Optimizing Multiple QoS for Workflow Applications using PSO and Min-Max Strategy

Faruku Umar Ambursa[1]*, Rohaya Latip[1,2], Azizol Abdullah[1], Shamala Subramaniam[1]

[1]Communication Technology and Network Department, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia.

[2]Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia.

*Corresponding author: fuambursa@gmail.com

Abstract. Workflow scheduling under multiple QoS constraints is a complicated optimization problem. Metaheuristic techniques are excellent approaches used in dealing with such problem. Many metaheuristic based algorithms have been proposed, that considers various economic and trustworthy QoS dimensions. However, most of these approaches lead to high violation of user-defined QoS requirements in tight situation. Recently, a new Particle Swarm Optimization (PSO)-based QoS-aware workflow scheduling strategy (LAPSO) is proposed to improve performance in such situations. LAPSO algorithm is designed based on synergy between a violation handling method and a hybrid of PSO and min-max heuristic. Simulation results showed a great potential of LAPSO algorithm to handling user requirements even in tight situations. In this paper, the performance of the algorithm is anlysed further. Specifically, the impact of the min-max strategy on the performance of the algorithm is revealed. This is achieved by removing the violation handling from the operation of the algorithm. The results show that LAPSO based on only the min-max method still outperforms the benchmark, even though the LAPSO with the violation handling performs more significantly better.

Keywords: Service-orientated Computing; Workflow application; Scheduling; multiple QoS; Particle Swarm Optimisation (PSO).

1. Introduction

Workflow Management Systems (WFMSs) enable seamless binding of computing infrastructures from geographically distributed service domains. The primary aim of a QoS-aware WFMS is to provide transparent and efficient access to these distributed services as well as provide the needed QoS satisfaction to EUs with diverse QoS requirements. The delivery of QoS is a challenging task due to the heterogeneous and dynamic nature of the service platforms. In addition, large number of service providers exists in the web market, each with different economic and social attributes. Therefore, efficient workflow scheduling strategies are needed to provide the needed level of QoS satisfaction.

To address this demand, numerous researches have been reported in literature [1]–[8]. These researches have been categorized based on: the number of user QoS objectives considered in scheduling model; the method for handling user constraints and preferences; and the kind of optimization strategy employed [6]. The schemes proposed in [4], [5] and [6] supported six user objectives, constraints and

preferences. However, economic cost is not considered in [4]. The scheduling model in [5] took into account the user parameters that included time, cost, reliability, availability, security and reputation, which were aggregated into single objective function using weighted sum approach. The authors proposed a rotary chaotic PSO (RCPSO) algorithm to optimize the multi-objective function for high quality scheduling solutions. The RCPSO hybridized chaos method and PSO optimization technique. However, it results to high violation of user-defined constraints in tight situations. To address this problem, recently a new scheme is proposed in [6] that enhanced both the scheduling model and optimization method of RCPSO. The new scheduling model is based on dual function approach, where a function for violation detection is incorporated in addition to the RCPSO's fitness function for better violation handling. Then a Look-ahead PSO (LAPSO) optimization algorithm is proposed, which integrated a min-max heuristic in PSO. The idea of the optimization approach was to use a deterministic guided approach for PSO-based solution search. Simulation results showed a great potential of LAPSO algorithm to handling user requirements even in tight situations. In this paper, the performance of the algorithm is anlysed further. Specifically, the impact of the min-max strategy on the performance of the algorithm is revealed. This is achieved by removing the violation handling from the operation of the algorithm. The results show that LAPSO based on only the min-max method still significantly outperform the benchmark, even though the LAPSO with the violation handling performed far better.

2. The LAPSO Algorithm

In this section, the overview of LAPSO Algorithm is explained. LAPSO algorithm is based on the interaction between two important components: constraint handling algorithm and hybrid metaheuristic optimization algorithm. These components are described in detail in the following subsections.

