2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Using Estimation of Distribution Algorithm to Coordinate Decentralized Learning Automata for Meta-task Scheduling

Jie Li and Junqi Zhang[†]
Department of Computer Science and Technology
Key Laboratory of Embedded System and Service Computing, Ministry of Education
Tongji University, Shanghai, 200092, China.
[†] Corresponding author (*E-mail*: lijietjsh@gmail.com, zhangjunqi@tongji.edu.cn)

***Abstract*—Learning automaton (LA) is a reinforcement learning model that aims to determine the optimal action out of a set of actions. It is characterized by updating a selection probability vector through a sequence of repetitive feedback cycles interacting with an environment. Decentralized learning automata (DLAs) consists of many learning automata (LAs) that learn at the same time. Each LA independently selects an action based on its own selection probability vector. In order to provide an appropriate central coordination mechanism in DLAs, this paper proposes a novel decentralized coordination learning automaton (DCLA) using a new selection probability vector which is combined with the probability vectors derived from both LA and estimation of distribution algorithm (EDA). LA contributes to the own learning experience of each LA while EDA estimates the distribution of the whole swarm's promising individuals. Thus, decentralized LAs can be coordinated by EDA using the swarm's comprehensive knowledge. The proposed automaton is applied to solve the real problem of meta-task scheduling in heterogeneous computing system. Extensive experiments demonstrate a superiority of DCLA over other counterpart algorithms. The results show that the proposed DCLA provides an effective and efficient way to coordinate LAs for solving complicated problems.**

I. INTRODUCTION

Heterogeneous computing (HC) system is an emerging platform that is composed of some distributed and different high performance machines to perform different computationally intensive applications. Scheduling problem is a crucial problem that needs to be solved efficiently in HC system. There are two categories of scheduling problems that are classified according to the types of tasks: one is the scheduling of meta-tasks, the other is the scheduling of directed acyclic graph composed of communicating tasks. In this paper, we focus on meta-task scheduling problem, its goal is to find a task assignment solution that is to minimize the schedule length of several independent tasks with no data dependencies in HC system. It has been proved to be a NP-complete problem [1]. Initially some suitable heuristic algorithms are proposed to solve it, such as Min-min, Max-min [2] and Segmented Min-min [3]. Next, in order to improve the solutions' quality, meta-heuristics approaches are presented for solving it.

Besides the famous algorithms like genetic algorithm (GA) [4][5], simulated annealing (SA) [6] and particle swarm optimization (PSO) [7], learning automaton (LA) [8][9] as an important learning model has been successfully applied to solve scheduling and assignment problems in recent years. Such as static task graph scheduling [10], multiple cost optimization for task assignment [11] and multi-constraint assignment problems [12]–[16]. The goal of LA is to determine the optimal action out of a set of actions, the optimal action is defined as the one that maximizes the rewarded probability [17]. Recently, learning automata (LAs) have been shown to be valuable tools for designing multi-agent reinforcement learning algorithms and controlling the stochastic games [18]. The pioneers of all the classes of LAs belong to the fixed-structure stochastic automata (FSSA) family, they have the property that their transition and output matrices do not change with time. A subclass of FSSA have been used to solve the equipartitioning problem [19]. In order to accelerate the convergence speed, variable structure stochastic automata (VSSA) are proposed and they have been proved to be the fastest converging LAs, whose transition and output matrices are time-varying. They have been widely employed to systems that have time-varying environments, e.g., channel assignment [20], wireless and wired networks [21][22], dominating set problem [23] and tutorial-like system [24]. Normally, only one LA is used to construct the learning model of related problems. In order to supply a learning model that consists of many LAs that learn at the same time, decentralized learning automata (DLAs) [12] is proposed. In general, these LAs are VSSA. DLAs can increase the flexibility to some extent, but its performance is not ideal due to each LA in DLAs selects an action based on its own selection probability vector independently without an appropriate central coordination.

How to provide an appropriate central coordination mechanism to coordinate different LAs in DLAs becomes an important and interesting problem. Estimation of distribution algorithm (EDA) that is firstly introduced in [25] maybe very potential to solve it. Because the most important feature of EDA is to estimate the distribution of promising solutions in the search space. Thus, EDA can be used in DLAs to estimate the distribution of promising solutions that selected by different LAs. By this perspective, each LA in DLAs can obtain other LAs' learning experiences to a certain degree, and uses this knowledge to guide its own learning. In recent years, EDA has been successfully applied to a large number of discrete optimization problems by using its estimation feature. Such as permutation flowshop [26], job shop [27], hybrid flow-shop [28], nurse scheduling [29], semiconductor final test [30], resource-constrained project [31], multiobjective resource-constrained project [32], permutation-based combinatorial optimization [33], large scale global optimization [34] and max-cut problem [35].

In the field of hybrid of LA and EDA, the work [36] proposes learning automata based estimation of distribution algorithm (LAEDA) that belongs to a class of EDAs, it uses a team of learning automata to build the probability distribution model of high quality solutions. That is, LA is used to provide distribution information for EDA.

This paper proposes a novel decentralized coordination learning automaton (DCLA) that owns a new selection probability vector, which is combined with the probability vectors derived from both LA and EDA. LA contributes to the own learning experience of each LA while the EDA provides a swarm's comprehensive knowledge through estimating the distribution of promising individuals. Thus, each LA will possess more comprehensive learning and search abilities, decentralized LAs will be coordinated by the EDA using the swarm's comprehensive knowledge. The information of combined selection probabilistic vector in DCLA obtains from not only the learning experience of each LA itself, but also the promising distribution experience of the whole swarm. The proposed automaton is applied to solve the real problem of meta-task scheduling in HC system. Extensive experiments demonstrate a superiority of DCLA over other counterpart algorithms. The results show that the proposed DCLA provides an effective and efficient way to coordinate LAs for solving complicated problems.

