Location-Aware and Budget-Constrained Application Replication and Deployment in Multi-Cloud Environment

Tao Shi, Hui Ma and Gang Chen
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: {tao.shi, hui.ma, aaron.chen}@ecs.vuw.ac.nz

Sven Hartmann
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
Email: sven.hartmann@tu-clausthal.de

***Abstract*—To gain technical and economic benefits, enterprise application providers are increasingly moving their workloads to the cloud. With the increasing number of cloud resources from multiple cloud providers at different locations with differentiated prices, application providers face the challenge to select proper cloud resources to replicate and deploy applications to maintain low response time and high quality of user experience without running into the risk of over-spending. In this paper, we study the global-wide cloud application replication and deployment problem considering the application average response time, including particularly application execution time and network latency, subject to the budgetary control. To address the problem, we propose a GA-based approach with domain-tailored solution representation, fitness measurement, and population initialization. Extensive experiments using the real-world datasets demonstrate that our proposed GA-based approach significantly outperforms common application placement strategies, i.e., NearData and NearUsers, and our recently proposed hybrid GA approach.**

***Keywords*-Location, budget, multi-cloud, application deployment, application replication, genetic algorithm**

I. Introduction

According to Forbes [1], 41% of enterprise workloads will move to public cloud platforms by 2020. Gartner forecasts that the worldwide public cloud service revenue will exceed 300 billion U.S. dollars in 2021 [2]. In the thriving public cloud market, *multi-cloud* is becoming a popular cloud ecosystem, because it allows cloud users to share the workload across multiple cloud service providers to achieve high quality of service with low operation cost and also avoid vendor lock-in [3]. A third-party, *broker*, is usually responsible for the deployment and management of multiple clouds on behalf of the cloud users, i.e., the individual consumers or enterprise application providers [4]. In this paper, we focus on *application providers* who offer domain specific applications to end users by using multi-cloud Infrastructure as a Service (IaaS) made available by cloud providers.

It is typical for application providers to offer business applications abstracted as workflows [5]. These applications consist of a set of specific functionalities, i.e., constituent services. In the remaining of this paper, we use *application services* or *services* to refer constituent services.

The deployment of applications in cloud is critical to application providers since it affects the Quality of Service (QoS) of applications [6]. In [6], a business workflow scheduler is proposed to achieve the optimal execution time and cost in federated heterogeneous clouds. On the one hand, the network latency between users/datasets and application services in different locations significantly affect the application performance in multi-cloud [7]. Therefore, to evaluate accurately the performance of applications, we consider the *average response time*, including both the application execution time and the network latency, in this paper. On the other hand, in order to improve user Quality of Experience (QoE), it is essential to deploy application replicas at multiple locations to bring the application close to its users [8]. With the goal to minimize the application response time, we take *application replication* into account. Because enterprises are often more interested in budgetary control to ensure that their actual costs adhere closely to their financial plan [9], we formulate the application replication and deployment problem as a budget-constrained optimization problem.

Evolutionary computation (EC) is a promising approach for effective resource management in cloud computing [10]. For example, driven by a population-based solution improvement framework, genetic algorithm (GA) has been widely used to select proper multi-cloud resources for scalable cloud application deployment [7], [11]. However, we need new technical innovations to effectively apply GAs for the application deployment problem. (1) Considering the application replication, we first need to determine the *replication plan*, including both the amount and the locations of application replicas. However, the GA-based approaches for application deployment in the literature can only support applications with a pre-determined number of replicas [12]. (2) The numerous cloud resources, e.g., virtual machines (VMs), at globally distributed data centres result in large and complex search space, which can impair the scalability and applicability of the existing GA-based approaches [10]. (3) The randomly generated initial population of GAs cannot produce high-quality final solutions reliably in our experiments. Consequently, we need to propose a problem-specific seeding strategy to ensure the quality of deployment solutions [13], [14].

To deal with the above challenges, we propose a new two-level optimization approach under the GA framework for effective multi-cloud application replication and deploy-

978-1-7281-8786-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ICWS49710.2020.00022

ment: (1) Because replication plans significantly impact the performance and deployment cost of applications, the first level optimization, i.e., application replication, is realized by GA. That is, GA is used to optimize replica location selection from the global data centres. (2) To reduce the complexity/size of the search space, the second level optimization, i.e., VM types selection for service deployment, is performed as the part of the fitness evaluation process. Particularly, after deploying all application services to the cheapest VMs that satisfy the capacity requirement, we progressively upgrade the service hosting VM types with the largest benefit in terms of response time and cost as long as there is budget available. (3) We initialize the population of our GA-based approach by adding a heuristic-based solution to improve solution quality and convergence speed. The main contributions of this paper are summarized as follows:

Firstly, we formally define the application replication and deployment problem in multi-cloud with the goal to minimize the average response time subject to a budget constraint. To the best of our knowledge, this is the first study in the literature on the multi-cloud application deployment problem at the global scale considering application replication and budget impact (justified further in Section II).

