Inf Syst Front (2017) 19:1057–1066
DOI 10.1007/s10796-017-9743-5

The resource allocation model for multi-process instances based on particle swarm optimization

Weidong Zhao[1] · Qingfeng Zeng[2] · Guangjian Zheng[1] · Liu Yang[3]

Published online: 18 March 2017

Abstract Resource allocation in process management focuses on how to maximize process performance via proper resource allocation since the quality of resource allocation determines process outcome. In order to improve resource allocation, this paper proposes a resource allocation method, which is based on the improved hybrid particle swarm optimization (PSO) in the multi-process instance environment. Meanwhile, a new resource allocation model is put forward, which can optimize the resource allocation problem reasonably. Furthermore, some improvements are made to streamline the effectiveness of the method, so as to enhance resource scheduling results. In the end, experiments are conducted to demonstrate the effectiveness of the proposed method.

Keywords Particle swarm optimization · Genetic algorithm · Resource allocation model · Multi-process · Update mechanism

✉ Qingfeng Zeng
qfzeng@shufe.edu.cn

Weidong Zhao
wdzhao@fudan.edu.cn

Guangjian Zheng
gk_cheng@126.com

Liu Yang
liuyang_2011@yeah.net

[1] Shanghai Key Laboratory of Data Science, School of Software, Fudan University, 220 Handan Rd, Shanghai 200433, China

[2] Department of Information Management and Engineering, Shanghai University of Finance & Economics, 777 Guoding Rd, Shanghai 200433, China

[3] School of Software, Fudan University, 220 Handan Rd, Shanghai 200433, China

1 Introduction

Workflow comes from the computer-supported business process automation. It enables the coordination between each process link, and ultimately successfully achieves the overall business objectives. From the 90's, more and more scholars began to pay attention to and research the field of business process, and also established the related subject fields of process management and process mining. The quality of the whole process can not only determine the performance of enterprises, and even directly affect the competitiveness of enterprises. How to improve the process quality and thus enhance the competitiveness of enterprises is a long-term problem faced by the enterprises (Zhao 2014).

In the workflow system, resources are defined as necessary entities for the execution of activities. The resources can be entities such as software systems, printers, fax machines, or personnel. In the runtime, actors usually have access to some resources, and before the implementation of activities, they must get all the resources needed. If two activities simultaneously require the same resource exclusively (the resource can only execute one activity at one time), then both of the activities compete for the resource, causing a conflict. However, if the resource is randomly assigned to one of the activities and the other is delayed due to the lack of available resources, it may violate the time constraint and affect the overall workflow performance.

The purpose of process resource allocation is to allocate resources to the most appropriate activity (Cheng et al. 2013; Jin et al. 2011; Huang et al. 2012). Business process resource allocation method can be divided into single instance resource allocation and multi instance resource allocation according to the applicable environment. At present, the resource optimization research, which is based on process mining under a single instance environment is mainly rely on the mining resource allocation rules to optimize resource models. Such as Huang and others set the participants, the start time of the activity, the

completion time and other activities related data as input, set the logical sequence of activities as the constraint conditions, and mined association rules of resource allocation from process logs (Huang et al. 2011). Ly and others considered the resource allocation rule as a classification problem, set the activity participants and activity types as input, set whether the participants were involved in activities as a result of classification, then used the decision tree algorithm to find activity assignment rules from the historical data and the organization structure of the running process, to improve the resource redistribution of the process (Ly et al. 2006). On this basis, Senkul and Toroslu (2005) and Yang et al. (2008) proposed a more sophisticated resource allocation model, which is to select the most appropriate resources by mining the matching relationship between the resources and activities in the process log.

Due to the fact that a large number of process instances exist at the same time is very common in daily operation, the resource allocation method of multi-process instances is more close to reality, but this problem has not gained much focus in previous research. When analyzing the process resource allocation model, choosing multi-process instances over single-process instances can show the work situation of the resource set from the overall perspective. But it also brings a lot of difficulties in the management of resource allocation as well as showing flexibility. In the environment of multi-process instances, the resource scheduling has become the core issue in workflow management. It is very important to allocate related resources effectively in the multi-process instances so that each process instance can obtain the appropriate resources at the appropriate time (Smanchat et al. 2011). Multi instance process systems are more complex, which are composed of multi-process tasks and various kinds of resources, and so on (Petkov et al. 2005).

Obviously, multiple process instances running at the same time will lead to a problem: when a resource is used, other activities which need this resource have to be in waiting state. In addition to the activities in parallel execution of the same process may occur in resource conflict, the resource conflicts between different activities in concurrent process of multiple instances also can't be ignored (Delias et al. 2010). At present, there are few studies which discussed concurrent operation of multiple process instances. From the perspective of messaging, these researches are divided into two types of methods: synchronous and asynchronous message, respectively using activate - stop multi instances process and running nested processes in the exclusive method. These methods can reduce the access conflicts of the process instance to the resources. From the perspective of process performance, resource allocation can be regarded as a multi-objective optimization problem, minimum time or minimum cost is the common optimization objectives. But the efficiency of these methods needs to be further enhanced. To solve the many to many problems mentioned above of multi-process instances. Proclet theory arises at the historic moment. Each Proclet is a process and can be represented by activating the message between the Proclet into multi acting examples (Van Der Aalst et al. 2001). This method can't achieve single activity synchronization under multiple instances, because it may need to run an instance first, then to activate or copy a single or multiple acting instances to run repeatedly or simultaneously.

