A particle swarm optimization algorithm for batch processing workflow scheduling

Yiping Wen[1,2], Zhigang Chen[2] , Tiemin Chen[3], Jianxun Liu[1], Guosheng Kang[1]
1. Key Lab of Knowledge Processing and Networked Manufacturing, Hunan University of Science and Technology, China
2. School of Information Science and Engineering, Central South University, Changsha, China
3. Hunan mobile communication company, Yongzhou, China

***Abstract*—Aiming at shortcomings in existing scheduling methods for batch processing workflow, this paper attempt to investigate and solve the optimization problem for grouping and scheduling multiple activity instances in batch processing workflow. A multiple objective optimal model of problem with constraints is presented firstly. Then, a discrete particle swarm optimization algorithm is proposed to produce a set of optimal Pareto solutions by optimizing the two objective functions simultaneously. The result of simulation experiment shows the effectiveness of this algorithm.**

I. INTRODUCTION

Workflow is the automation of business processes [1–3]. Scheduling of workflows is a problem of finding a correct execution sequence for the workflow tasks [4]. These tasks may be manual jobs or computer programs. The aim of batch processing in workflow is to achieve some run-time optimizations of business processes by scheduling activity (task) instances in multiple workflow cases of the same workflow type to run as a group. Therefore, its scheduling requirements are different from conventional workflow to some extent. Specifically, the following two differences can be identified in comparison to conventional workflow scheduling paradigms: 1) multiple instances of the same workflow activity in batch processing area should be vertically divided into groups and combined respectively before execution; 2) resources are allocated according to the status of those groups.

Literatures [2, 3] have discussed the implementation of batch processing in workflows and a dynamic scheduling method is also proposed. The proposed scheduling methods can help to reduce or save execution costs. However, the cost model for batch processing is not provided. Besides, cost can't be considered alone in practice because batch processing in workflows always leads to increasing the dwelling time of activity instances, which can influence the workflow time performance and the satisfactory degree of users. In fact, such scheduling is always subject to resource capacity and availability in practice, and the demands for resource capacity of activity instances are always non-constant. Grouping and scheduling multiple activity instances should guarantee their requirements to be satisfied. Therefore, the practical requirements for batch processing workflow scheduling strategies are: 1) the satisfaction of resources' capability requirements for activity instances; 2) a well balance between the objectives of minimizing the execution cost and the dwelling time of activity instances.

To meet the requirements above, we propose an optimization model on activity instances' total cost and dwelling time for batch processing workflow scheduling and an algorithm to resolve dynamic grouping and scheduling optimization with resource capacity constraint. The theory of intelligent optimization of particle swarm optimization is utilized to produce a set of optimal Pareto solutions with constraints by means of optimizing two objective functions simultaneously.

II. PROBLEM DEFINITION AND OPTIMIZATION MODEL

In this section, we first present some basic definitions such as activity instances, instance groups executors, et al., which are necessary for our further study. Based on this model, the problem we intend to investigate in this paper is formulated and modeled as a mixed-integer linear optimization problem. Detailed introductions of the mechanism of batch processing workflow and related definitions are referred to [2, 3].

Definition1. A activity instance for batch processing is a four triple $a_i =< A, GC_i, w_i, t_i >$, in which A is the name of workflow activity, GC_i stands for the grouping characteristics values of a_i , w_i describes its execution capability requirement for resource and t_i describes its waiting time, which is a time interval from its arrival to the time ready for scheduling.

Definition2. An instance group G_j consists of several instances of the same activity, which have the same grouping characteristics values and can be combined for execution. Its execution capacity requirement for resource can be denoted as $WG_j = \sum_{a_i \in G_j} w_i$.

Definition3. An executor e_k is a resource capable of executing instance group. It can be denoted by $e_k =< WE_k, c_k, TE_k >$, in which WE_k stands for its capacity, c_k and TE_k denote its cost and time duration of one execution respectively.

For simplicity, we assume that each instance group needs only one executor for execution, where there exist situations that several executors are needed for one instance group.

