Indicator-Based Multi-Objective Genetic Programming for Workflow Scheduling Problem

Qin-zhe Xiao, Jinghui Zhong(Corresponding Author), Wen-Neng Chen, Zhi-Hui Zhan, and Jun Zhang(Corresponding Author)
School of Computer Science and Engineering
South China University and Technology
Guangzhou, China
jinghuizhong@gmail.com,junzhanghk@qq.com

ABSTRACT

This paper proposes an Indicator-Based Multi-objective Gene Expression Programming (IBM-GEP) to solve Workflow Scheduling Problem (WSP). The key idea is to use Genetic Programming (GP) to learn heuristics to select resources for executing tasks. By using different problem instances for training, the IBM-GEP is capable of learning generic heuristics that are applicable for solving different WSPs. Besides, the IBM-GEP can search for multiple heuristics that have different trade-offs among multiple objectives. The IBM-GEP was tested on instances with different settings. Compared with several existing algorithms, the heuristics found by the IBM-GEP generally perform better in terms of minimizing the cost and completed time of the workflow.

CCS CONCEPTS

•**Computing methodologies** → *Heuristic function construction;*

KEYWORDS

Workflow scheduling, Multi-objective optimization , Genetic programming

ACM Reference format:
Qin-zhe Xiao, Jinghui Zhong(Corresponding Author), Wen-Neng Chen, Zhi-Hui Zhan, and Jun Zhang(Corresponding Author). 2017. Indicator-Based Multi-Objective Genetic Programming for Workflow Scheduling Problem. In *Proceedings of GECCO '17 Companion, Berlin, Germany, July 15-19, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3067695.3075600

1 INTRODUCTION

Workflow Scheduling Problem (WSP) is a popular and active research topic that has a range of practical applications [2]. Existing methods for WSPs generally can be classified into three categories: 1) to minimize the total cost of renting resources with a deadline constraint by using evolutionary algorithms [1, 2]; 2) regards the WSP as a multi-objective optimization problem and uses multi-objective evolutionary algorithms to solve the problem; 3) adopts manually designed greedy heuristics to assign each ready task to the proper resource [3]. However, the first two categories focus on solving static WSPs where the entire workflow is fixed, which limits their applications in dynamic environment (e.g., tasks may be added into or removed from workflow). Meanwhile, the third category focus on minimizing a single objective (i.e., completed time of the workflow), which limits their flexibility to balance trade-offs among multiple objectives.

To address the above issues, this paper proposes an Indicator based Multi-objective Gene Expression Programming (IBM-GEP) algorithm. In the IBM-GEP, the tasks are assigned to resources in a step-by-step manner. For each task and resource pair, a priority value is calculated by a priority function. The pair with the highest priority is selected at each step. To search for the best priority function, several low-level heuristics are defined at first and used as building blocks to construct the final heuristics. Then, we integrate the Indicator based Multi-objective optimization technique [7] with a recent publish GP variant (called SL-GEP) [6] to search for multiple heuristics that have different trade-offs among the execution cost and execution time of the workflow.

2 PROBLEM DEFINITION

A workflow W can be modeled as a directed acyclic graph where each node represents a task. An edge e_{ij} represents a computing dependency between task t_i and task t_j, where t_i is the parent task of t_j. A task can be executed only when all of its parent tasks are completed. For resource r_i, its rental cost per unit time is denoted as C_{r_i} and its capacity is denoted as P_{r_i}. Each resource r_i has a lease start time (LST_{r_i}) and a lease end time (LET_{r_i}). A task t_i must run on a resource and the starting time and completed time of t_i on r_j are denoted as ST_{t_i,r_j} and ET_{t_i,r_j}, respectively. The execution cost of the workflow is calculated by

$$cost = \sum_{i=1}^{|R|} C_{r_i} \cdot \lceil LET_{r_i} - LST_{r_i} \rceil \tag{1}$$

where R is the resource set. The total completed time is calculated by

$$time = max\{CT_{t_i} | t_i \in T\} \tag{2}$$

where T is the task set, CT_{t_i} is the competed time of t_i. The goal is to properly assign resources to tasks so as to minimize *cost* and *time*.

2.1 Proposed Algorithm

In the proposed method, the resources are assigned to tasks step-by-step until all tasks are completed. At each step, a priority function is used to calculate a priority value for each task in ready state (t_i) and resource pair. The pair that has the maximum priority will be selected and assigned to the task. In this way, finding the optimal scheduling strategy is converted to finding the optimal priority function. In this paper, we use GP to search for the optimal priority function. First, two low-level heuristics (LH) are defined as building blocks. The first LH is the average remain time of t_i:

$$ART_{t_i} = \frac{\sum_{rj \in R} ET_{t_i,r_j}}{|R|} + \frac{\sum_{tj \in A_{t_i}} (TT_{t_i,t_j} + ART_{t_j})}{|A_{t_i}|} \quad (3)$$

where A_{t_i} is the set of successors of t_i, TT_{t_i,t_j} is the data transfer time from t_i to t_j. The second LH is the average remain cost of t_i:

$$ARC_{t_i} = \frac{\sum_{rj \in R}(ET_{t_i,r_j} \cdot C_{r_j})}{|R|} + \frac{\sum_{tj \in A_{t_i}} ARC_{t_j}}{|A_{t_i}|} \quad (4)$$