2.1. Violation handling Algorithm

A multi-QoS constraint handling algorithm is proposed to minimize constraints violations. The purpose of the algorithm is to aid in selecting good scheduling solutions from the pool of solutions produced by PSO. The algorithm is based on the introduced scheduling model with dual functions: the violation function λ_ϕ and objective function $\mathcal{W}_\phi$. These models are defined in Equation (1) and Equation (2) for evaluation and selection of solutions. The violation function tracks cumulative violation levels of scheduling solutions with the aim of meeting user *hard* requirements. The objective function computes the aggregate value of the multiple objective functions. It evaluates solutions based on their cumulative achievement for the optimization of user *soft* requirements. More details of the selection algorithm can be found in [6].

$$minimize\ \lambda_\phi = \delta_{\pi_{dln}} + \delta_{\pi_{bgt}} + \delta_{\pi_{rel}} + \delta_{\pi_{avl}} + \delta_{\pi_{sec}} + \delta_{\pi_{rep}}$$
$$s.t\ \lambda_\phi = 0. \quad (1)$$

$$maximize\ \mathcal{W}_\phi = w_1\left(1 - \frac{M_\phi}{\mathcal{D}}\right) + w_2\left(1 - \frac{C_\phi}{\mathcal{B}}\right) + w_3\left(\frac{R_\phi}{\mathcal{R}} - 1\right) + w_4\left(\frac{A_\phi}{\mathcal{A}} - 1\right) + w_5\left(\frac{Sec_\phi}{\mathcal{S}} - 1\right) + w_6\left(\frac{Rep_\phi}{\mathbb{R}} - 1\right) \quad (2)$$

2.2. Hybrid PSO and Min-Max

LAPSO algorithm is based on PSO algorithm and min-max heuristic. PSO is a population-based heuristic strategy for a solution search. It belongs to the class of swarm intelligence and shares similarity with GA in terms of starting with random solutions [9]. The LAPSO algorithm optimizes scheduling solution by using PSO as baseline algorithm to iteratively search for and produce good solutions. In each iteration, the best solution produced by PSO is enhanced using the auxiliary min-max heuristic. The min-max heuristic improves PSO's solution by relieving a task randomly assigned to a poor service and reassigning it to an alternative service that improves its QoS. The min-max also reduces CPU time by minimizing the number of invalid solutions produced by PSO.

2.3. Particle Updating

PSO's particle updating stage is the core phase where min-max heuristic is employed in LAPSO. In the beginning, the algorithm adopted the RD rule and history velocity cancellation methods [5]. Then a task remapping strategy based on the min-max heuristic is designed to improve the global best solution ($gBest$) found at the end of each iteration. Firstly, the fitness function is de-aggregated into the multiple distinct objective functions. In order to improve the global best solution, the idea is to maximise the QoS dimension that has the minimum normalised QoS value from among all its six QoS dimensions. This is based on the premise that the quality of each solution is composed of its value with respect to each of the six QoS objectives. The problem is considered a max-min problem based on the Decision Theory and is expressed as:

$$\max\left[\min\left(\mathcal{W}_{\phi}^{m}, \mathcal{W}_{\phi}^{c}, \mathcal{W}_{\phi}^{r}, \mathcal{W}_{\phi}^{a}, \mathcal{W}_{\phi}^{s}, \mathcal{W}_{\phi}^{rep}\right)\right] \quad (3)$$

A task remapping algorithm called Look-ahead QoS-aware Task-Remapping Algorithm (LATR) (Algorithm 1) is designed based on min-max heuristic. The whole idea of this approach is that maximising the inferior mapping string results to maximisation of the inferior parameter, which in turn improves the quality of the entire solution. The operation of the algorithm is detailed in [6].