According to literatures [2, 3], activity instances of an instance group will be executed as a whole in batch processing workflows. Besides, several instance groups can be allocated to the same executor and executed in some order, which may lead to a time interval between the scheduling and their executions. Therefore, we have related definitions below.

Definition4. The cost of an activity instance depends the instance group it belongs to and the allocated executor to

978-0-7695-4864-7/12 $26.00 © 2012 IEEE
DOI 10.1109/CGC.2012.67

execute such instance group. That is, the total cost of activity instances in the same instance group is equal to the execution cost of the allocated executor.

Definition5. The dwelling time of an activity instance is equal to the sum of its waiting time and execution time, where its waiting time is the time interval from its arrival to the beginning time of its execution.

Based on the definition above, the problem investigated in this paper can be described as follows: given a set of activity instances with the same grouping characteristics values and activity name, the problem is that how to group and schedule these activity instances on a set of resources so that the total cost and dwelling time of activity instances are both minimized. In detail, such problem of M activity instances and N executors can also be formulated below:

$$F = \min(f_1, f_2) \quad (1)$$

s.t.

$$f_1 = \sum_{i=1}^{M}(\sum_{k=1}^{N}\sum_{m=1}^{M} mE_{ikm}TE_k + t_i) \quad (2)$$

$$f_2 = \sum_{k=1}^{N}\sum_{m=1}^{M}(\sum_{i=1}^{M} E_{ikm}c_k / \max\{1, \sum_{i=1}^{M} E_{ikm}\}) \quad (3)$$

$$\sum_{i=1}^{M} w_i E_{ikm} \le WE_k \ , \ 1 \le k \le N \ , \ 1 \le m \le M \quad (4)$$

$$\sum_{k=1}^{N}\sum_{m=1}^{M} E_{ikm} = 1 \ , \ 1 \le i \le M \quad (5)$$

$$E_{ikm} \in \{0,1\} \ , \ 1 \le i, m \le M \ , \ 1 \le k \le N \quad (6)$$

Equation (2) describes the total dwelling time of activity instances and equation (3) describes their total cost. Constraint set (4) ensures that the execution capacity requirement of all activity instances in each instance group can be satisfied by the allocated executor. Constraint set (5) ensures that each activity instance is assigned to exactly one instance group on one executor. Constraint set (6) imposes the binary restrictions on the decision variable, where $E_{ikm} = 1$ means that activity instance a_i is assigned to the mth instance group on executor e_k.

Obviously, such problem involves two potentially conflicting objectives and the situation in which improvement in one objective may cause deterioration in the other. Therefore, the solution for it should balance such tradeoffs, which is achieved when a solution cannot improve any objective without worsening the other objectives. It is called nondominated solution or Pareto optimal [5].

III. SCHEDULING ALGORITHM BASED ON MODIFIED PSO

Finding the optimal solution to the above defined problem is shown to be NP-hard according to [6]. Therefore, developing approximate algorithms to treat it is a good alternative comparing to the exact methods. The particle swarm optimization (PSO) is a heuristic search technique developed by Kennedy and Eberhart in 1995 [7, 8]. It is a swarm intelligence-based algorithm, which is inspired by the seeking behavior to a food source of organisms such as bird flocking and fish schooling. It is initialized with a population of random particles and their positions represent candidate solutions of some problem. Each particle moves its position toward a better solution according to its personal best position and the best position found by all other particles so far. Due to its advantages such as simplicity to implement and fast convergence, numerous studies have successfully employed PSO and its variants to tackle the complex optimization problems.

Generally, there are two issues to be considered when applying PSO to multi-objective optimization. The first one is how to select each particle's personal and global best positions so far to guide particle towards the better area over the search process. The second is how to maintain good positions found so far. The original PSO is proposed for the optimization of single objective continuous problem. Therefore, we propose a scheduling algorithm based on modified PSO to solve the problem under study, which named DGSO (Dynamic Grouping and Scheduling Optimization). It selects the global and personal best positions of particles from archives of feasible and infeasible solutions with some probability. By this way, feasible and infeasible solutions are both preserved to help explore the search space and increase the possibility of find optimal solutions. The pseudo code for the DGSO algorithm is described as Figure 1. The following sections provide its detailed implementation of step 1, 4, 5 and 6 respectively.

