example to show our W-PM algorithm's effectiveness, parallelism and the higher convergence speed to an optimal solution. Secondly, we show the actual times and iterations used to solve LRPs on a cluster, which can speak for the efficiency and parallelism of our W-PM algorithm. Finally, we make a general comparison between W-PM algorithm and other benchmark nature-inspired algorithms. 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.

7.1. Effectiveness

Simulations of LRP verify our W-PM algorithm's advantages in terms of the W-PM algorithm's parallelism and the higher convergence speed to an optimal solution. The simulation used these parameters: $k = 0.8, \lambda_1 = 0.9, \lambda_2 < \frac{0.9}{n}$.

Because the parallel computing $r_{ij}(t)$ is the foundation of W-PM algorithm's parallelism, the W-PM algorithm is scalable. When the number of processors (n) in LRP is very large (e.g., n is more than 10,000), the W-PM algorithm can deal with well. Here we will give the experimental results of the LRP ($m = n = 500$) in Fig. 7.

Fig. 7(a) shows the the trajectories of two kinds of particles (E_j and Q_j) at the initial state. In Fig. 7(a), most of E_j particles are far from the circumference—that is, most E_j are large, which represent most execution times e_j of processors are long. However, all Q_j particles are on the circumference, which represent all migration times q_j of processors are equal to 0.

In Fig. 7(b) and (c), by some iterations, E_j particles move away from the origin along their radial orbits, which represent that the execution times of most of processors shorten. Meanwhile, Q_j particles move forwards the origin, because the decrease of the execution times will cause to the increase of the migration times.

In Fig. 7(d), at the end of W-PM algorithm, the two kinds of particles E_j and Q_j intersect in force-field, which represent that all execution times and all migration times of processors are balanced and shortened.

Fig. 7 (a)(b)(c) (d) show four key time points on the optimization process of the LRP example. We mark the four key time points on the two optimization curves of E_j and Q_j in Fig. 8. The **minE**$_j$ curve represents the optimization process of E_j particles. The **minQ**$_j$ curve represent the optimization process of Q_j particles.

7.2. Efficiency and parallelism

The W-PM algorithm provides a valuable alternative to traditional methods because of its inherent parallelism. The subtasks, r_{ij}, can be computed and updated in parallel without any information exchange, which is the foundation of W-PMA's efficiency. The experimental results have verified the outstanding parallel capability of W-PM algorithm (see Table 3). We use 1,8,16 computing nodes of the cluster, respectively.

Table 3 includes the sequential version which comes from using one computing node of the cluster. The other parts are for the parallel version, using 4 and 16 computing nodes of the cluster. "Iterations" and "time" are the number of iterations and the time the W-PM algorithm takes to converge. As shown in Table 3, for a LRP with 510 tasks and 510 processors, the convergence time using 8 computing nodes is about 1/7 the time of the sequential version; and the convergence time using 16 computing nodes is about half the time with 8 computing nodes. The convergence time is almost inversely proportional to the number of computing nodes used by W-PM algorithm.