The rest of this paper is organized as follows: meta-task scheduling problem formulation is described in Section II. In Section III, traditional LA and EDA are introduced. The proposed DCLA is presented in Section IV. In Section V, DCLA is applied to meta-task scheduling problem. Extensive simulation results that indicate the superiority of DCLA over other algorithms are presented in Section VI. Finally, concluding remarks are given in section VII.

II. Meta-task scheduling problem formulation

In this paper, we focus on meta-task scheduling problem (MTSP) that is composed of a set of tasks and computing machines. Meta-task is defined as these tasks that are independent of each other, that is they have no inter-task data dependencies. Next, a formulation of meta-task scheduling problem is presented. Firstly, HC system is assumed to be composed of M heterogeneous machines, which is defined as $M = \{M_1, M_2 \ldots, M_m\}$, and a set of tasks $T = \{T_1, T_2 \ldots, T_n\}$ is submitted to HC system at a specific time, m is the number of computing machines and n is the number of tasks. Secondly, an expected time to compute (ETC) matrix is used to estimate the expected execution times of a task when it runs on different machines. ETC matrix is a $n \times m$ matrix, every element in ETC matrix denotes the expected execution time of a task on a particular machine. That is, E_{qp} ($1 \leq q \leq n,\ 1 \leq p \leq m$) is the expected execution time of a task q on machine p. q is an arbitrary task and p is an arbitrary machine. In addition, $C[p]$ is the sum of the expected execution times of tasks that assigned to the p-th machine. Formally, $C[p]$ is defined as follows:

$$C[p] = \sum_{Z(q)=p} E_{qp}, (1 \leq q \leq n, 1 \leq p \leq m) \qquad (1)$$

here, Z is a set of size n, $Z(q)$ indicates task q is assigned on different machines.

Then, the makespan of the meta-task scheduling problem is the maximal total execution time among those machines. And the goal of scheduler is to minimize makespan in this paper. Formula is defined as follows:

$$makespan = \max\{C[p]\}, (1 \leq p \leq m) \qquad (2)$$

III. Related works

A. Learning Automaton (LA) and Linear Reward-Penalty (L_{RP}) Algorithm

LA is a reinforcement learning model [8][9]. The goal of such an automaton is to determine the optimal action out of a set of actions, the optimal action is defined as the one that maximizes the rewarded probability [17]. LA selects the optimal action through interacting with an environment, which provides an appropriate response that is defined as reward or penalty, then this response is used to update the selection probability vector of actions by LA. The simple steps of LA are described in **Algorithm 1**:

Algorithm 1 LA
```
1: begin
2:    do
3: LA selects an action according to the selection probability
   vector;
4: The selected action is supplied to an environment, which
   provides an appropriate response that is reward or penalty;
5: LA updates the selection probability vector of actions
   according to this response.
6:    until The end condition is satisfied.
7: end
```

In order to understand the learning process of LA better, we will simply introduce linear reward-penalty (L_{RP}) algorithm that is adopted in this paper. L_{RP} is a typical LA algorithm and perhaps it is the earliest scheme considered in mathematical psychology [37]. In L_{RP} algorithm, the selection probability vector of actions are computed strictly dependent on the environmental response, which is described as β, $\beta \in \{0, 1\}$, "1" is reward and "0" is penalty. If the k-th action of i-th LA is attempted at time t, The updating formulas of the selection probability vector are as follows:

If $\beta = 1$

$$LA\_p_i^j(t) = (1-a)LA\_p_i^j(t-1), j \neq k;$$
$$LA\_p_i^k(t) = LA\_p_i^k(t-1) + a[1 - LA\_p_i^k(t-1)].$$

Else

$$LA\_p_i^j(t) = \frac{b}{r-1} + (1-b)LA\_p_i^j(t-1), j \neq k;$$
$$LA\_p_i^k(t) = (1-b)LA\_p_i^k(t-1).$$

here, a and b are reward and penalty parameters, respectively. $0 < a < 1, 0 \leq b < 1$. $LA\_p_i = (LA\_p_i^1, LA\_p_i^2, \ldots, LA\_p_i^r)$ is the selection probability vector of the i-th LA, $\sum_{R=1}^{r} LA\_p_i^R = 1$. r is the number of actions. The probability $LA\_p_i^k(t)$ is increased at time t if the response is reward, other actions' probabilities

$LA\_p_i^j(t)$, ($j \neq k$ and $1 \leq j,k \leq r$) are decreased by an amount proportional to their values at time $t-1$. Conversely, the probability $LA\_p_i^k(t)$ is decreased at time t if the response is penalty, other actions' probabilities $LA\_p_i^j(t)$ are increased by an amount proportional to their values at time $t-1$.

B. Estimation of Distribution Algorithm (EDA)

EDA [25] is a new class of evolutionary computation. The main difference between EDA and traditional evolutionary computations is that the interrelations in EDA are expressed explicitly through a joint probability distribution model that constructed through some selected promising individuals from the previous iteration. Then new offspring are generated by sampling the constructed probability distribution model. The most difficult work in EDA is the estimation of the joint probability distribution model associated with selected promising individuals. The following **Algorithm 2** is a description for EDA:

```
Algorithm 2 EDA
Require:
   Input: The initial population generated randomly.
1: begin
2:    do
3: Select promising individuals from population according to their fitness values by using a certain selection procedure;
4: Estimate the probability distribution of the selected promising individuals;
5: Sample new offspring according to the estimated probability distribution;
6: Use new sampled offspring to replace some worse individuals in the previous population.
7:    until The end condition is satisfied.
8: end
```

IV. Proposed DCLA

One of the most serious defects that weakens the performance of decentralized learning automata (DLAs) is that each LA independently selects an action based on its own selection probability vector without an appropriate central coordination. In order to solve this problem, this paper proposes a novel decentralized coordination learning automaton (DCLA) using a new selection probability vector to combine two probability vectors that derived from both LA and EDA. That is, the information in combined probabilistic vector obtains not only the learning experience of each LA itself, but also the promising distribution experience of the whole swarm. The decentralized LAs can be coordinated by EDA using the swarm's comprehensive knowledge.