Algorithm 1: Look-ahead QoS-aware Task-Remapping Algorithm (LATR) [6]

Input: ϕ_{best}^{t}
Output: ϕ_{best}^{+}
1 $\sigma_{inferior} \leftarrow Search()_{QoSSet_{best}}$
2 $\Phi_{\mathrm{F}}^{\sigma_{inferior}} \leftarrow Search(\sigma_{inferior})_{MapSet_{best}}$
3 counter ← 1 //first attempt to improve $\Phi_{\mathrm{F}}^{\sigma_{inferior}}$
4 $\phi_{determ} \leftarrow \phi_{best}^{t-1}$
5 $\Phi_{\Gamma}^{\sigma_{inferior}} \leftarrow Search(\sigma_{inferior})_{MapSet_{determ}}$
6 **If** $\Phi_{\Gamma}^{\sigma_{inferior}} > \Phi_{\mathrm{F}}^{\sigma_{inferior}}$ **then**
7 $AS \leftarrow s_{\Gamma}^{h}$
8 temp ← copy of ϕ_{best}^{t}
9 $temp \leftarrow Remap(h, AS)$
10 Calculate λ_{temp} using Equation (1)
11 **If** $\lambda_{best} = 0$ **then** //best solution is feasible
12 **If** $\lambda_{temp} = 0$ **then**
13 Calculate $\mathcal{W}_{temp}$ using Equation (2)
14 **If** $\mathcal{W}_{temp} > \mathcal{W}_{best}$ **then**
15 $\phi_{best}^{+} \leftarrow Remap(h, AS)$
16 **Else** //$\lambda_{best} > 0$ (best is infeasible)
17 **If** $\lambda_{temp} < \lambda_{best}$ **then**
18 $\phi_{best}^{+} \leftarrow Remap(h, AS)$
19 **Else**
20 **If** counter ← 1 **then**
21 counter ++ //second attempt to improve $\Phi_{\mathrm{F}}^{\sigma_{inferior}}$
22 $\phi_{determ} \leftarrow rand(current\ PSO\ swarm)$
23 **Go to line** 5
24 **Return** ϕ_{best}^{+}

3. LAPSO Algorithm without Violation Handling (VH)

In this section, the LAPSO algorithm without VH is highlighted. LAPSO without VH means removing Equation (1) in the operations of LAPSO. The equation is responsible for detecting the presence of violation of user requirements in scheduling solutions, which has significant influence in the performance of LAPSO. This is because it directs the LAPSO algorithm towards choosing solutions with less or zero violation.

The purpose of removing VH is to see how LAPSO performs based on only the operations of min-max heuristic and PSO technique. For evaluation and selection of solutions, only Equation (2) is used. The Equation (2), which was proposed in [5], aggregate the six QoS functions, constraints and preferences into one objective function. It suffers high violation because violation is not explicitly detected by its constraint handling method.

The LAPSO without VH is shown in Algorithm 2. In the algorithm, line 3 shows the modification, where the fitness of a solution $f(x_i^t)$ is computed using Equation (2) only. Detail of LAPSO can be found in [6].

Algorithm 2: Look Ahead PSO without VH

1 **foreach** particle
2 initialize x_{ij}^t and v_{ij}^t
3 compute $f(x_i^t)$ using Equation (2)
4 initialize $pBest_i^t$ with x_i^t
5 compute $gBest_j^t$
6 **while** stopping condition not satisfied **do** //next iteration
7 **foreach** particle
8 update v_{ij}^{t+1}
9 update x_{ij}^{t+1}
10 compute $f(x_i^t)$ using Equation (2)
11 update $pBest_i^{t+1}$
12 update $gBest_j^{t+1}$
13 //decode PSO's previous and current global best particles to schedules
14 $\phi_{best}^t = gBest_j^t$
15 $\phi_{best}^{t+1} = gBest_j^{t+1}$
16 invoke Algorithm 1 to improve the schedule ϕ_{best}^{t+1}
17 **return** ϕ_{best}^+

4. Simulation Experiment

In this section, a simulation was carried out to evaluate the performance of LAPSO without VH. Results were compared with the original LAPSO and the RCPSO algorithms. The algorithms were tested using a business workflow application as used in reference [5] and [6]. As in [6], to each of the tasks in the workflow, depending on the test case, a medium scale was randomly assigned – in the range 30 to 60 – or a large scale – in the range 60 to 120 – set of service instances. The values for the six QoS parameters of each service instance were also randomly generated [5], but they followed the rule that for the same task the service cost depended on the values of the other parameters. The results were compared with respect to Cumulative Violation Rate (CVR).