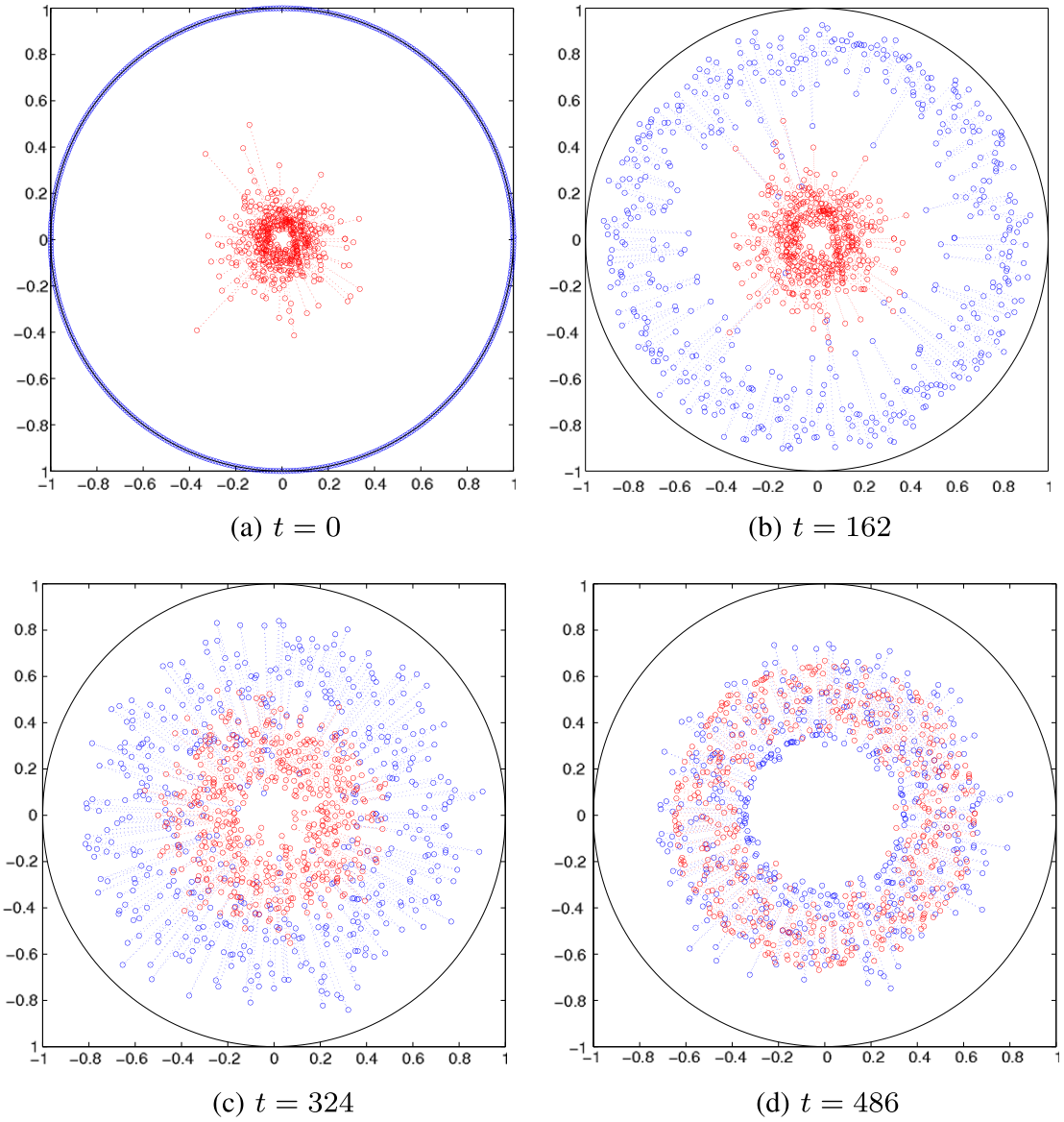

Fig. 7. The trajectories of E_j particles and Q_j particles using W-PM algorithm for LRP ($m = n = 500$), where the red circles represent the E_j particles and the blue ones represent the Q_j particles.

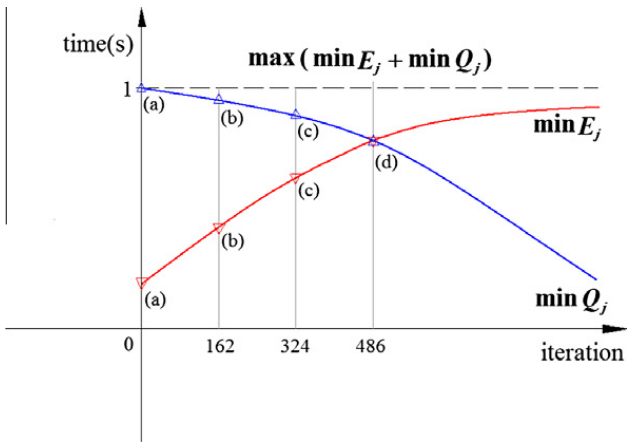

Fig. 8. The optimization process of the LRP example in Fig. 7.

7.3. Comparison between W-PM and other benchmark NAs

Popular nature-inspired approaches (NAs) include genetic algorithm (GA), simulated annealing algorithm (SA), ant colony optimization (ACO), particle swarm optimization (PSO), etc. We summarize the relative differences between our W-PM algorithm and the benchmark NAs in Table 4.

8. Conclusion

In this paper, we propose W-PM model and algorithm for LRP. The W-PM algorithm is inspired by physical models of waterflow and particle dynamics. W-PM algorithm is easy to use in spite of its seemingly abstruse theories and sinuate motivation. Without having to know the theories,

Table 3
Convergence time and speeds of W-PM algorithm with scale.