Besides, six more features are also considered: 1) Execution time of the task running on the resource (ET); 2) Execution start time of the task if it is assigned to run on the resource (ST);3) Cost of the resource per unit time (UC); 4) Current lease time of the resource (LT); 5) Obtained cost if the task is assigned to run on the resource(OC); 6) Number of child tasks of the task (CN). Seven basic functions (i.e., $\{+, -, *, /, sin, cos, max(a, b)\}$) are used as linking functions to link the features to construct the final priority function. A recent published GP variant named SL-GEP [6] is adopt to find the optimal heuristics, due to its promising performance in complex optimization [4, 5]. As SL-GEP is designed for single objective optimization problem, we integrate it with the indicator-based multi-objective optimization technique, forming the IBM-GEP, to solve the formulated problem. In the IBM-GEP, the indicator-based selection strategy proposed in [7] is used to calculate fitness values of individuals.

Table 1: Comparison Results

Environment		Static WSPs		Dynamic WSPs	
Goal	Method	*cost*	*time*	*cost*	*time*
cost	Γ_{cost}	**21.18**	88.79	**21.38**	86.35
	PSO_{cost}	27.98 -	**15.88** +	N/A	N/A
	ACS_{cost}	22.32	22.35 +	N/A	N/A
time	Γ_{time}	27.85	**7.18**	**27.43**	**7.22**
	PSO_{time}	42.81 -	11.13 -	N/A	N/A
	ACS_{time}	**26.93**~	7.74~	N/A	N/A
	$HEFT$	33.21-	8.52 -	32.41-	8.63-

3 EXPERIMENTAL RESULTS

We generate a number of static workflows with different settings for training. The number of tasks of each workflow ranges from 50 to 200 randomly so that the data include small, medium and large scale workflows. The two best heuristics found by our method are as follows:

$$\Gamma_{cost} = ARC - max(ARC - UC - ST, ET) \quad (5)$$

$$\Gamma_{time} = sin(ST) - max(ST, ARC) - LT + UC - sin(sin(ST)) \quad (6)$$

where Γ_{cost} minimizes *cost* and Γ_{time} minimizes *time*. Then, these two heuristics are tested on 200 new instances with different settings, and compared with three methods, i.e., PSO [2], ACS [1] and HEFT [3]. The comparison results are shown in Table I, where $-$, $+$ and ~ represent that the competitor is significantly worse than, better than and similar to the evolved heuristics according to the students t-test at $\alpha = 0.05$. It can be observed that Γ_{cost} and Γ_{time} achieve the best *cost* and *time* respectively on both static and dynamic cases. It should be noted that PSO and ACS are not applicable for dynamic cases, as their solutions are problem-specific. These results demonstrate that the heuristics provided by IBM-GEP are general and effective for both static and dynamic WSPs.

4 CONCLUSIONS

This paper proposes a GP-based method to solve static and dynamic WSPs. Compared with several existing algorithms, the proposed method offers very promising performances in terms of minimizing cost and completed time of workflows. In future, we will try to apply our method to real world WSPs. In addition, we plan to extend our method by considering the complexity of the heuristic as another objective so as to find simpler and more effective scheduling heuristics.

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundations of China (Grant Nos. 61602181, 61622206, 61379061, 61332002, and 61402545), the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars (Grant No. 2014A030306038 and 2015A030306024), and the NSFC Key Program (Grant No. 61332002).

REFERENCES

[1] Zong-Gan Chen, Zhi-Hui Zhan, Hai-Hao Li, Ke-Jing Du, Jing-Hui Zhong, Yong Wee Foo, Yun Li, and Jun Zhang. 2015. Deadline Constrained Cloud Computing Resources Scheduling through an Ant Colony System Approach. In *Cloud Computing Research and Innovation (ICCCRI), 2015 International Conference on.* IEEE, 112–119.

[2] Maria Alejandra Rodriguez and Rajkumar Buyya. 2014. Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds. *IEEE Transactions on Cloud Computing* 2, 2 (2014), 222–235.

[3] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. 2002. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems* 13, 3 (2002), 260–274.

[4] Jinghui Zhong and Wentong Cai. 2016. A Hyper-Heuristic Framework for Agent-Based Crowd Modeling and Simulation. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 1331–1332.

[5] Jinghui Zhong, Wentong Cai, Michael Lees, and Linbo Luo. 2017. Automatic model construction for the behavior of human crowds. *Applied Soft Computing* 56 (2017), 368 – 378. DOI:http://dx.doi.org/10.1016/j.asoc.2017.03.020

[6] Jinghui Zhong, Yew-Soon Ong, and Wentong Cai. 2016. Self-Learning Gene Expression Programming. *IEEE Transactions on Evolutionary Computation* 20, 1 (2016), 65–80.

[7] Eckart Zitzler and Simon Künzli. 2004. Indicator-Based Selection in Multiobjective Search. In *International Conference on Parallel Problem Solving from Nature.* Springer, 832–842.