Many enterprises have complex business and organization structures, the number of process instances increases correspondingly. They may have the situation of seasonal business or stage surge of the task volume. However, in order to control the process cost of the enterprise, the existing resource allocation method is difficult to meet the peak demand on the number of tasks, and the resource allocation problem in the multi-process instance environment is still not well solved. Currently the heuristic algorithm has very good effect when solving this kind of problem, in which the particle swarm optimization (PSO) has obvious advantage in solving this kind of multi-process instance scheduling assignment problem. It has fast convergence speed, high efficiency of dealing with multi-targets and multi-process instances. And it can easily computed in a distributed way, overcome early puberty, obtain the optimal solution under dynamic environments (Salman and Ahmad 2002; Blackwell 2007). But the ordinary PSO may be easy to fall into the optimal solution and so on, needs to be improved on the basis of the algorithm. Therefore, this paper focuses on efficient resources allocation problem in multi-process instance concurrency, introduces PSO to achieve multi-objective optimization of global search strategies, improves the algorithm on the basis of PSO and combination of partial mapping crossover algorithm, and finally solves the resource allocation problems effectively in case of constrained business process resources.

2 Multi-process instance resource allocation model and particle swarm optimization

Multi-process instance resource allocation method can be seen as a multi-objective optimization problem. It may have multiple competing objectives in the multi-process instance environment, which is different from the resource allocation optimization of single process instance. The optimization results obtained on this basis are a set of feasible solutions. Multi objective optimization problems are usually accompanied with high dimensional search space. If traditional optimization algorithms are used to solve the resource allocation problem under the multi-process instances environment, it will lead to high time complexity. Therefore, it is necessary to optimize the algorithms.

2.1 Multi-process instance resource allocation model

The multi-process instance resource allocation model is more complex than single process instance resource allocation model, so the former should be considered overall, and then

refined to the various aspects to make specific analysis. The multi-process instance resource allocation model needs to meet three basic constraints:

(1) The sequence of each process activity is non preemptive constrain. Process activities can be triggered only after executing all the precursor activities. In the same process instance, the resource can be allocated only after executing all the precursor activities, otherwise errors will occur.
(2) Multiple process instances should avoid resource conflict constraints in the parallel operation. The resource capacities are limited, For example, a resource is implementing an activity of certain process instance and has no idle time to receive new tasks from other concurrent process instances, or resources are too overloaded to complete the tasks of multiple process instances in a specified time, which will lead to unavailable resources. The availability of resources at various time points must be guaranteed in parallel multiple process instances; otherwise it will cause resource conflicts.
(3) The overall cost of multiple processes must be limited to a certain extent. Time and cost are the two most important indicators to measure the performance of business processes. Except execution time of activities, there exists waiting period between activities. The cost will often change while controlling the completion time. The pursuit of shortening process time is likely to result in increased cost. In order to avoid such situations, it is needed to control the cost no more than certain set of constraints in the process of minimizing the process time.
(4) The resource allocation follows the priority rules of the activity. Every activity has its own priority, so resource allocation should also consider dynamic changes in the priority of each activity. Affected by the activity's attributes, the priority value of each activity is also determined by waiting and delay time. Generally, the longer waiting time of an activity is, the greater its priority is.

The multi instance resource allocation model is as follows:

$$\text{Sequence}(v) > \text{Sequence}(v.\text{precurser}), v \in SA \quad (1)$$

$$\begin{aligned} &\text{Resourceconflict}(tm, s) = \text{ture}, s \in SP \\ &\textstyle\sum_p \sum_v \text{cost}(s, v, v.\text{precurser}) < \text{cost}_{lim}, v \in SA, s \in SP, p \in P \end{aligned} \quad (2)$$

$$D(P) = min \textstyle\sum_p \sum_v (\text{Time}(v) + \text{Wtime}(v.\text{precurser})) \quad (3)$$

$$k(v) = (Dn * \text{Wtime}(v.\text{precurser}))/(Vn - Vfn), v \in SA \quad (4)$$

In Formula (1) and (2), *S*A represents the set of process activities; *S*P represents the set of process resources; *P* represents the process set. *Sequ*ence (*v*) represents the scheduling order number of the activity *v*, *v. precurser* is the precursor activity of *v*. *Resourceconflict(tm, s)* represents the availability of the resource *s* in the time *tm cost(s , v , v . precurser)*, is the cost of *s* executing *v* and the waiting cost before executing *v*. $cost_{lim}$ represents the given process cost constraint. In Formula (3), *D*(*P*) represents the performance of *P*. The model performance metric is processing time, of which *Time (v)* represents the consumed time *v* taken in business processes. *v. precurser* is the precursor activity of *v*. *Wtime*(*v.precurser*) represents the interval between *v* and its precursor activity. Process performance is the final measure of using the following algorithm to allocate multi-process resources, which the optimal solution obtained by the final algorithm should minimize the total process time. *k*(*v*) represents the priority of *v*, *Dn* represents the total number of delays before execution, *Vn* represents the total number of activities in the process, and *Vfn* means the number of activities which has been completed in the process.

2.2 Particle swarm optimization

Particle swarm optimization (PSO) is a simplified model based on swarm intelligence, and it is a kind of bionic algorithm which imitates the flock of birds flying to a habitat in a multidimensional space. The PSO model is similar to the genetic algorithm, which controls the search behavior by updating the individuals (particles) in the swarm. Each particle represents a candidate solution to the problem. Similar to the chromosome in the genetic algorithm, a particle is considered as a point in the multi-dimensional space, and the state of the particle is described by its position and velocity (Eberhart and Shi 1998; Zhang and Gong 2013).

Compared with the traditional multi-objective optimization methods and other evolutionary optimization algorithms, the arithmetic coding of PSO is relatively simple and has a faster convergence rate. PSO is parallel in nature, and the resource allocation of multi-process instances is often in the distributed and concurrent application scenarios, so it is feasible to use the PSO to optimize resource allocation of multi-process instances.

For PSO has better performance in multi objective optimization, activity scheduling, combinatorial optimization and other NP-complete problems, properly setting the evaluation function of the particle can solve the problem of the diversity of processor velocity (Wang et al. 2011; Gao and Li 2010; Liu et al. 2015). In this paper, the improved particle swarm optimization algorithm is used to generate the resource allocation strategy in consideration of the resource allocation conflict of multiple process instances and the characteristics of variable work load. On the basis of general PSO, the improved algorithm redefines the representation and updating methods of particles in the process of resource allocation, which makes the improved PSO converge faster and more efficient. Faster convergence to the optimal solution in the finite iteration is

more suitable for the resource allocation model of multi-process instance resources in the business process.

