A New Nature-inspired Algorithm for Load Balancing

Xiang Feng
Department of Computer Science
East China University of
Science and Technology
Shanghai, China 200237
Email: xfeng{@ecust.edu.cn, @cs.hku.hk}

Francis C.M. Lau
Department of Computer Science
The University of
Hong Kong, Hong Kong
Email: fcmlau@cs.hku.hk

Dianxun Shuai
Department of Computer Science
East China University of
Science and Technology
Shanghai, China 200237
Email: shdx411022@online.sh.cn

***Abstract*—The classical Load Balancing Problem (LBP) is to map tasks to processors so as to minimize the maximum load. Solving the LBP successfully would lead to better utilization of resources and better performance. The LBP has been proven to be NP-hard, thus generating the exact solutions in a tractable amount of time becomes infeasible when the problems become large. We present a new nature-inspired approximation algorithm based on the Particle Mechanics (PM) model to compute in parallel approximate efficient solutions for LBPs. Just like other Nature-inspired Algorithms (NAs) drawing from observations of physical processes that occur in nature, the PM algorithm is inspired by physical models of particle kinematics and dynamics. The PM algorithm maps the classical LBP to the movement of particles in a force field by a corresponding mathematical model in which all particles move according to certain defined rules until reaching a stable state. By anti-mapping the stable state, the solution to LBP can be obtained.**

***Index Terms*—Load balancing, approximation algorithm, nature-inspired algorithm, particle mechanics model, distributed and parallel algorithm.**

I. Introduction

An increasingly large number of scientific pursuits use computational resources as their backbone. Parallelism at the scale of tens of thousands of processors is being seen. It is imperative that more and better techniques for efficient utilization of the large scale parallel machines be developed. The main resources in a large parallel machine are its compute nodes and the interconnection network; both must be utilized efficiently. Load balancing is an age-old but effective technique for achieving better utilization of computational resources in a parallel environment.

The classical Load Balancing Problem (LBP) is to map tasks to processors so as to minimize the maximum load. Of course, there are also a variety of other problems involving task migration to balance load. The systems community has shown considerable interest in the problem of process migration in the context of scheduling tasks on multiple processors. Some of the benefits derived from process migration include better resource utilization and better overall performance.

The LBP has been proved to be NP-hard [1], which naturally means that we are unable to deliver an exact solution to the problem in a reasonable amount of time when the problem gets large. For instance, if we have 2000 tasks and 10 processors that we could use, we have 10^{2000} configurations to enumerate and compare for the LBP. While solving these problems is not feasible under such a situation, approximation algorithms may possibly be implemented to achieve a reasonable estimate within a fixed error ratio in a reasonable amount of time.

Recently, several load balancing assignment policies have been proposed [2–5]). According to two main strategies, these policies attempt to balance the load among the back-end servers:

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

In this paper, we present a nature-inspired approximation algorithm based on the Particle Mechanics (PM) model to compute in parallel approximate efficient solutions for LBPs.

II. Problem model for LBP

The Load Balancing Problem (LBP)
Given:

- A network of processors, A_j, and their processing capacities x_j (in task units per second). There is a communication channel between any two processors. A processor is characterized by the attribute: its “remaining workload”, w_j (in task units).
- A set of tasks, T_i, that are spread across these processors in a given “initial mapping”; a task is characterizied by the attribute: its “size”, s_i (in bytes).

Goal: To map the tasks to the processors so that the execution time which is the maximum of the individual execution times after the mapping is minimized.

The main notations in any LBP instance are shown in Table I.

We can formalize LBP using a matrix Λ (see Table II).

In Table II, the task mapping variable, r_{ij}, is the key of the LBP. If an algorithm can compute and update r_{ij} in parallel without any information exchange, the algorithm has a chance to solve the LBP in parallel efficiently.

TABLE II
THE REPRESENTATION MATRIX Λ OF LBP, WITH PROCESSORS (A) AND TASKS (T)

	A_1	$\cdots$	A_j	$\cdots$	A_n	s_i
T_1	r_{11}, e_{11}	$\cdots$	r_{1j}, e_{1j}	$\cdots$	r_{1n}, e_{1n}	$s_1 = \sum_{j=1}^{n} r_{1j}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
T_i	r_{i1}, e_{i1}	$\cdots$	r_{ij}, e_{ij}	$\cdots$	r_{in}, e_{in}	$s_i = \sum_{j=1}^{n} r_{ij}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
T_m	r_{m1}, e_{m1}	$\cdots$	r_{mj}, e_{mj}	$\cdots$	r_{mn}, e_{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 I
THE MAIN NOTATIONS IN LBP