Algorithm DGSO

N – size of the swarm
t – iteration counter
T – maximum iteration times
Ar – archive of feasible solutions
Ar' – archive of infeasible solutions

1. Initialize randomly N particles, $t = 0$, set Ar and Ar' to be empty;
2. While $t \le T$ do
3. Evaluate the fitness and violation values of each particle;
4. Update the archives Ar and Ar';
5. Select the particles' personal best position and global from Ar and Ar';
6. Update the position of each particle;
7. $t = t + 1$;
8. End while
9. Output Ar;

Figure 1. Pseudo code for the DGSO algorithm

A. Coding strategy

Developing a suitable mapping between problem solution and PSO particle is one of the key issues in applying PSO. In this paper, we set up a search space of M dimensions for the problem of M activity instances and N executors. Each dimension has discrete set of possible integer values limited to $[0, M \times N - 1]$. For example, a solution to a problem of 5 activity instances and 3 executors may be expressed in an integer array as [14 4 14 4 3]. Consequently, the position of particle j in the ith iteration is represented as

$X_j^t = (x_{j1}^t, \cdots, x_{ji}^t, \cdots, x_{jM}^t)$, where x_{ji}^t is the position of the *j*th particle with respect to the *i*th dimension (or activity instance). Using these position values, instance groups can be formed. That is, activity instance a_i will be assigned to the *m*th instance group of executor e_k satisfying $k = N - \lfloor x_{ji}^t / M \rfloor$ and $m = M - (x_{ji}^t \% M)$.

B. Update of archives

As stated by [9], some infeasible individuals maybe exist near the global optimum and holds high fitness values. Further operations on them may help create better feasible offspring and guide the particles toward feasible regions. Therefore, we keep two archives to preserve feasible nondominated individuals and infeasible nondominated ones respectively. Besides, inspired by the direct comparison-method proposed by [10], the method employed in this paper to handle constraints and update archives can be described as Figure 2, where *viol*(*x*) is a function of measuring the magnitude that all inequality constraints are violated and ε is a predefined constant greater than zero.

Method UpdateA

Ar – archive of feasible solutions
Ar' – archive of infeasible solutions
1. For each particle *x* do
2. If *viol*(*x*)=0 then //satisfy all inequality constraints
3. If *x* is nondominated compared with all the elements in *Ar* then
4. delete all the elements in *Ar* that dominated by *x* ;
5. put *x* into *Ar* ;
6. End if
7. else if $viol(x) \le \varepsilon$
8. If *x* is nondominated compared with all the elements in *Ar'* then
9. delete all the elements in *Ar'* that dominated by *x* ;
10. put *x* into *Ar'* ;
11. End if
12. End if
13. End for

Figure 2. Pseudo code for the update of archives

C. Selection of personal and global best positions

The personal best position denotes the best solution achieved by the particle itself so far, while the global best position is the best solution obtained from neighbors of the particle so far. In this paper, the personal best position of each particle is selected from the archive of feasible solutions randomly. To balance the exploring capability to unknown feasible regions and known feasible regions, [11] introduces a dynamic strategy based on time-variation probability to select the global best positions of particle from the two archives. However, the spread density of solutions in archives is neglected. Therefore, we incorporate cluster analysis with such dynamic strategy to help enhance the spread and diversity of solutions. It can be described as Figure 3, where *cluster*(*S*) is a function of performing cluster analysis.