3 Resource scheduling based on PSO

In this paper, the PSO crossover operation is introduced to realize the resource scheduling in the multi-process instance environment. The detailed description of the specific improved optimization algorithm of PSO is given as the following.

(1) First, initialize a particle swarm, all the particles in the particle swarm are randomly generated, according to the best position (local optimal) experienced by the particle and the best position (global optimal) all the particles found so far, along the trajectory to regulate the following direction and position of the particles. For the M-dimensional vector, the position of the j-th iteration of the i-th particle can be expressed as $P_i(j)=\{p_{i1}(j), p_{i2}(j),..., p_{iM}(j)\}$. Similarly, the velocity of the j-th iteration of the i-th particle in the M-dimensional vector can be expressed as $VE_i(j)=\{ve_{i1}(j), ve_{i2}(j), \ldots, ve_{iM}(j)\}$. The updating mechanism of particles during flight can be denoted by Formula (5) and (6):

$$VE_i(j) = w(j)\ VE_i(j-1) + c_1 r_1 \left(P_i^L - P_i(j-1)\right) + c_2 r_2 \left(P^G - P_i(j-1)\right) \quad (5)$$

$$P_i(j) = VE_i(j) + P_i(j-1) \quad (6)$$

where $i \in \{1,2, \ldots, n\}$, n is the total number of particle swarm, called the population size; $j \in \{1,2,...,J\}$, J is the limit of the iteration, as well as the maximum value; $P_i^L = \{p_{i1}^L, p_{i2}^L, \ldots, p_{iM}^L\}$ represents the best local location obtained by the particle i iterates after j-1 times. $P^G = \{p_1^G, p_2^G, \ldots, p_M^G\}$ represents the global best position of all the particles so far. c_1 and c_2 are normal values, represent learning factors; r_1 and r_2 are random numbers between 1 and 0; w(j) is the inertia weight, which is used to show the influence degree of the previous velocity on the current velocity.

Formula (3) calculated the new velocity of particles according to the previous velocity,current position, best global position and distance between the local optimal positions. $w(j)\ VE_i(j-1)$ stands for the extent of maintaining the original velocity to continue to explore the solution space, the inertia weight $w(j)$ is the extent of following the original velocity; $c_1 r_1 (P_i^L - P_i(j-1))$ stands for the degree of how close the optimal solution is to the local optimal solution while updating velocity, and the random variables are introduced to prevent the particles from falling into the local optimal solution; $c_2 r_2 (P^G - P_i(j-1))$ represents the particle shares information beyond self-bound with other particles in the particle swarm, namely close to the global optimal solution in the whole environment. Lack of this item means particles lack communication, then the probability of obtaining the optimal solution will be reduced. Formula (4) is used to calculate the new position of the particles, which can be obtained by multiplying new particle velocity with the unit time and summing its previous position.

3.1 Particle representation

The goal of the heuristic algorithm is to coordinate the limited resources. The resource allocation model of multi-process instances is needed during scheduling and allocation of each resource.

In the PSO algorithm, the position which each particle goes through represents a possible solution. In the set of multi-process tasks, there are several resources to be allocated from sorted business processes with the number H, then each particle G_i can be represented by the multi-dimensional vector $G_i = (Q_{1,1}, Q_{1,2}, \ldots, Q_{j,N})$,where every element $Q_{j,N}$ in the vector means that the resource $Q_{j,N}$ performs the N-th activities($j \in \{1, \ldots H\}$) in the j-th industry business process instance. In the multidimensional vector, the sort between the elements represents the sequence of resource allocation.

Assuming that there are three process instances for a total of 10 activities (process instance 1 has 3 activities, process instance 2 has 3 activities, process instance 3 has 4 activities) waiting for the resource allocation. By the improved PSO algorithm, the optimal solution particles G_i with the particle representation are as follows:

$Q_{1,1}$	$Q_{2,1}$	$Q_{1,2}$	$Q_{1,3}$	$Q_{3,1}$	$Q_{3,2}$	$Q_{3,3}$	$Q_{2,2}$	$Q_{2,3}$	$Q_{3,4}$
1	3	2	1	4	5	5	1	3	2

10 activities are allocated by 5 resources, namely $Q_{1,1}=1, Q_{2,1}=3, \ldots, Q_{3,4}=2$. For instance, $Q_{2,3}=3$ represents that in the resource allocation results represented by the optimal solution particle Gi, the activities of process instance 2 are completed by resource 3, and the sequence of all elements denotes the sequence of resource allocation, which corresponds with the relationship between the resources and the distribution of activities.

3.2 Particle update mechanism

Suppose that particle swarm is $\exists = \{G_1, G_2, \ldots, G_i, \ldots, G_n\}, (1 \le i \le n)$, where n represents the size of the particle swarm. As mentioned above, each particle Gi corresponds to a resource allocation method. And for any i and j ($i \le j$), the process performance which uses G_i as resource allocation way is less than one uses G_j as resource allocation way performance, at this time the population $\exists$ maintain orderly. The particle velocity calculated by formula (3), which essentially is used to measure the distance between the current and to-be updated position of the particle. By the formula (3) we

can also know that if the distance between one element from the array of particle's current position and one element from the array of particle's best position (including the local best position and the global best position) is too large, then the absolute value of the element corresponding to the updated velocity array will be relatively larger. Hence, the updated particles will not necessarily move toward the direction of the optimal solution. Therefore, the introduction of PMX (partially mapped crossover algorithm) in the particle update mechanism, which refers to the particle update process, the two individual particles based on the probability of cross-exchange of some elements, so as to achieve better update results. In this way we can convert the value of each element of the N-dimensional velocity to absolute one, and then calculate the ratio α of the absolute value of each element to that of the particle element at the optimal position. α indicates the possibility of selecting the particle element as the base point under the PMX operation (partial mapping crossover algorithm). After getting the probability that each element is selected as the base point of the crossover operation and the concept of PMX operation, we can carry out the cross operation of particle elements.