The advantage of incorporating EDA into DLAs is that the promising distribution experience of the whole swarm is able to reflect the distribution of promising solutions in the search space. EDA can build a distribution model through making full use of the information of swarm's promising learning experience, then this model is used to coordinate the decentralized LAs.

In DCLA, a new selection probability vector updating formula is designed by combining the learning experience of LA and the whole swarm's promising distribution experience that is obtained by EDA. Here, we assume that each component in a solution corresponds to a LA, and a variable in a component corresponds to an action. That is, the r variables in each component correspond to r actions in each LA, denoted as an action set $a_i = \{a_i^1, a_i^2, \ldots, a_i^r\}$ in the i-th LA. The combined selection probability is as follows:

$$p_i^j(t) = \alpha \times LA\_p_i^j(t) + (1-\alpha) \times EDA\_\widetilde{p}_i^j(t) \quad (3)$$

where $p_i = (p_i^1, p_i^2, \ldots, p_i^r)$ denotes the combined selection probability vector, $p_i^j(t)$ is the combined selection probability of the j-th action in the i-th LA at time t, $1 \leq j \leq r$. $p_i^j(t)$ is calculated by $LA\_p_i^j(t)$ and $EDA\_\widetilde{p}_i^j(t)$. $LA\_p_i^j(t)$ denotes the learning experience of the j-th action in the i-th LA. $EDA\_\widetilde{p}_i^j(t)$ shows the promising distribution experience of the j-th variable in the i-th component. $\alpha \in [0,1]$ is a learning parameter and it is used to balance the contributions between the individual learning experience of each LA and the promising distribution experience of the whole swarm derived from EDA. That is, the bigger α is, the greater contribution of LA, the smaller α is, the greater contribution of EDA. Thus, the setting of learning rate α has a direct impact on the tradeoff the learning ability between LA and EDA. For example, if α is 0, new offspring will be sampled based on EDA. As α increases, the amount of LAs' learning abilities increase. Next, we will simply analyze the learning process of DCLA. All LAs will initially expect to be explorative until their rewards and penalties have balanced out, and as the search goes on, the probabilistic model constructed by EDA to converge, EDA can be thought of as an exploitation behavior as sampled solutions focus round LAs' "learned" values with high probability.

A. Learning Process of LA_p_i

At each time, a new candidate solution is generated by all LAs using their combined selection probability vectors, then environment gives a response according to the quality of the new candidate solution. In this paper, the response from environment is described as β, $\beta \in \{0,1\}$,"1" is reward and "0" is penalty. Then each LA will utilize the response to update its own selection probability vector. **Algorithm 3** displays the learning process of LA_p_i.

```
Algorithm 3 The learning process of $LA\_p_i$
1: begin
2: Each LA selects an action (a variable in a component) according to the combined selection probability vector $p$ in equation (3). A new candidate solution is generated when all LAs have selected their actions;
3: If the new candidate solution is better than the global best solution, environment will give a reward response, else the response is penalty;
4: $L_{RP}$ algorithm is used to update the selection probability vector $LA\_p_i$ in the $i$-th LA according to this response.
5: end
```

B. Distribution Experience of $EDA\_\widetilde{p}_i$

Population-based incremental learning (PBIL) [38] is used to model the promising distribution experience of the whole swarm $EDA\_\widetilde{p}_i$. In PBIL, there is a probability vector $EDA\_\widetilde{p}_i = (EDA\_\widetilde{p}_i^1, EDA\_\widetilde{p}_i^2, \ldots, EDA\_\widetilde{p}_i^r)$, which is to characterize the distribution of the promising solutions in the search space, i is a component of solution and r is the number of all potential variables on this component. The updating formulae of the probability vector $EDA\_\widetilde{p}_i$ at time t are as follows:

$$EDA\_\widetilde{p}_i^j(t) = (1-\lambda) \times EDA\_\widetilde{p}_i^j(t-1) + \lambda \times EDA\_\overline{p}_i^j(t) \quad (4)$$

$$EDA\_\overline{p}_i^j(t) = \frac{\sum_{k=1}^{Q} ps_{ik}^j}{Q} \quad (5)$$

where $EDA\_\widetilde{p}_i^j$ is the percentage of the value of the j-th potential variable on the i-th component, and it is learned and updated from the promising solutions' historical experience and current experience that calculated by $EDA\_\overline{p}_i^j$. That is, $EDA\_\widetilde{p}_i$ can be regarded as a probability vector for modeling the distribution of the promising solutions' historical and current experience. $\lambda \in [0, 1]$ is a learning parameter and it is used to balance the contributions between historical experience and current experience. The process of building $EDA\_\widetilde{p}_i$ can be described as **Algorithm 4**:

Algorithm 4 The distribution experience of $EDA\_\widetilde{p}_i$

```
1: begin
2: Calculate the fitness value of each individual solution in
   the swarm. The fitness value is makespan;
3: Select $Q$ promising solutions (ps) from the swarm to
   calculate equation (5), then the distribution experience
   of good regions in the search space is calculated using
   equation (4).
4: end
```

V. DCLA for MTSP

In this section, the proposed DCLA is applied to the MTSP. Firstly, solution representation for this problem and initial population are given. Next, the learning process of LA will be described. At last, the complete process of DCLA for MTSP is given.