Notations	Meanings
A_j	The j-th processor ($j = \overline{1, n}$)
T_i	The i-th task ($i = \overline{1, m}$)
r_{ij}	The subtask of task T_i that is mapped to processor A_j
x_j	The processing capacities of processor A_j
w_j	The remaining workload of processor A_j
s_i	The size of task T_i
e_{ij}	The execution time of r_{ij}, $e_{ij} = r_{ij}/x_j$

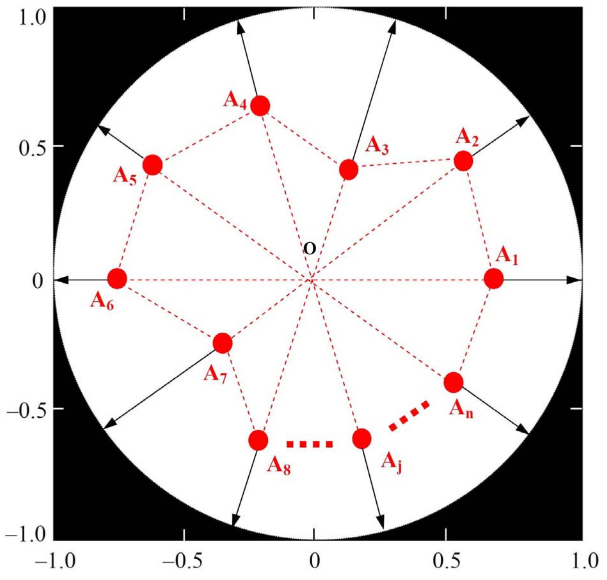

Fig. 1. The architecture of a particle mechanics model for LBP

III. THE MOTIVATION AND ARCHITECTURE OF PM MODEL

Figure 1 illustrates the physical model of the PM model. Here we introduce two key concepts (particles and force-field) of our PM model to describe and model the optimization of the LBP. We treat processors as particles, which are surrounded by a circumferential force-field. The force-field can make the particles' minimal utility increase (which will be proved in next section). The analogical relation between the PM model and the LBP can be seen in Table III.

TABLE III
THE ANALOGICAL RELATION BETWEEN PM MODEL AND LBP

In PM model	In LBP
Particles	Processors
Force-field	Max-min fairness balancing of LBP
Radial distance between the origin and a particle	The processor's personal utility
Radial distance between a particle and the circumference	The standardization of the execution times of the processor
Gravitation of force-field	Make LBP optimize
Motion of all processors	The process of LBP optimization
Forces that force-field exerts on particles	The trend and degree of processors will be optimized

In the PM model, all particles (processors) are evenly distributed at an even radian. Each particle is being exerted upon simultaneously by the force-field of the circumference surrounding the particles. The radial distance between a particle and the origin represents the corresponding particle's personal utility—that is, a processor's profit according to the current situation of the LBP. Based on the LBP, the shorter the execution times of a processor, the larger the processor's personal utility. The LBP aims to minimize the maximum of the individual execution times of all processors. The force-field will make the particle that is the closest from the origin move out along the radius—that is, to maximize the minimum of all particles' (processors') utility, the corresponding particle in force-field will try to move as close as possible to the circumference surrounding it.

IV. THE MATHEMATICAL MODEL AND CORRESPONDING ALGORITHM OF PM

We now examine the evolutionary model that can mathematically describe the PM's physical model for the LBP.

We can obtain the execution times of every r_{ij} as

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

and the execution times of processor A_j as

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

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

$$E_j(t) = 1 - \exp[-e_j(t)] = 1 - \exp\,[\,-\sum_{i=1}^{m} \frac{r_{ij}(t)}{x_j(t)}]; \tag{3}$$

where E_j represents the radial distance from particle A_j to the circumference in force-field. The radius of the force-field will always be 1 and E_j will be in $[0, 1]$.

We define the personal utility function for particle (processor) $A_j \in \{A_1, \cdots, A_n\}$ as

$$u_j(t) = 1 - E_j(t) = \exp\,[\,-\sum_{i=1}^{m} \frac{r_{ij}(t)}{x_j(t)}]; \tag{4}$$

where u_j represents the radial distance from the origin to particle A_j in the force-field. The longer the execution times of a processor, the smaller the processor's personal utility, and the corresponding particle will be closer to the origin in the force-field.