We can randomly select two elements in particles as the reference point of cross-operation according to the ratio of α, the two particles make cross-operation, and produce new resource allocation results. It can be seen that the larger the difference between the elements is, the greater the probability of the PMX operation is selected. The purpose of this method is to optimize the iterative updating mechanism, and the update and exchange between similar particles can effectively avoid the updated results away from particle motion, prevent the particles from evolving rapidly into local optimal solutions and help to generate the optimal resource allocation result. After the whole population is updated, the particles remove and re calculate the velocity and position to obtain the optimal solution. Each particle's position represents a possible solution. According to the solution, we can allocate specific resources to the corresponding activities (Wang Wang et al. 2015).

3.3 Resource allocation process based on PSO

The steps of resource allocation based on the PSO are as follows:

Step 1: Initialize the iteration $j = 0$.

Step 2: Randomly generate the particle swarm with n particles and get initial position $P_i(0)$and initial velocity $VE_i(0)$. The population size of the particle swarm is n, initial position $P_i(0)=\{p_{i1}(0), p_{i2}(0), \ldots, p_{iM}(0)\}, i \in \{1, \ldots X\}$ are randomly generated,initial velocity $VE_i(0) = \{ve_{i1}(0), ve_{i2}(0), \ldots, ve_{iM}(0)\}, i \in \{1, \ldots X\}$ are randomly generated as well. Noted that each element in the M-dimensional array of particle positions is a positive integer, and the range of values is in [1, M], any element value over this range will be adjusted: if $p_{ii'}(j)>M$, then $p_{ii'}(j)=M$; if $p_{ii'}(j)<1$, then $p_{ii'}(0)=1$. M-dimensional particle velocity arrays also have a limitation of $[-VE^{max}, VE^{max}]$ to ensure that particles do not deviate from the range of particle position when moving according to the updated velocity. When the value of the element in the velocity array exceeds that range, it will be adjusted: if $VE_{ix}(j) > VE^{max}$, then $VE_{ix}(j) = VE^{max}$;if $VE_{ix}(j) < -VE^{max}$, then $VE_{ix}(0) = -VE^{max}$.

Step 3: Set the initial position of the particle as the initial local optimal P_i^L, analyse the position of all the particles to get the initial global optimal PG.

Step 4: Enter the next round of iteration $j = j + 1$.

Step 5: Update the particle velocity$VE_{ix}(j)$. Using the formula (3) with the consideration of the inertia of the original velocity and the distance between the current position and the local optimal position (the global optimal position), calculate the updated velocity.

Step 6: Step 6: update the position of the particle $P_{ix}(0)$. The particle has introduced the partial mapping crossover algorithm. The particle update mechanism is as follows:

① Use formula (3) to update the velocity.

② Then calculate the probability that each dimension may be selected to cross substitute by the velocity.

③ According to this probability, randomly select two suitable elements to substitute each other, then form particles information on a new resource allocation method.

④ According to the updated velocity and the position information after substitution, use formula (4) to update the position of the particle information and complete the update and iteration of the particles.

Step 7: To evaluate the position coordinate array of the particle, obtain the new local optimal P_i^L and global optimal P^G. We assess the fitness of the original and updated particles. The fitness is the total time of multiple process instances after the resource allocation. Different particles represent one kind of resource allocation, and unreasonable resource allocation will cause that the activities cannot be suspended in a timely manner, resulting in different total process time. The shorter the cost of total process time is, the higher adaptation of the process is. Every time the particles in the swarm updates, the particle fitness will be evaluated. If the particle fitness is greater than that of the locally optimal P_i^L (global optimal P^G), then the location information will be replaced by local optimal (global optimal) location information.

Step 8: Determine whether to stop iteration. If you meet one of the following situations, the PSO particle update iteration process will be terminated:

① When the global optimal P^G remains unchanged after N times iterations, the iteration is stopped and N is the maximum number of iterations. If no updated global optimal solution P^G is generated after N times iterations, it can be explained that the PSO algorithm has found the global optimal solution.
② Achieve the maximum number of particle iteration times. If there is no such particle update iteration above, then loop to step 4 and continue next round of particle swarm iteration.

Step 9: Set the global optimization P^G as the best solution to resource allocation. When the iteration stops, remark the number of iterations as D^{max}. The global optimal P^G is the global optimal solution, and the activity resource allocation in the position vector is the optimal activity resource allocation.

The calculation of this algorithm mainly consists of the updated particle velocity, position of the basic algorithm, and the fitness value and the improved PMX operation. As mentioned above, given the conditions that the population size of the particle swarm is n, the number of iterations is D^{max}, the size of the problem is S,initial position $P_i(0)$ and initial velocity $VE_i(0)$, we can get the time complexity of the basic algorithm is Dm*$ax * O(n * 3S)$; and the time complexity of PMX is $D^{max} * O(n * S)$, so the total time complexity of the algorithm is $D^{max} * O(n^*4S) \approx D^{max} * O(n * S)$.

4 Experiments

We obtained a total of 2262 customer compensation data from a Shanghai airplane company, and tested the data as input. The data contained the number of customers for compensation, application time, compensation process operators, tickets ID for compensation etc. Analysis was carried out on the compensation process model of the airline company. In Fig. 1, the process consisted of 7 activities, "X" indicates the choice relationship, "+" indicates the parallel relationship. The process is composed of several activities in parallel, but the total number of resources is fixed, so in every steps of the process the resources were mutually exclusive. For example, there are two customers who apply for ticket check at the same time, ticket check staff will check one of the tickets a time, and the other will be delayed. An appropriate resource allocation algorithm can provide the system with the solution of less cost, less consuming time.