A. Solution Representation and Initial Population Generation

Solution representation is an important factor that can affect the performance of DCLA, so a well designed solution representation can better solve related problems. In this paper, a potential solution is an integer set $T[q], q \in \{1, 2, \ldots, n\}$ with n elements, and the q-th element indicates that a specific machine p is assigned to the q-th task, i.e., $T[q] = p, p \in \{1, 2, \ldots, m\}$. n is the number of tasks, m is the number of machines.

At initial time, the population is randomly generated by uniform distribution, and the size of population is PN. A generated solution in population is described as $x_{k,i} = randint(1, m)$, which is denoted the i-th component of the k-th solution, $k \in \{1, 2, \ldots, PN\}$. $randint$ is a function that generates an integer uniformly distributed in the range $[1, m]$. Then, the makespan of a solution is used as its fitness value.

B. Selection Probability Updating Strategy of LA

Here, each LA corresponds to a task and its actions are m machines. After all LAs select their actions, a new candidate solution is generated. If the new candidate solution is better than the global best solution, environment will give a reward and all LAs will update their selection probability vectors according to L_{RP}. However, if the response is penalty, a new selection probability updating strategy is designed to better use of the characteristics of the problem itself. Firstly, the heaviest load machine in the candidate solution is selected; secondly, those tasks (LAs) that are located in the heaviest load machine will decrease their selection probabilities on this machine, so that they will select a more appropriate machine in the next iteration. Fig. 1 shows the effect of tasks that are located in the heaviest load machine before and after penalty, respectively.

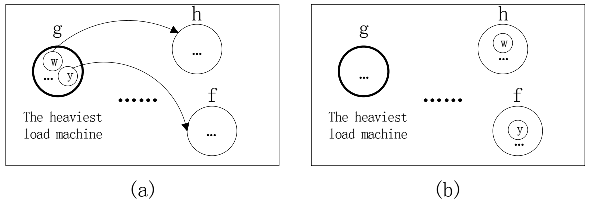

Fig. 1. The effect of tasks that are located in the heaviest load machine before and after penalty, respectively. (a). Before penalty; (b). After penalty.

In Fig. 1(a), the heaviest load machine is g, tasks w and y are located in this machine. After tasks w and y update their selection probability vectors according to the penalty response, task w is likely to migrate to another machine h, task y may migrate to another machine f, as displayed in Fig. 1(b). The updating strategy can be elaborated in **Algorithm 5**.

Algorithm 5 The selection probability updating strategy

```
1: begin
2: All LAs will use equation (3) to select their actions and
   generate a new candidate solution;
3: When the new candidate solution is better than the global
   best solution, environment will give a reward response "1",
   otherwise the response is penalty "0";
4: If response is reward "1", all LAs will increase their
   selection probability vectors of the selected candidate
   solution;
5: Else response is penalty "0", those tasks (LAs) that are
   located in the heaviest load machine will decrease their
   selection probabilities on this machine just like Fig. 1.
6: end
```

C. Complete Procedure of DCLA for MTSP

At each iteration, the individual learning experience of each LA and the promising distribution experience of the whole swarm are combined to generate new offspring. The details of the proposed DCLA for MTSP are presented in **Algorithm 6**. The flowchart of DCLA for MTSP is given in Fig. 2.

VI. Experimental Evaluation

Extensive simulations are carried out in order to compare DCLA algorithm with a genetic algorithm (GA) [4], a traditional particle swarm optimization (PSO) algorithm [7] and a

Algorithm 6 DCLA for MTSP

```
1:  begin
2:     do
3:  At iteration $t \geq 1$, calculate the fitness value for each potential solution;
4:  Select two solutions from the current swarm randomly;
5:  Compare the fitness values of two selected solutions through tournament selection procedure and the better one is selected. Then repeat step 4 and 5 until Q promising solutions are selected;
6:  PBIL is adopted to estimate the distribution of promising regions in the search space based on equations (4) and (5);
7:  The individual learning experience of each LA and the promising distribution experience of the whole swarm are combined to build a selection probability vector equation (3);
8:  A new offspring is generated by the combined selection probability vector;
9:  If the fitness value of the new offspring is better than the global best solution, environment will response a reward, else is a penalty;
10: $L_{RP}$ algorithm is used to update LAs' selection probability vectors according to the response and related strategy. Here, Q new offspring are generated by the combined selection probability vector;
11: Q current worse individuals in population will be replaced by the Q new offspring;
12:    until The end condition is satisfied.
13: end
```

simulated annealing (SA) algorithm [6]. All these algorithms are coded in MATLAB-R2010a and dedicated simulations are executed on a 3.20GHz Core i3 processor with 4GB main memory running under Windows 7 environment.

The benchmark of instances is based on ETC matrix that decides various possible heterogeneous computing environments. In this paper, these instances are classified into 12 different types according to three metrics in ETC matrices. That is task heterogeneity, machine heterogeneity and consistency. Task heterogeneity is defined as the amount of variance among the execution times of tasks for a given machine, machine heterogeneity is denoted as the variation that is possible among the execution times for a given task across all the machines, consistency represents whenever a machine executes any tasks faster than other machines. Besides, in order to better exert the performance of LA, the number of actions in each LA is as small as possible. So instances consist of 50 tasks (LAs) and 5 machines (actions) are considered in this paper. Other details about benchmark please refer to [4][6][7].