Method SelectG

t – iteration counter
T – maximum iteration times
Ar – archive of feasible solutions
Ar' – archive of infeasible solutions
1. Generate a uniform random number *r* between 0 and 1; p=0.5+0.3(1-2t)/T;
2. If *p*>*r* and *Ar'* is not NULL then
3. *S*= *Ar'*;
4. else
5. *S*= *Ar* ;
6. End if
7. *C*=*cluster*(*S*);
8. Output a random element from a random cluster of *C*;

Figure 3. Pseudo code for the selection of global best positions

D. Update of the particle's position

The original PSO requires users to tune control parameter such as acceleration coefficients and velocity clamping in order to obtain the desirable solutions. That is, the settings of these control parameters need to be adjusted for different problems. To avoid such strong dependence, [13] introduces a parameter-free method to update the particle's positions by particle's information, optimal solutions of population and individual. It is essentially a hybrid with the cross, mutation and selection operators of genetic algorithm. However, it discards infeasible new particles and may require multiple iterations to produce feasible ones. Therefore, we adopt such method with some modifications, where infeasible but well solutions are preserved to encourage exploring feasible regions. Its procedure is described as Figure 4, where the function *exchange* is to produce two new solutions by exchanging the element of X_j^t and P_j (g_j, X) when the corresponding elements of x (y, z) is 1. Besides, the particle's crowding distance in Step 5 is computed according to [12].

Method UpdateP

X_j^t – position of particle *j* in the *i*th iteration
P_j – personal best positions of particle *j* so far
g_j – global best positions of particle *j* so far
M – number of activity instances
1. Generate a random *M*-dimensional variable *x*, *y* and *z*, the elements of which are composed of 0 and 1;
2. Generate a random feasible particle *X*;
3. (T_1, T_2) = *exchange*(X_j^t, P_j, x); (T_3, T_4) = *exchange*(X_j^t, g_j, y); (T_5, T_6) = *exchange*(X_j^t, X, z);
4. If exists feasible particle in $T_1, \ldots, T_6$
5. select the feasible nondominated one with greatest crowding distance as X_j^{t+1};
6. else
7. select the infeasible nondominated one with smallest violation value as X_j^{t+1};
8. End if

Figure 4. Pseudo code for the update of particle's position

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results obtained by our algorithm. In our experiments, we simulate the express delivery process in Figure 5 to obtain related data

on activity instances and executors. It consists of eight activities: "Application of Parcel Delivery"(A), "Registration"(B), "Express Transferring"(C), "Express Distribution"(D), "Normal Transferring"(E), "Normal Distribution"(F) , "Signing of Receiver"(G) and "Informing Sender"(G). In this process, the transferring and distribution of parcels can be accomplished in a batch processing way for the receivers that have similar addresses. All the data are generated randomly within given ranges.

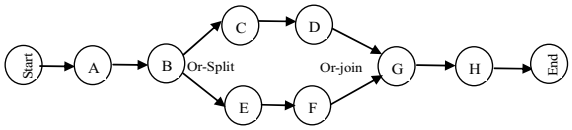

Figure 5. Simplified process of express delivery

We use the metric GD (i.e., Generational Distance) proposed by Zitzler [14] and execution time to evaluate the performance of algorithms. The GD measure denotes the closeness between the obtained solutions and real Pareto solutions. If GD is equal to zero, the obtained solutions all belong to real Pareto solutions. Besides, we construct an alternative multi-objective PSO-based algorithm (denoted as DMPSO for short) to compare with our algorithm. It employs the coding strategy of DGSO and the standard PSO [8] as a base, where the integral parts of particles' velocities are adopted to update their positions and particle that violates constraints will be randomly generated again until they are feasible. Its detailed settings are as follows: 1) acceleration coefficients: $c_1 = c_2 = 1.5$; 2) inertia weight: $w = 0.5$. All experiments are performed on a Personal Computer running with 3.2 GHz Intel Core CPU, 2 GB memory and Windows 7 OS platform. All algorithms are implemented by the software of Matlab 7.8, where size of the swarm is 10 and there exist four executors.