After customer compensation data is input in the compensation system and the process runs for some time, it can generate a lot of event logs. The logs contain the compensation results, operation time point of each activity, process operators, compensation time and so on. We can compare process instance efficiency through statistical analysis of these logs. Each process instance consisted of some logs, which record activities, resources, consuming time and cost in detail, as shown in Table 1. In addition, the waiting cost for the activity interval is 10 $/h.

4.1 Comparative analysis

The velocity updating in Formula (3) and the position update in Formula (4) of the general PSO algorithm involve the inertia weight $w(j)$, which mainly indicates the inertia of the current speed. Learning factors c_1 and c_2 mainly affect the closeness of the particle to the history optimal point and global optimal point, reflecting the communication between the particle and the swarm, the value which is usually between 0 to 2. r_1 and r_2 are random numbers, which are mutually independent between 0 and 1. As to the influence of the fixed weight of the PSO, according to the study of Trelea (2003), when the maximum speed V_{max} is less than 2, taking the inertia weight as 1 is more appropriate. But when the maximum speed is greater than 3, the inertia weight usually takes 0.8. And at this time the learning factors c_1 and c_2 are both set to 1 (Trelea 2003; Chen et al. 2012). The size of the particle swarm(the number of particles in a particle swarm)is commonly close to the general category of activities in the business process. Of course, bigger particle swarm size means increasing probability of obtaining the optimal solution, but also means the need to update more particles. Different from the ordinary PSO, the position $P_i(j)$and velocity $VE_i(j)$ of the particle are the factors which influence the particle swarm algorithm, thus continuously update them towards the local optimal solution to obtain the global optimal solution. In this paper, the hybrid particle swarm optimization algorithm, on the basis of general particle swarm optimization algorithm, redefines particle representation and updating method in the resource allocation

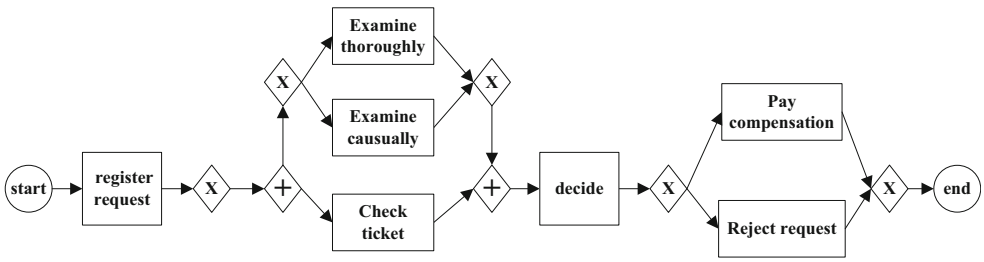

Fig. 1 Airline company compensation process

Table 1 Excerpt from compensation process logs

No.	CustomerName	Start time	End time	Activity	Operator	Cost ($)
1	Kai Lin	1–3-2016.08:00	1–3-2016.08:01	register request	Carol Wang	60
1	Kai Lin	1–3-2016.08:03	1–3-2016.08:03	Examine thoroughly	Sue Chen	165
1	Kai Lin	1–3-2016.08:04	1–3-2016.08:04	check ticket	Andrew Jin	70
1	Kai Lin	1–3-2016.08:05	1–3-2016.08:05	Decide	Ida Shi	325
1	Kai Lin	1–3-2016.08:06	1–3-2016.08:06	reject request	Fir Lu	80
2	Tina Tsai	1–3-2016.08:00	1–3-2016.08:03	register request	Carol Wang	60
2	Tina Tsai	1–3-2016.08:04	1–3-2016.08:05	check ticket	Andrew Jin	70
……..	……..	…….	…….	…….	……	……
2192	Xiaoyun Chang	6–3-2016.16:25	6–3-2016.16:35	check ticket	Carol Wang	70
2192	Xiaoyun Chang	6–3-2016.18:41	6–3-2016.19:22	decide	Ida Shi	325
2192	Xiaoyun Chang	6–3-2016.19:36	6–3-2016.19:42	reject request	Fir Lu	80
2193	Junbo Chow	6–3-2016.16:25	6–3-2016.16:26	register request	Carol Wang	60
2193	Junbo Chow	6–3-2016.16:29	6–3-2016.17:35	examine casually	Hubert Song	85
……..	……..	…….	…….	…….	……	……

process, and makes the improved PSO has stronger convergence and greater efficiency.

When the inertia weight of the particle swarm algorithm is not fixed, we get another PSO algorithm, which is the PSO algorithm with dynamic inertia weight (Pluhacek et al. 2013). The inertia weight (j) is set to one random number in line with randomly distributed numbers within a certain range, so to some extent, this dynamic inertia weight will not only help the local search, but also help to increase the search scope later. Herein, the inertia weight is set to $w(j)=w(j)_{max}+[w(j)_{max}-w(j)_{min}][\ rand()-0.5]$, where $w(j)_{max}$ is represented as the maximum value of the random inertia weight, and $w(j)_{min}$ represents the minimum value of the random inertia weight, and $rand()$ is the random number artificially set between 0 ~ 1 with evenly distribution.

When the learning factor c_1 and c_2 are non-fixed, the particle swarm optimization algorithm with dynamic learning factor is introduced (MAO et al. 2010). At this time, the two learning factors c_1 and c_2 will change with time. Herein, the change of dynamic learning factor is set as the following formula:

$$c_1 = c_{1s}-(C_{1s}-C_{1e})*cos(w(j)); c_2 = c_{2s}-(C_{2s}-C_{2e})*cos(w(j))$$

At this point, c_{1s} *and* c_{2s} represent the initial values of the two learning factors respectively, while c_{1e} and c_{2e} represent their end values respectively.

We compare the effectiveness of three different resource allocation algorithms: hybrid particle swarm optimization algorithm, PSO with dynamic inertia weight and PSO with dynamic learning factor. Through analysis of the logs, we can compare the effect of the algorithm. Firstly, we discuss the effectiveness of each resource allocation algorithm. Herein, we select seven airline ticket refund service: applications registration for compensation service, examine thoroughly, examine casually, check ticket, decide, pay compensation, and reject request. We got total time of each activity. The results are shown in Fig. 2. In it, 1 to 7 stand for applications registration for compensation service, examine thoroughly, examine casually, check ticket, decide, pay compensation, and reject request, separately.