In our experiments, the number of population for proposed DCLA is set to 100. Besides, since some earlier studies have shown that the best GA solution always come from the population that had been seeded with the Min-min solution [4], the same method of initiating population is adopted in DCLA and PSO. Also, the result of Min-min is used as the initial solution of SA. These algorithms are terminated when they have experienced a maximum number of iterations (this number is set to 5000). All algorithms are stochastic approaches and each independent run of the same algorithm on

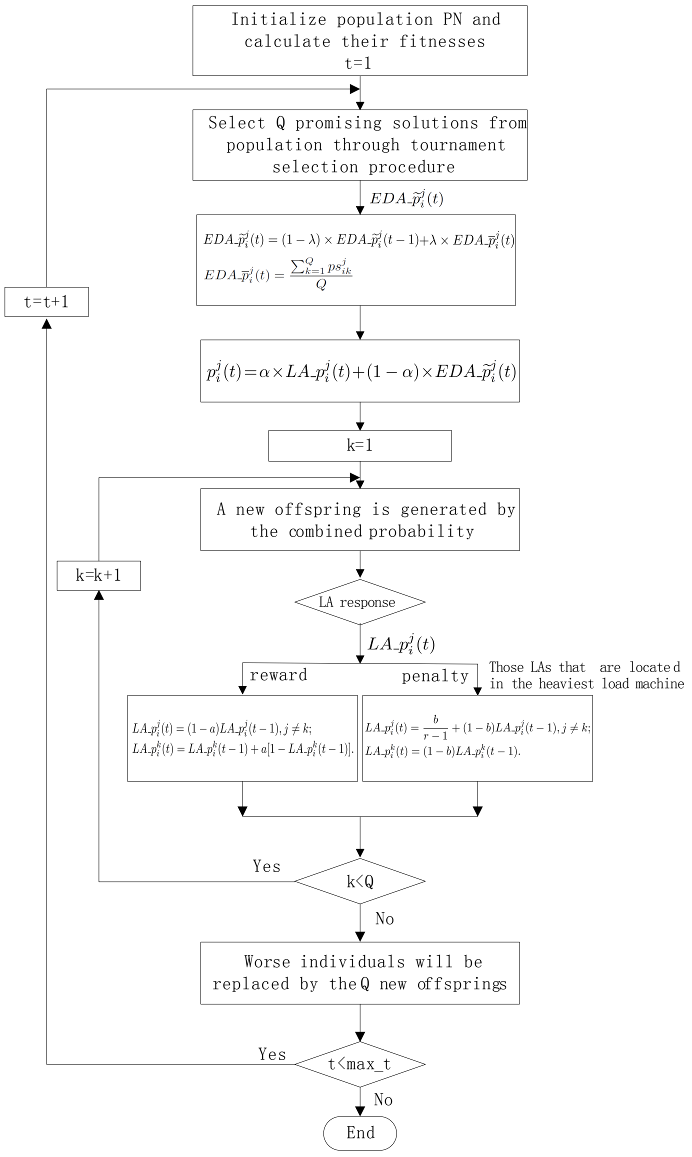
Initialize population PN and calculate their fitnesses t=1
Select Q promising solutions from population through tournament selection procedure
$EDA\_\widetilde{p}_i^j(t)$
$EDA\_\widetilde{p}_i^j(t) = (1-\lambda) \times EDA\_\widetilde{p}_i^j(t-1) + \lambda \times EDA\_\overline{p}_i^j(t)$
$EDA\_\overline{p}_i^j(t) = \frac{\sum_{k=1}^{Q} ps_{ik}^j}{Q}$
$p_i^j(t) = \alpha \times LA\_p_i^j(t) + (1-\alpha) \times EDA\_\widetilde{p}_i^j(t)$
t=t+1
k=1
A new offspring is generated by the combined probability
k=k+1
LA response
$LA\_p_i^j(t)$
reward
penalty
Those LAs that are located in the heaviest load machine
$LA\_p_i^j(t) = (1-a)LA\_p_i^j(t-1), j \neq k;$
$LA\_p_i^k(t) = LA\_p_i^k(t-1) + a[1 - LA\_p_i^k(t-1)].$
$LA\_p_i^j(t) = \frac{b}{r-1} + (1-b)LA\_p_i^j(t-1), j \neq k;$
$LA\_p_i^k(t) = (1-b)LA\_p_i^k(t-1).$
Yes
k<Q
No
Worse individuals will be replaced by the Q new offsprings
Yes
t<max_t
No
End

Fig. 2. Flowchart of DCLA for MTSP. PN: population number, Q: number of new offspring, max_t: the max time (iteration).

a particular problem instance may yield a different result. Thus, we run each algorithm 20 times for every problem instance and report the statistical results.

The experimental results are shown in Table I where the first column indicates the instance name. M_{avg}, V_{avg}, M_{best} and RPD, denote, respectively, the average makespan, the average variance, the best makespan and the average relative percentage deviation obtained by the corresponding algorithms to solve the problem instances. Here, RPD is calculated as follows:

$$RPD = (M_i - M_{DCLA})/M_{DCLA} \times 100 \qquad (6)$$

where M_{DCLA} is the average makespan across 20 independent runs obtained by the proposed DCLA algorithm and M_i is the average result provided by each of the three comparator algorithms for each instance. Fig. 3. shows the evolution of the mean makespan derived from the four algorithms.

TABLE I. Experimental results of SA, PSO, GA and DCLA over 20 independent runs on 12 instances in 5000 iterations. M_{avg}, V_{avg}, M_{best} and RPD indicate the average makespan, the average variance, the best makespan and the average relative percentage deviation respectively. "+" and "-" denote the performance of the corresponding algorithm is worse than, better than DCLA.