4.1. Results and Discussion

4.1.1. *Test Case 1: Moderate Constraints.* In this experiment, all the user-defined hard constraints, including budget, are set to moderate and the grid scale is set to large and then to medium. From the

simulation results, the Fig. 1 (a) (large grid scale) and Fig.1(b) (medium grid scale) show performances of the schemes.

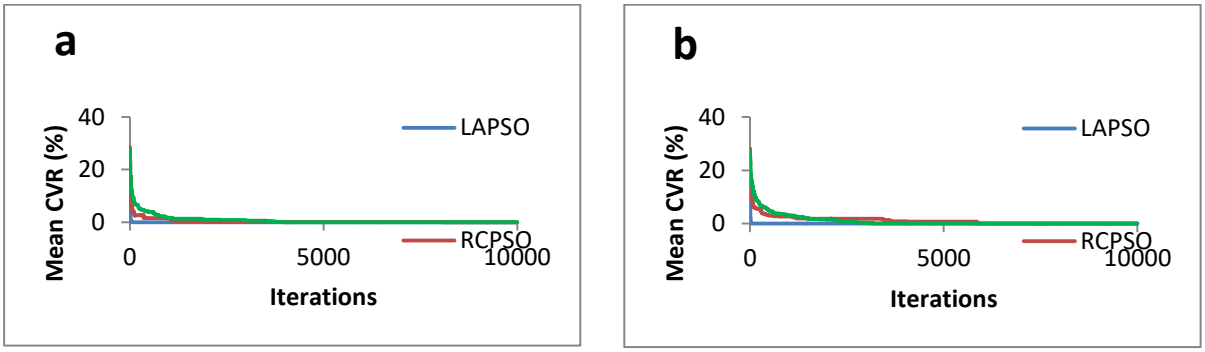

Fig. 1: (a) Violation Rate under moderate constraints and large grid scale (b) Violation Rate under moderate constraints and medium grid scale

In Figure 1(a) and (b), the three algorithms have managed to fulfil all the user-defined *hard* constraints with no violation (0% violation). However, LAPSO and LAPSO without VH achieved faster convergence than RCPSO.

4.1.2. *Test Case 2: Tight Constraints.* In this experiment, all the user-defined hard constraints, including budget, are set to tight and the grid scale is set to large and then to medium. From the simulation results, Fig. 2c (large grid scale) and Fig. 2d (medium grid scale) show performances of the schemes.

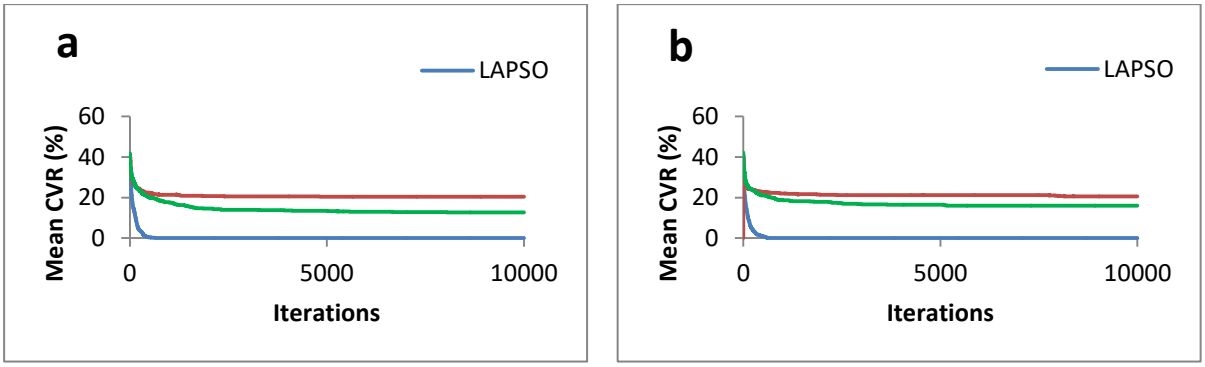

Fig. 2. (a) Violation Rate under tight constraints and large grid scale (b) Violation Rate under tight constraints and large grid scale