As can be seen in Fig. 2, the average time of the hybrid particle swarm optimization algorithm is significantly less than the other dynamic particle swarm optimization algorithms. Because of various specific activities, time consuming of the algorithms is also different. In the activity of thorough examination and decision, because of computational complexity, the hybrid particle swarm algorithm can make the particles iterate quickly towards the optimal solution through cross-mapping, so the average time was more outstanding than that of the other two algorithms, which only changed their own attributes dynamically, lack of information exchange among particles, so they easily fell into the local optimal solution and were time-consuming. It can be seen that the hybrid particle swarm optimization algorithm can effectively reduce the total time in the multi-instance process.

We further verify the effectiveness of the algorithm by changing the number of resources and keeping the algorithm unchanged. Doubling the number of compensation lines in the experiment above. Input the compensation data obtained from the airline system, and got the results in Fig. 3.

It can be concluded from Fig. 3 that after doubling the activity resources, the restriction of the system was reduced and the average time of each activity was obviously reduced. On the whole, the hybrid particle swarm algorithm descended more in the whole. Especially in the most time-consuming, resource-consuming thorough review phase, the average time-consuming of the mixed particle algorithm declined more obvious. The hybrid particle swarm algorithm converges faster

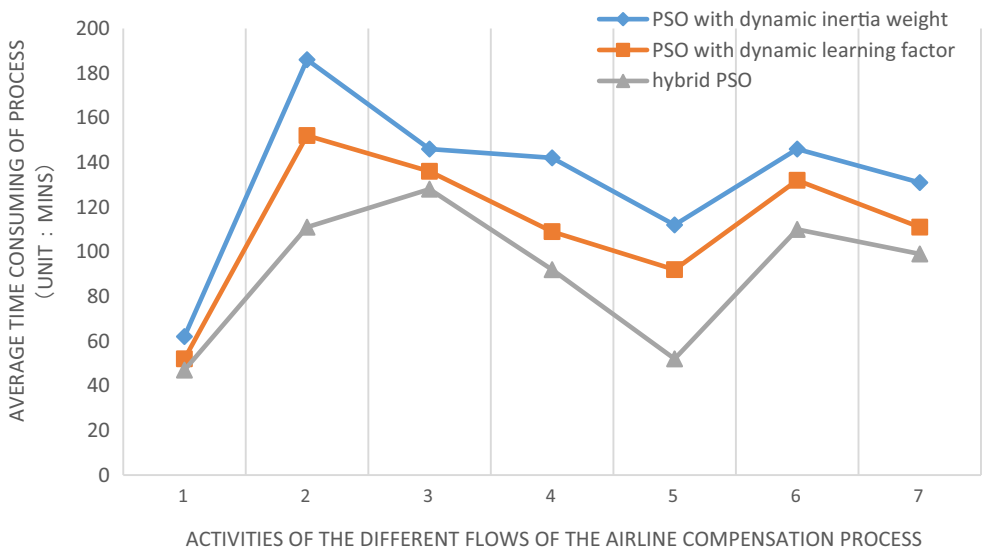

Fig. 2 Average time consuming for different processes using three resource allocation algorithms

after optimization, and converged to the optimal solution in the limited iterations.

4.2 Experimental analysis based on different resource allocation model

It is important to solve resource conflict and cost control in the multi-instance resource allocation model. Except the non-preemptive constrain of activities and the dynamic changes in weight, we also can focus on the process cost balance to design multi-instance resource allocation model so as to improve resource performance and so on. The minimum cost resource allocation model Xu et al. (2008) found all the resources with the lowest activity cost, checked the availability of the resource, and then verify whether the total process time exceeded the limit in condition of resource availability.

Average time consuming of different resource allocation model in the same process are shown in Fig. 4.

In Fig. 4, as to the multi-instance resource allocation model proposed by this paper, the average time-consuming of each activity was lower than the resource allocation model. Especially in the thorough examination and ticket checking stage, the resource allocation model proposed by this paper paid much attention to the weight of these two stages. Application rejection is the final decision stage as all prerequisite aspects needed to be completed. The model took into account the non-preemptive constraints of the activity sequence of each process, so it resulted in better results. The minimum-cost resource allocation model focused more on the balance between costs, considering resources availability after meeting the minimum cost requirement, but ignoring the activity priority, which easily led to the situation which meets a balance of costs but performed unsatisfactorily. We took the cross-validation and kept

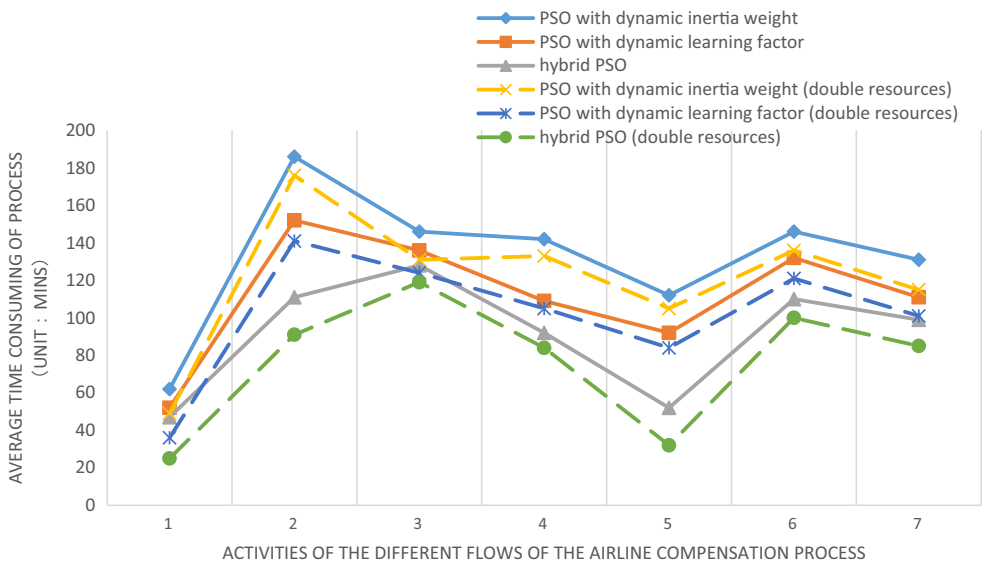

Fig. 3 Average time consuming of three algorithms and different resource numbers in the same process