We define the potential energy function of the force-field as

$$P(t) = k^2 \ln \sum_{j=1}^{n} \exp\,[\,\frac{E_j^2(t)}{2k^2}]; \tag{5}$$

Theorem 1: In equation 5, if k is very small, the decrease of the potential energy $P(t)$ of the 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

$$[e^{\frac{H(t)}{2k^2}}]^{2k^2} \le [\sum_{j=1}^{n} e^{\frac{E_j^2(t)}{2k^2}}]^{2k^2} \le [ne^{\frac{H(t)}{2k^2}}]^{2k^2} \tag{6}$$

Taking the logarithm for both sides of Eq. 6 leads to

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

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

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

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

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

$$r_{ij}(t+1) = r_{ij}(t) + \triangle r_{ij}(t) \tag{7}$$

The dynamic equation is seen as the "PM evolution", which manipulates the update and iteration of r_{ij} until an equilibrium is reached. By the PM algorithm, r_{ij} can be computed and updated in parallel without any information exchange, which is the foundation of the 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

$$\triangle r_{ij}(t+1) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} \tag{8}$$

where $0 < \lambda_1 < 1$.

Theorem 2: The update and iteration of r_{ij} according to Eq. 8 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. 8).

We determine the increment of the potential energy $P(t)$ in the unit time period as follows:

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

So, the update and iteration of $r_{ij}(t)$ according to Eq. 8 will reduce the potential energy $P(t)$, 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 the utility function $u_j(t)$. Because $u_j(t)$ needs to maximize, we use its opposite function $E_j(t)$ to define the second item of $\triangle r_{ij}(t+1)$ as

$$\triangle r_{ij}(t+1) = -\lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)} \tag{9}$$

Theorem 3: The update and iteration of r_{ij} according to Eq. 9 will always cause a decrease of the processors' execution times.

Proof. Consider the effect of only the execution times 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. 8, 9).

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

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

Thus

$$\triangle E_j(t) \leq 0$$

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

Combining Eq. 8 and Eq. 9, we have

$$\triangle r_{ij}(t+1) = -\lambda_1 \frac{\partial P(t)}{\partial r_{ij}(t)} - \lambda_2 \frac{\partial E_j(t)}{\partial r_{ij}(t)} \tag{10}$$

In order to satisfy the constraints of the LBP, r_{ij} will be dealt with using the following two steps in the 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, $\sum_{j=1}^{n} r_{ij} = 1, i = 1, \cdots, m$.

We can obtain the radial velocity of particle A_j along its radial orbit to the circumference of the force-field by the equation

$$v_j = \frac{dE_j}{dt} = \sum_{i=1}^{m} \frac{\partial E_j}{\partial r_{ij}} \frac{dr_{ij}}{dt} = \frac{u_j}{x_j} \sum_{i=1}^{m} \frac{dr_{ij}}{dt} \approx \frac{u_j}{x_j} \sum_{i=1}^{m} \triangle r_{ij} \tag{11}$$

When all $v_j (j = 1, \cdots, n)$ are equal to 0, the PM model reaches a stable state, and the solution to LBP is obtained.

The parallel PM algorithm for LBP

```
0.  Input:
        $s_i, x_j$
1.  Initialization:
        $t \leftarrow 0$
        $\triangle t, \lambda_1, \lambda_2, r_{ij}(t)$
2.  while ($v_j(t) \neq 0$) do
        $t \leftarrow t+1$
        Compute $E_j(t)$ according to equation 3
        Compute $\triangle r_{ij}(t)$ according to equation 10
        $r_{ij}(t) \leftarrow r_{ij}(t-1) + \triangle r_{ij}(t)$
        Compute $v_j(t)$ according to equation 11
        If $\min_{i,j}(t) < 0$, then $r_{ij}(t) \leftarrow r_{ij}(t) - \min_{i,j} r_{ij}(t)$
        $r_{ij}(t) \leftarrow \frac{r_{ij}(t)}{\sum_{j=1}^{n} r_{ij}(t)}$
```

At the end, when an equilibrium is reached, $r_{ij} = r_{ij} \cdot s_i$ is the solution to LBP.

V. Convergence and parameters analysis

A. Convergence analysis

In this section, we construct a Lyapunov function, which is an energy-related positive definite function. We can judge the stability of the PM model by analyzing if the Lyapunov function monotonically decreases 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 the PM model. If the PM model loses energy over time, then eventually the model must grind to a halt and reach some final resting state. This final state is called the stable equilibrium state.

Theorem 4: If the condition (Eq. 12) remains valid, then the PM model will converge to a stable equilibrium state.

Proof. For the physical 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, $L(r_{ij}(t)) > 0$.