Scale		16 parallel nodes		8 parallel nodes		1 parallel node	
Processors	Tasks	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations
110	110	0.85	210	1.65	144	14.58	153
160	160	2.25	522	4.99	303	51.15	342
200	200	4.74	910	10.97	523	147.65	710
225	225	9.39	1530	22.5	846	298.83	1141
250	250	13.8	1937	36.55	1124	494.8	1593
275	275	25.58	2893	55.32	1497	908.65	2471
300	300	33.83	3617	86.12	1979	1366.93	3227
320	320	55.41	4966	113.27	2330	1838.12	3783
340	340	73.71	5918	159.99	2965	2920.07	5077
360	360	103.46	7308	214.81	3584	3718.58	5877
375	375	135.37	8536	292.91	4337	5078.08	7170
390	390	228.43	10908	388.61	5073	7070.19	8663
405	405	266.97	12095	503.36	5952	8460.17	9577
420	420	325.48	13590	608.68	6711	11763.71	11360
435	435	369.57	14939	749.19	7486	13299.61	12451
450	450	517.2	17264	867.04	8285	15746.12	13844
465	465	607.03	19063	1427.58	10331	21770.07	16051
480	480	673.87	20806	1233.41	10020	24305.55	17412
495	495	858.57	23287	1845.78	11946	32329.32	19966
510	510	1158.45	26450	1909.96	12413	33727.35	21143

Table 4
Relative differences between W-PM algorithm and other benchmark NAs.

	W-PM	GA	SA	ACO	PSO
Inspired by	Kinematics and dynamics of waterflow	Natural evolution	Thermodynamics	Behaviors of real ants	Biological swarm (e.g., swarm of bees)
Key components	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

equations and other W-PM details, W-PM can be applied or used according to our proposed algorithmic steps (at the end of Section 5).

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No.60905043, the General Research Fund of Hong Kong Research Grant Council under Grant No.7137/08E and Chinese Universities Scientific Fund.

References

[1] P. Bruckner, Scheduling Algorithms, third ed., Springer-Verlag, 2001.
[2] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, 2002.
[3] S.M. Ross, Probability Models for Computer Science, Academic Press, 2002.
[4] V. Ungureanu, B. Melamed, M. Katehakis, P.G. Bradford, Deferred assignment scheduling in cluster-based servers, Cluster Computing 9 (1) (2006).
[5] W. Winston, Optimality of the shortest line discipline, Journal of Applied Probability 14 (1977) 181C189.
[6] M. Harchol-Balter, M.E. Crovella, C.D. Murta, On choosing a task assignment policy for a distributed server system, in: Proceedings of Performance Tools 98, in: Lecture Notes in Computer Science, vol. 1468, 1998, pp. 231–242.
[7] G. Ciardo, A. Riska, E. Smirni, EquiLoad: A load balancing policy for clustered web servers, Performance Evaluation 46 (2–3) (2001) 101C124.
[8] A. Riska, W. Sun, E. Smirni, G. Ciardo, AdaptLoad: Effective balancing in clustered web servers under transient load conditions, in: 22nd International Conference on Distributed Computing Systems, ICDCS02, 2002.
[9] P. Brucker, Scheduling Algorithms, third ed., Springer, Berlin, 2001.
[10] Hans-Ulrich Heiss, Michael Schmitz, Decentralized dynamic load balancing: the particles approach, Information Sciences 84 (2) (1995) 115–128.

Xiang Feng received the M.Sc. degree in System Engineering from Naval Engineering University, Wuhan, in 2003. She received the Ph.D. degree in Control Theory and Engineering from East China University of Science and Technology, Shanghai, in 2006. She worked as a postdoctoral fellow in the Department of Computer Science of the University of Hong Kong from 2006 to 2008. She is presently an associate professor of the Department of Computer Science and Engineering, East China University of Science and Technology. Her research interests include mechanics-related nature-inspired algorithm, parallel and distributed computing and computer networks.

Francis C. M. Lau received the PhD degree in computer science from the University of Waterloo in 1986. He has been a faculty member in the Department of Computer Science of the University of Hong Kong since 1987, and served as the head of department from 2000 to 2005. He received a Golden Core recognition in 1998 and an IEEE Third Millennium Medal in 2000 for outstanding achievements and contributions to the IEEE Computer Society. His research interests include parallel and distributed computing, mobile and pervasive computing, and computer art and music.