In Figure 2(c) and Figure 2(d), LAPSO has managed to achieve 0% violation in both large and medium scale cases. LAPSO-without-VH recorded 12.72% and 16% mean CVR, in Fig. 2(c) and (d), respectively. Lastly, RCPSO suffered violations of 20.4% and 20.55% mean CVR for the large and medium scales, respectively.

From the above results, it can be observed that: firstly, LAPSO without VH still outperforms the benchmark RCPSO algorithm, this is because LAPSO uses deterministic strategy in improving global best solution, while RCPSO uses stochastic methods; secondly, the original LAPSO is by far better than both LAPSO without VH and RCPSO. This is because the LAPSO without VH uses the RCPSO's fitness model, which leads PSO search towards solutions with relatively high violation, making it more difficult for min-max method, in LAPSO, to perform effectively.

5. Conclusion

In this paper further analysis of the LAPSO algorithm is presented. In LAPSO dual functions are used for violation handling and multi-objective optimisation. For optimization of the multiple objectives, PSO is used as baseline and a min-max based method is used for deterministic search. Previous simulation results indicated that LAPSO significantly minimizes violation of user constraints even in tight cases – when user demands are high. In this study, LAPSO is examined without its violation handling method in order to reveal the effectiveness of the synergy of its components. The results show that LAPSO, based on RCPSO's scheduling model and the min-max method, still outperforms the benchmark, even though the LAPSO with its violation handling method performed far better.

References

[1] R. Aron, I. Chana, and A. Abraham, "A hyper-heuristic approach for resource provisioning-based scheduling in grid environment," *J. Supercomput.*, pp. 1427–1450, 2015.

[2] H. Khajemohammadi, A. Fanian, and T. A. Gulliver, "Efficient Workflow Scheduling for Grid Computing Using a Leveled Multi-objective Genetic Algorithm," *J. Grid Comput.*, vol. 12, no. 4, pp. 637–663, Aug. 2014.

[3] K. Kianfar, G. Moslehi, and R. Yahyapour, "A novel metaheuristic algorithm and utility function for QoS based scheduling in user-centric grid systems," *J. Supercomput.*, vol. 71, no. 3, pp. 1143–1162, 2015.

[4] A. A. Pourhaji Kazem, H. Pedram, and H. Abolhassani, "BNQM: A Bayesian Network based QoS Model for Grid service composition," *Expert Syst. Appl.*, vol. 42, no. 20, pp. 6828–6843, 2015.

[5] Q. Tao, H. Y. Chang, Y. Yi, C. Q. Gu, and W. J. Li, "A rotary chaotic PSO algorithm for trustworthy scheduling of a grid workflow," *Comput. Oper. Res.*, vol. 38, no. 5, pp. 824–836, May 2011.

[6] F. U. Ambursa, R. Latip, A. Abdullah, and S. Subramaniam, "A particle swarm optimization and min–max-based workflow scheduling algorithm with QoS satisfaction for service-oriented grids," *J. Supercomput.,* 2016.

[7] M. Wang, L. Zhu, and K. Ramamohanarao, "Reasoning task dependencies for robust service selection in data intensive workflows," *Computing*, vol. 97, no. 4, pp. 337–355, Dec. 2015.

[8] X. Wang, C. C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Futur. Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1124–1134, Oct. 2011.

[9] F. U. Ambursa and R. Latip, "A survey: Particle swarm optimization-based algorithms for grid computing scheduling systems," *J. Comput. Sci.*, vol. 9, no. 12, pp. 1669–1679, 2013.