Instances	SA				PSO				GA				DCLA		
	M_{avg}	V_{avg}	M_{best}	RPD	M_{avg}	V_{avg}	M_{best}	RPD	M_{avg}	V_{avg}	M_{best}	RPD	M_{avg}	V_{avg}	M_{best}
1	3891	269.48	3442	13.67+	3685	95.77	3558	7.65+	3483	57.87	3362	1.75+	**3423**	27.29	3362
2	8971	320.60	8434	13.42+	8437	118.93	8186	6.67+	8201	65.29	8084	3.69+	**7909**	57.23	7790
3	729	0	729	3.99+	712	3.74	705	1.56+	705	3.10	699	0.57+	**701**	2.58	698
4	7488	67.67	7290	8.13+	7097	76.56	7019	2.48+	6959	56.42	6878	0.49+	**6925**	26.65	6891
5	4315	182.64	3984	19.03+	3904	255.62	3662	7.69+	3884	37.49	3803	7.14+	**3625**	93.70	3540
6	7899	174.12	7460	5.44+	7782	131.50	7568	3.88+	7702	158.48	7425	2.81+	**7491**	110.44	7313
7	3172	0	3172	3.49+	3172	0	3172	3.49+	3072	14.29	3049	0.22+	**3065**	13.44	3041
8	9560	56.12	9322	5.06+	9112	138.93	8969	0.14+	**9050**	62.56	8956	0.53-	9099	79.97	8889
9	3556	476.33	3191	11.33+	4061	369.33	3444	27.14+	3337	93.01	3191	4.47+	**3194**	18.35	3135
10	5842	22.16	5777	7.31+	5668	23.87	5640	4.11+	5595	24.05	5571	2.77+	**5444**	46.47	5386
11	4421	9.61	4380	4.71+	4330	54.87	4248	2.55+	4250	20.94	4200	0.66+	**4222**	26.88	4187
12	8113	127.86	7743	10.27+	7646	67.52	7563	3.92+	7492	59.50	7395	1.83+	**7357**	21.52	7326

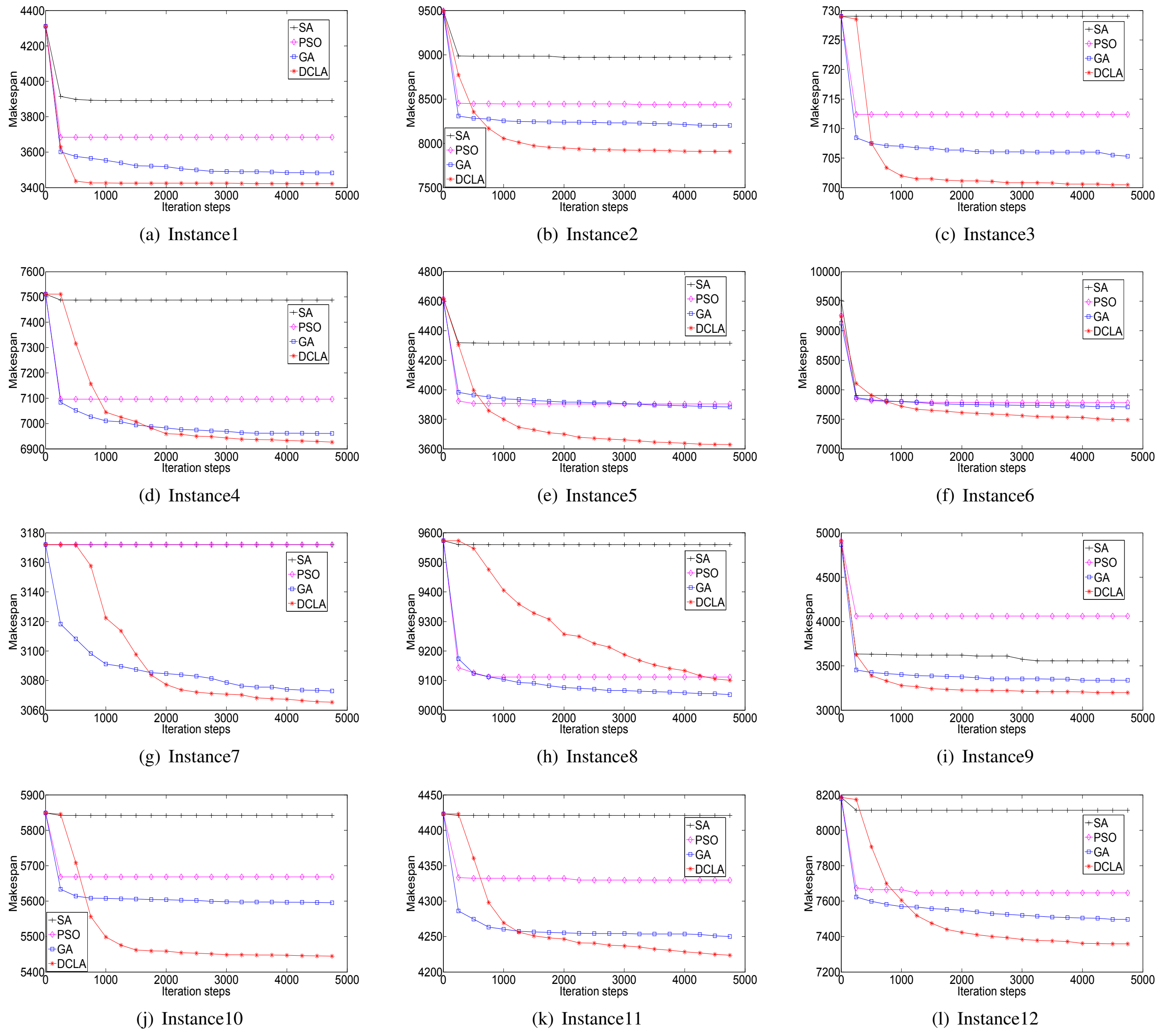

Fig. 3. The evolution of the mean makespan derived from DCLA and other algorithms over 20 independent runs on 12 instances.