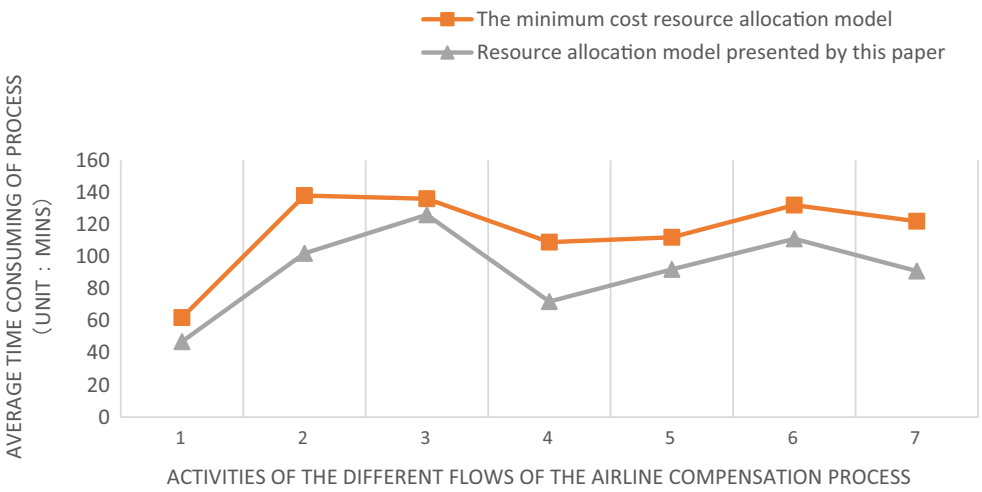

Fig. 4 Average time consuming of different resource allocation model in the same process

two multi-process resource allocation model unchanged, but change the number of resources. We got Fig. 5.

It can be seen from Fig. 5 that the average time of the two resource allocation model was both reduced after the number of resources increased. However, in the stage of compensation, the average consuming time of the minimum resource allocation model is improved obviously after replenishing the resource. The airline's backstage office finance department involved in the compensation phase, so it was more important to focus on the balance between the waiting cost and execution cost. But the stages casual examination and thorough examination are most resource-intensive. Therefore, we need to focus more on superior activities rather than the balance of cost.

In all, the improved hybrid particle swarm optimization (PSO) algorithm and the multi-process resource allocation model proposed in this paper have a good effect on resource allocation. The hybrid particle swarm algorithm has more efficient information sharing mechanism, which is more obvious with the increase of the number of resources. The cross-mapping mechanism of hybrid particle swarm optimization can make the particles update towards the optimal solution quickly. As to the priority of process activities, the model proposed in this paper can dynamically change the priority according to the current status of resource allocation and the number of postponed events, which can also lead to better allocation results. In this way, it can improve process performance and process structure.

5 Conclusions

In this paper, the PSO algorithm is introduced to find the global optimal solution of resource scheduling. The algorithm is an

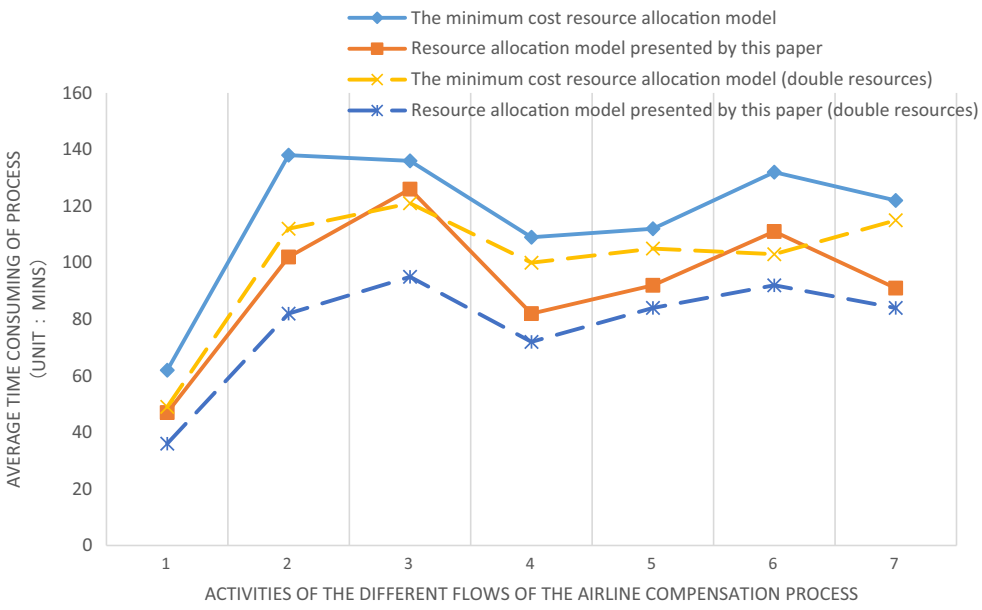

Fig. 5 Average time consuming of two resource model and different resource number in the same process

effective method to solve the NP-hard problem, which can find the global optimal solution. In addition, this paper proposes the improved PSO to accelerate the convergence of the particle and find the better way of resource scheduling. The resource allocation model proposed in this paper can fully consider the resource cost, time and other process performance evaluation indicators.