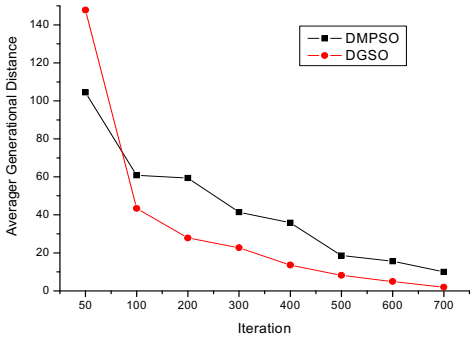

Figure 6. Comparison of the results on generational distance

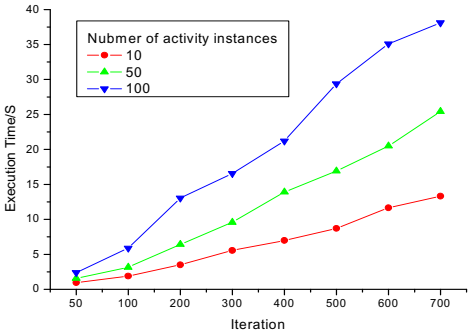

Figure 7. Comparison of the results on execution time

Fig. 6 and Fig. 7 shows the experimental results of algorithms, where 30 experiments are performed for 9 activity instances to obtain such average results. It can be easily concluded that the solutions obtained by DGSO has better average GD values and searching performance. However, its average execution time will grow greatly with the increment of iterations.

V. RELATED WORKS

Many methods have been proposed to solve the problem of workflow scheduling from different views. Generally speaking, existing methods can be divided into three categories: static approaches, dynamic ones and phased ones.

[4, 15] et al. have proposed some static approaches, by which all the activities involved in a workflow will be pre-scheduled before execution and the global optimal results of resource allocation can be reached under current static circumstance. However, they can not adapt to the uncertainties and dynamics such as changes of execution path and resources, concurrent execution of multiple workflow instances in practice.

Dynamic workflow scheduling approaches such as [16-18] et al. take those uncertainties and dynamics into account, by which the resources are allocated to the appropriate run-time activity instances. For example, [18] proposes a dynamic scheduling algorithm that consider constraints of resources and the match between the difficulty of tasks and the capability of resources. However, the results achieved by most dynamic methods are usually local optimal for single task and not global optimal for all the tasks.

Literature [19] proposes a phased scheduling method that combines the merits of static ones and dynamic ones. It divides activities of workflows into several groups to be scheduled in different phases, by which to reach global optimization to some extent and adapt to the requirements of dynamic circumstances.

Researches have been done on applying PSO to the optimization of workflow scheduling in several application areas. However, those proposed methods are not suitable for the optimization of batch processing workflow because of its special characteristics, some of which has been investigated in [2, 3]. Besides, as far as our knowledge, little researches

have been done on the discrete PSO algorithms for workflow scheduling with multiple objectives and resource constraints.

VI. CONCLUSION AND FURTHER STUDY

Scheduling of workflows is one of the most important and hottest problems in both research and practice fields. Aiming at shortcomings in existing scheduling methods for batch processing workflow, we investigate the optimization problem for its scheduling and construct a model for optimal grouping and scheduling of multiple activity instances in this paper. An algorithm DGSO (dynamic grouping and scheduling optimization) to solve such optimization problem is also presented, where the theory of intelligent optimization of particle swarm optimization is utilized to produce a set of optimal Pareto solutions.

Though experimental results have preliminarily illustrated the effectiveness of our DGSO algorithm, there still exist much improvement work to put it into practical applications. Our further study will include the following steps: 1) investigate the complex situations that several executors are needed for executing an instance group; 2) improve the performances of DGSO by exploring better coding and search strategies; 3) employ the business data from real-world application scenarios such as express delivery to further evaluate our proposed algorithms.

ACKNOWLEDGMENT

This paper was supported by NSFC, under grant number: 90818004, 61272063, 61073186, 61100054, 61073104, Program for New Century Excellent Talents in University, under grant number: NCET-10-0140, Excellent Youth Found of Hunan Scientific Committee, under grant number: 11JJ1011, Scientific Research Fund of Hunan Provincial Education Department, under grant number: 09K085, 12C0119, and Planned Science and Technology Project of Hunan Province of China, under grant number: 2011FJ3133.