The setting of parameter values of all algorithms are described as follows:

- SA:
  1) The initial solution is generated by Min-min;

2) Cooling factor = 0.9;
3) Increasing factor = 1.05.

- PSO:
  1) The initial population is generated randomly and a Min-min solution is seeded;
  2) Population size = 100;
  3) c1 = c2 =2.05, w=0.9.
- GA:
  1) The initial population is generated randomly and a Min-min solution is seeded;
  2) Population size = 100;
  3) Crossover probability = 0.9, mutation probability = 0.2.
- DCLA:
  1) The initial population is generated randomly and a Min-min solution is seeded;
  2) Population size = 100;
  3) λ=0.1, α=0.1, a=b=0.1.

It is observed from Table I that the DCLA algorithm achieves the best results in terms of the best makespan and the average performance among all the instances comparing with other algorithms. On average, the improvements of DCLA over SA, PSO and GA are 13.67%, 7.65% and 1.75% in instance 1, 13.42%, 6.67% and 3.69% in instance 2, 3.99%, 1.56% and 0.57% in instance 3, 8.13%, 2.48% and 0.49% in instance 4, 19.03%, 7.69% and 7.14% in instance 5, 5.44%, 3.88% and 2.81% in instance 6, 3.49%, 3.49% and 0.22% in instance 7, 11.33%, 27.14% and 4.47% in instance 9, 7.31%, 4.11% and 2.77% in instance 10, 4.71%, 2.55% and 0.66% in instance 11, 10.27%, 3.92% and 1.83% in instance 12, respectively. In instance 8, the performance of DCLA is better than SA and PSO. These results indicate that the proposed DCLA is better at solving meta-task scheduling problem. From Fig. 3, we can see clearly that the convergence of DCLA is faster than SA, PSO and GA significantly and the accuracy is conserved. Specially, DCLA outperforms PSO that utilizes the global best position ($gbest$) of the swarm. Normally, PSO is easy to be trapped into a local optimum when $gbest$ remains unchanged for a long time. DCLA can avoid such premature convergence problem to a certain extent because the diversity of swarm in DCLA is maintained. This is mainly due to new offspring are sampled through probabilities. In addition, there seems to be an interesting behavior of DCLA taking longer to learn in the early stages of runs, even though it mostly gets better solutions if the waiting time long enough. This is maybe due to all LAs will take a long time to converge their "learned" values and is particularly marked in instances 7 and 8. The good performance of the DCLA can be attributed in large part to the incorporation of effective EDA, which is used to build the promising distribution experience of the whole swarm, then this promising distribution experience information is used to coordinate the learning of each LA. Therefore, the learning experience of DCLA obtains from not only LA itself, but also the whole swarm's comprehensive knowledge.

VII. Conclusion

In this paper, a novel decentralized coordination learning automaton (DCLA) is proposed. It owns a new selection probability vector, which is combined with two probability vectors that derived from both LA and EDA. The decentralized LAs can be coordinated by the EDA using the swarm's comprehensive knowledge. The results on meta-task scheduling problem in heterogeneous computing system show that the proposed DCLA provides an effective and efficient way to coordinate LAs for solving complicated problems. In the future, we will upgrade the performance of DCLA in more discrete optimization problems. Besides, we will try to apply DCLA to solve some continuous function problems.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC), under Grants No. 61272271 and 61332008. This program is also supported by the National Basic Research Program of China (973 Program) under Grant No. 2014CB340404, Natural Science Foundation Program of Shanghai under Grant No.12ZR1434000 and Fundamental Research Funds for the Central Universities under Grants No. 0800219201.

References

[1] D. Fernańdez-Baca,"Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427-1436, 1989.

[2] M. Maheswarana, S. Alib, H. J. Siegelb, D. Hensgenc and F. Freundd,"Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *8th IEEE Heterogeneous Computing Workshop (HCW 99), San Juan, Puerto Rico*, pp. 30-44, 1999.

[3] M. Y. Wu and W. Shu,"Segmented Min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems," *Proceeding of the 9th Heterogeneous Computing Workshop, Cancun Mexico*, pp. 375-385, 2000.

[4] T. D. Braun, H. J. Siegel, N. Beck, L. LBölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R. F. Freundk,"A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing system," *Journal of Parallel and Distributed Computing*, pp. 810-837, 2001.

[5] R. Subrata, A. Y. Zomaya and B. Landfeldt,"Artificial life techniques for load balancing in computational grids," *Journal of Computer and System Sciences*, pp. 1176-1190, 2007.

[6] A. A. P. Kazem, A. M. Rahmani and H. H. Aghdam,"A modified simulated annealing algorithm for static task scheduling in grid computing," *International Conference on Computer Science and Information Technology*, pp. 623-627, 2008.

[7] A. Salman, I. Ahmad and S. Al-Madani,"Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, pp. 363-371, 2002.

[8] K. S. Narendra and M. A. L. Thathachar,"Learning automata: An introduction," *Englewood Cliffs, NJ: Prentice-Hall*, 1989.

[9] M. S. Obaidat, G. I. Papadimitriou and A. S. Pomportsis,"Learning automata: Theory, paradigms and applications," *IEEE Transactions on Systems, Man and Cybernetics-partB:Cybernetics*, vol. 32, pp. 706-709, 2002.