With the increasing complexity of practical industry processes, we also need further combine control flow logic and organization view. Therefore, in the future, we will further discuss various situations of scheduling and distribution of resources, and integrate the organizational view of the process to conduct more scientific resource allocation in a more comprehensive way.

References

Blackwell, T. (2007). *Particle Swarm optimization in dynamic environments. Evolutionary computation in dynamic and uncertain environments* (pp. 29–49). Berlin: Springer.

Chen C Y, Chuang C H, Wu M C. (2012). Combining concepts of inertia weights and constriction factors in particle swarm optimization. 2012 I.E. International conference on computational intelligence for measurement systems and applications, Tianjin, China, (pp. 73–76).

Cheng, K., Zhang, H., & Zhang, R. (2013). A task-resource allocation method based on effectiveness. *Knowledge-Based Systems, 37*(1), 196–202.

Delias, P., Doulamis, A., Doulamis, N., et al. (2010). Optimizing resource conflicts in workflow management systems. *IEEE Transactions on Knowledge & Data Engineering, 23*(3), 417–432.

Eberhart, R., & Shi, Y. (1998). *Comparison between genetic algorithms and particle swarm optimization*, *Evolutionary programming VII (pp. 611–616)*. Berlin Heidelberg: Springer.

Gao Y, Li S. (2010). Improved particle swarm optimization algorithm. Proceedings of 2010 international conference on computational intelligence and software engineering (pp. 1–4), Springer-Verlag NewYork.

Huang, Z., Lu, X., & Duan, H. (2011). Mining association rules to support resource allocation in business process management. *Expert Systems with Applications, 38*(8), 9483–9490.

Huang, Z., Lu, X., & Duan, H. (2012). A task operation model for resource allocation optimization in business process management. *IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans, 42*(5), 1256–1270.

Jin, X. U., Fei, S. M., Zhang, S. Y., et al. (2011). Adaptive particle swarm optimization for the project scheduling problem with dynamic allocation of resource. *Computer Integrated Manufacturing Systems, 17*(8), 1790–1797.

Liu, Z., Zhu, P., Chen, W., et al. (2015). Improved particle swarm optimization algorithm using design of experiment and data mining techniques. *Structural & Multidisciplinary Optimization, 524*(4), 1–14.

Ly, L., Rinderle, S., Dadam, P., et al. (2006). *Mining staff assignment rules from event-based data. Lecture Notes in Computer Science* (pp. 177–190). Berlin: Springer.

MAO, K.-f., Guang-qing, B. A. O., & Chi, X. U. (2010). Particle swarm optimization algorithm based on non-symmetric learning factor adjusting. *Computer Engineering, 36*(19), 182–184.

Petkov, S., Oren, E., & Haller, A. (2005). *Aspects in workflow management* (pp. 167–178). Galway: National University of Ireland.

Pluhacek, M., Senkerik, R., Davendra, D., et al. (2013). On the behavior and performance of chaos driven PSO algorithm with inertia weight. *Computers & Mathematics with Applications, 66*(2), 122–134.

Salman, A., & Ahmad, I. (2002). Particle swarm optimization for task assignment problem. *Microprocessor and Microsystems, 26*(8), 363–371.

Senkul, P., & Toroslu, I. H. (2005). An architecture for workflow scheduling under resource allocation constraints. *Information Systems, 30*(5), 399–422.

Smanchat, S., Indrawan, M., Ling, S., et al. (2011). A scheduler based on resource competition for parameter sweep workflow. *Procedia Computer Science, 4*(4), 176–185.

Trelea, I. (2003). The particle swarm optimization algorithm. *Information Processing Letters, 85*(6), 317–325.

Van Der Aalst, W. M. P., Barthelmess, P., Ellis, C. A., et al. (2001). Proclets: A framework for lightweight interacting workflow processes. *International Journal of Cooperative Information Systems, 10*(4), 443–481.

Wang, W.-B., Feng, Q., & Liu, D. (2011). Application of chaotic particle swarm optimization algorithm to pattern synthesis of antenna arrays. *Pier, 115*(1), 173–189.

Wang, X. Y., Zhang, G. X., Zhao, J. B., et al. (2015). A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning. *International Journal of Computers, Communications & Control, 10*(5), 732–745.

Xu, J., Liu, C., & Zhao, X. (2008). Resource allocation vs. In *Business process improvement: How they impact on each other. Procedings of Business Process Management, International Conference, Milan, Italy* (pp. 228–243).

Yang H, Wang C, Liu Y, et al. (2008). An optimal approach for workflow staff assignment based on hidden Markov models: on the move to meaningful Internet systems. Proceedings of 2008 Workshops on OTM. Springer Berlin Heidelberg, pp 24–26

Zhang, Y., & Gong, D. W. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing, 103*, 172–185.

Zhao W. (2014). Smart Business Process Management. Shanghai: Fudan University Press. (pp. 10–15)

Weidong Zhao is currently an Associate Professor at the Software School of Fudan University in China. He received his M.S. degrees in machine-electricity control and automation from Hefei University of Technology in 1998. He received his Ph.D. from the Institute of System Engineering at Southeast University, China, in 2001. He was a visiting scholar in Stern School of Business, New York University between 2011 and 2012. His current research interests include process intelligence, intelligent data analysis and decision support systems. He has also published some research papers in international conferences and journals such as PAKDD, Knowledge and Information Systems, Information Processing & Management etc.

Qingfeng Zeng is currently an Associate Professor at the Shanghai University of Finance and Economics (SUFE) in Shanghai, China. He received his Ph.D. Degree in Information Systems from the Fudan University in China in 2005. His current research interests focus on social media analysis, e-business transformation of tradition enterprises, and online brand management, he has published more than 20 papers in journals and conferences proceedings in recent years.

Guangjian Zheng is currently a master student in the Software School of Fudan University in China. His research interests include process intelligence, and data analysis.

Liu Yang received M.S. degrees from the Software School of Fudan University in China in 2015. Her research interests include process intelligence, and recommendation systems.