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

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

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

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

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

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

Because $\frac{u_j(t)}{x_j} > 0$, if the following Eq. 12 remains valid, then $dL(t)/dt < 0$.

$$\lambda_2 < \lambda_1 \frac{E_j(t) \exp(\frac{E_j^2(t)}{2k^2})}{\sum_{j=1}^{n} \exp(\frac{E_j^2(t)}{2k^2})} \tag{12}$$

Based on the Lyapunov second theorem on stability, as long as we properly select the parameters λ_1, λ_2 according to Eq. 12, the convergence and stability can be guaranteed. That is, when $t \to \infty$, then all $r_{ij}(t) \to r_{ij}$ (constants). $\square$

B. Parameters analysis

There are only three parameters in the PM model and algorithm, k in Eq. 5 and λ_1, λ_2 in Eq. 10.

k represents the strength of the gravitational force in the 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 the PM algorithm. As required in Theorem 1, k must be small. Usually, $0 < k < 1$.

$$\because 0 < \frac{E_j(t)\exp(\frac{E_j^2(t)}{2k^2})}{\sum\limits_{j=1}^{n}\exp(\frac{E_j^2(t)}{2k^2})} \leq \frac{\exp(\frac{E_j^2(t)}{2k^2})}{\sum\limits_{j=1}^{n}\exp(\frac{E_j^2(t)}{2k^2})} \approx \frac{1}{n} < 1$$

According to Eq. 12, λ_2 should be much smaller than λ_1—that is,

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

where n is the number of processors in the LBP.

VI. Simulations

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

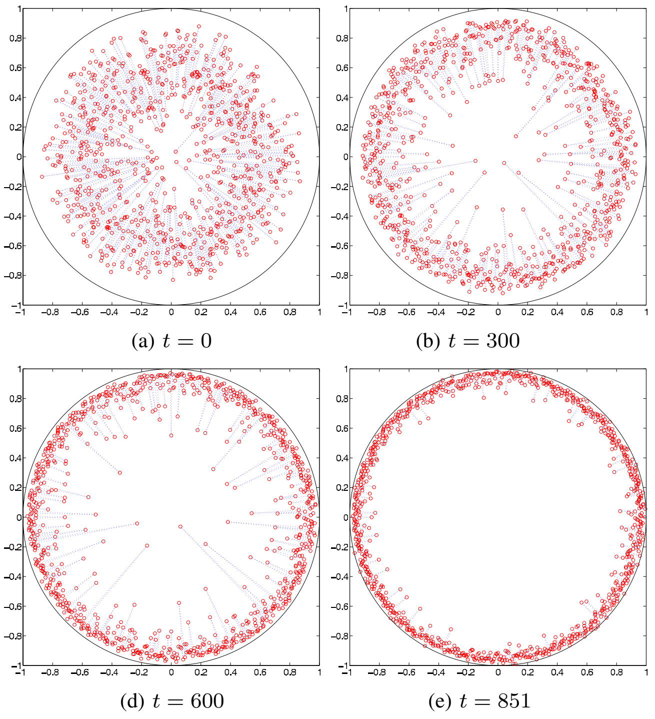

Fig. 2. The trajectories of particles using the PM algorithm for the LBP ($n = 900$)

Because the parallel computation of $r_{ij}(t)$ is the foundation of the algorithm's parallelism, the PM algorithm is scalable. When the number of processors (n) in the LBP is very large (e.g., n is more than 10000), the PM algorithm can still perform well. Here we will give the experimental results for the LBP ($n = 900$) in Figure 2.

As shown in Figure 2(a), most of particles are far from the circumference—that is, most E_j are large, which represent most execution times e_j of processors are long. In Figure 2(b) and (c), by some iterations, particles move away from the origin along their radial orbits, which represents that the execution times of most of the processors are being shortened. In Figure 2(d), at the end of the PM algorithm, particles move as close as possible to the circumference in the force-field, which represents that all execution times of processors are balanced and substantially shortened.

VII. Conclusion

In this paper, we propose the PM model and algorithm for the LBP. The PM algorithm is inspired by the physical model of particle dynamics. Our future research direction include solving the Load Rebalancing Problem (LRP), where progresses of the computation may dynamically alter the load distribution, and is therefore more difficult than the LBP. We will conceive a physical Water Flow model to model the LRP, and its corresponding mathematical model and algorithm.

Acknowledgment

This project is supported in part by the Hong Kong General Research Fund (HKU 713708E).

References

[1] Kleinberg G. and Tardos E. Algorithm Design, Cornell University, Spring 2004.
[2] P. Bruckner, Scheduling Algorithms, third ed., Springer-Verlag, 2001.
[3] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, 2002.
[4] S.M. Ross, Probability Models for Computer Science, Academic Press, 2002.
[5] V. Ungureanu, B. Melamed, M. Katehakis, P.G. Bradford, Deferred assignment scheduling in cluster-based servers, Cluster Computing 9 (1) (2006).