REFERENCES

[1] M. Dumas, W.M.P. van der Aalst and A.H. ter Hofstede, *Process-aware Information Systems: Bridging People and Software Through Process Technology*, Wiley&Sons, 2005.

[2] J.X. Liu and J.M. Hu, "Dynamic Batch Processing in Workflows: Model and implementation", Future Generation Computer Systems, vol. 23, pp. 338–347, Mar. 2007.

[3] J.X. Liu, Y.P Wen, T. Li and X.Y. Zhang, "A data-operation model based on partial vector space for batch processing in workflow," Concurrency and Computation: Practice and Experience, vol. 17, pp. 1633-1639, 2011.

[4] P. Senkul and I. Toroslu, "An architecture for workflow scheduling under resource allocation constraints," Information Systems, vol. 30, pp. 399-422, 2005.

[5] C.A.C. Coello, D.A.V. Veldhuizen and G.B. Lamount, Evolutionary Algorithms for Solving Multiobjective Problems, Kluwer Academic Publishers, 2001.

[6] M. Garey and D. Johnson, Computers and intractability: a guide to the theory of NP-completeness, New York: W.H.Freeman and Company, 1979.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization", Proc. of IEEE International Conference on Neutral Networks, pp. 1942-1948, 1995.

[8] Y.H. Shi and R. Eberhart, "A modified particle swarm optimizer," Proc. of the IEEE International Conference on Evolutionary Computation, pp. 63-69, 1998.

[9] F. Gao, Q. Zhao, H. Liu and G. Cui, "Cultural particle swarm algorithms for constrained multi-objective optimization," Proc. of the 7th international conference on Computational Science (ICCS 07), pp. 1021-1028, 2007.

[10] W.F. Leong, "Multi-objective particle swarm optimization: integration of dynamic population and multiple- swarm concepts and constraint handing," Stillwater: Oklahoma State University, 2008.

[11] Y. Zhang, D.W. Gong, Y.Q. Ren et al, "Barebones multi-objective particle swarm optimizer for constrained optimization problems," Acta Electronica Sinica, vol. 39, pp. 1436-1440, 2011.

[12] K. Deb, A. Pratap and S. Agarwal, "A fast and elitist multi-objective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182-1971, 2002.

[13] Z. Shu, H.H. Chen and X.S. Luo, "A Web service dynamic selection method based on improved hybrid particle swarm optimization algorithm," Journal of Central South University (Science and Technology) , vol. 42, pp. 3086-3094, 2011.

[14] E. Zitzler, K. Deb and L. Thiele. "Comparison of multi-objective evolution algorithms: empirical results," Evolutionary Computation, vol. 8, pp. 173-195, 2000.

[15] S. Julia, F. Francielle and R. Valette, "Real time scheduling of workflow management systems based on a p-time Petri net model with hybrid resources," Simulation Modelling Practice and Theory, vol. 16, pp. 462-482, 2008.

[16] M. Shen, G. Tzeng, and D. Liu, "Multi-criteria task assignment in workflow management systems," Proc. of the 36th Hawaii International Conference on System Sciences, pp. 9, 2003.

[17] M. Lee, "Adaptive resource scheduling for workflows considering competence and preference," M. Gh. Negoita, et al. (Eds.), KES 2004, LNAI 3214, Springer-Verlag, Berlin, Heidelberg, pp. 723-730, 2004.

[18] Y.Q. Duan, J. Cao and S.S. Zhang, "Dynamic task scheduling method for workflow management," China mechanical engineering, vol. 13, pp. 233-235, 2002.

[19] Z.J. Xiao and M. Zhong, "A method of workflow scheduling based on colored Petri nets," Data & Knowledge Engineering, vol. 70, pp. 230-247, 2011.