[10] Y. Masoudi, S. Lotfi, D. Karimzadgan, F. Fathy and K. Abdi,"Static task graph scheduling in real time homogenous multiprocessor systems using learning automata," *2011 International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 423-429, 2011.

[11] R. D. Venkataramana and N. Ranganathan,"Multiple cost optimization for task assignment in heterogeneous computing systems using learning automata," *Proceeding of Heterogeneous Computing Workshop, 1999. (HCW '99)*, pp. 137-145, 1999.

[12] G. Horn and B. J. Oommen,"Solving multiconstraint assignment problems using learning automata," *IEEE Transactions on Systems, Man and Cybernetics-partB:Cybernetics*, vol. 40, pp. 6-18, 2010.

[13] G. Horn and B. J. Oommen,"A fixed-structure learning automaton solution to the stochastic static mapping problem," *Proceedings 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[14] G. Horn and B. J. Oommen,"Generalised pursuit learning automata for non-stationary environments applied to the stochastic static mapping problem," *Proceedings 11th International Conference on Information Systems Analysis and Synthesis (ISAS) and The 2nd International Conference on Cybernetics and Information Technologies, Systems Application (CITSA)*, vol. 1, pp. 91-97, 2005.

[15] G. Horn and B. J. Oommen,"Towards a learning automata solution to the multi-constraint partitioning problem," *Proceedings of the 2006 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1-8, 2006.

[16] G. Horn and B. J. Oommen,"An application of a game of discrete generalised pursuit automata to solve a multi-constraint partitioning problem," *Proceedings IEEE International Conference SMC, Taipei, Taiwan*, vol. 2, pp. 1042-1049, 2006.

[17] A. S. Poznyak and K. Najim,"Learning automata and stochastic optimization," *Berlin, Germany: Springer-Verlag*, 1997.

[18] B. Masoumi and M. R. Meybodi,"Speeding up learning automata based multi agent systems using the concepts of stigmergy and entropy," *Expert Systems with Applications*, vol. 38, pp. 8105-8118, 2011.

[19] B. J. Oommen and D. C. Y. Ma,"Deterministic learning automata solutions to the equipartitioning problem," *IEEE Transactions on Computers*, vol. 37, pp. 2-14, 1988.

[20] H. Beigy and M. R. Meybodi,"Cellular learning automata with multiple learning automata in each cell and its applications," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 40, pp. 54-65, 2010.

[21] V. L. Kakali, P. G. Sarigiannidis, G. I. Papadimitriou and A. S. Pomportsis,"A novel adaptive framework for wireless push systems based on distributed learning automata," *Wireless Pers Communication*, pp. 591-606, 2011.

[22] S. Misra, B. J. Oommen, S. Yanamandra and M. S. Obaidat,"Random early detection for congestion avoidance in wired networks: A discretized pursuitlearning-automata-like solution," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 40, pp. 66-76, 2010.

[23] J. A. Torkestani and M. R. Meybodi,"Finding minimum weight connected dominating set in stochastic graph based on learning automata," *Information Sciences*, pp. 57-77, 2012.

[24] B. J. Oommen and M. K. Hashem,"Modeling a students behavior in a tutorial-like system using learning automata," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 40, no. 2, pp. 481-492, 2010.

[25] M. H. PaaG,"From recombination of genes to the estimation of distribution," *Parallel problem solving from nature, PPSN, 1996, IV. Lecture notes in computer science*, pp. 178-787, 1996.

[26] H. C. Liu, L. Gao and Q. K. Pan,"A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Systems with Applications*, pp. 4348-4360, 2011.

[27] X. J. He, J. C. Zeng, S. D. Xue and L. F. Wang,"An efficient estimation of distribution algorithm for job shop scheduling problem," *Proceedings of First International Conference on Swarm, Evolutionary, and Memetic Computing, SEMCCO 2010, Chennai, India*, pp. 656-663, 2010.

[28] S. Y. Wang, L. Wang, M. Liu and Y. Xu,"An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines," *Journal of Advanced Manufacturing Technology*, pp. 2043-2056, 2013.

[29] U. Aickelin and J. P. Li,"An estimation of distribution algorithm for nurse scheduling," *Journal of Annals of Operations Research*, pp. 289-309, 2007.

[30] X. C. Hao, J. Z. Wu, C. F. Chien and M. Gen,"The cooperative estimation of distribution algorithm: a novel approach for semiconductor final test scheduling problems," *Journal of Intelligent Manufacturing*, pp. 1-13, 2013.

[31] L. Wang and C. Fang,"A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem," *Expert Systems with Applications*, pp. 2451-2460, 2012.

[32] L. Wang, C. Fang, C. D. Mu and M. Liu,"A pareto-archived estimation-of-distribution algorithm for multiobjective resource-constrained project scheduling problem," *IEEE Transactions on Engineering Management*, pp. 617-626, 2013.

[33] J. Ceberio, E. Irurozki, A. Mendiburu and J. A. Lozano,"A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems," *Progress in Artificial Intelligence*, pp. 103-117, 2012.

[34] Y. Wang and B. Li,"A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization," *Nature-Inspired Algorithms for Optimisation*, pp. 171-198, 2009.

[35] S. D. Sousa, Y. Haxhimusa and W. G. Kropatsch,"Estimation of distribution algorithm for the max-cut problem," *Graph-Based Representations in Pattern Recognition*, pp. 244-253, 2013.

[36] R. Rastegar and M. R. Meybodi,"A new estimation of distribution algorithm based on learning automata," *2005 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1982-1987, 2005.

[37] S. Lakshmivarahan,"Learning algorithms theory and applications," *New York: Springer-Verlag*, 1981.

[38] S. Baluja,"Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," *CMU-CS-94-163, Carnegie Mellon University*, 1994.