Secondly, we propose a novel GA-based approach to the Application Replication and deployment Problem (GA-ARP), featuring a newly designed and domain-tailored solution representation, fitness measurement, and population initialization.

Finally, to evaluate the proposed approach, we collect the information about VM capacity and pricing offered by the three global industry-leading cloud providers, i.e., Amazon, Microsoft, and Alibaba, and conduct extensive experiments with different budget constraints. Experimental results show GA-ARP can significantly improve application performance comparing with the common application placement strategies, i.e., NearData and NearUsers [15], and our recently proposed approach for budget-constrained application deployment, i.e., H-GA [7].

The remainder of this paper is organised as follows: Section II provides relevant works about the multi-cloud application replication and deployment problem and GA-based approaches. Section III defines the problem. Section IV presents the proposed GA-ARP. Section V describes the design of experiment and analyses the simulation results. Section VI concludes the paper.

II. Related work

This section introduces the related work about application replication and deployment in multi-cloud and reviews the GA-based approaches, because GAs have been widely studied in the research community and are considered as dominant methods to solve service/data location allocation problems in the literature.

A. Multi-cloud Application Replication and Deployment

In recent years, cloud computing forges a novel paradigm for flexible and scalable delivery and deployment of enterprise-scale business applications. A business application is usually an abstraction of well-defined business process [5]. In many business domains, tens of thousands or even millions of application requests with varied user distributions must be fulfilled on a daily basis. Queuing models are appropriate to capture the performance of a business application [7]. In this paper, we model the request processing through a queuing model and focus on IaaS to select appropriate VMs from global cloud marketplace for deploying application services. As shown in Figure 1, application deployment in multi-cloud is usually managed by a third party known as the multi-cloud broker [3], which can provide a single entry point to multiple clouds.

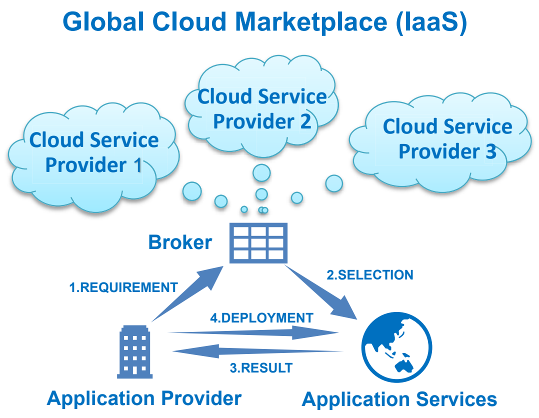

Figure 1. A simplified multi-cloud broker architecture based on [11]

The service deployment problem is similar to the multi-cloud service brokering problem [11], [16]. Because network latency has significant impact on the performance of cloud services, the service brokering problem was explored to cope not only with the deployment cost, but also the network latency between users and cloud data centres in [14], [16]. However, the service brokering problem in [14], [16] assumes that all services as being independent, which is different from our workflow-based applications with interdependent functionalities.

In our previous work [7], composite applications deployment problem is studied for deploying several business applications with shared constituent services in multi-cloud. The problem jointly considers both the performance and budget control, regardless application replication. In industry, deploying application replicas as close to target users as possible is widely exercised to maintain low response time [8]. Considering the application replication, the application deployment problem shares some similarities with the data placement and replication problem, which has also received some research attention in cloud [17], [18]. However, different from the problem about data placement and replication, the application replication and deployment problem must decide both the locations and the types of cloud resources.

The budgetary control is of immense practical significance for application providers, which emphasizes more on performance optimization within a given budget [7]. Some heuristics, e.g., Loss, Gain [19], and ScaleStar [20], have been proposed for workflow scheduling and explicitly recognized budget control as a key constraint. However, these strategies take no account of the network latency caused by different geographical locations of cloud resources.

B. *GA-based Approaches*

GAs are meta-heuristics inspired by the process of natural selection. In [21], two GA-based approaches are proposed to decide what and where to replicate some of the data objects at multiple sites to avert undesired long delays experienced by end-users. To optimise social media data placement and replication in cloud data centres, a GA-based approach is presented to minimise monetary cost while satisfying latency requirements for all users in [15].

GA-based approaches also have been proposed to solve a similar problem, web service location-allocation problem, that aims to find optimal locations for deploying multiple instances of atomic web services such that the overall response time becomes minimal for a target user group. In [22], an enhanced GA with self-adaptive feature and memory filter has been proposed to tackle the web service location-allocation problem. However, like the data placement and replication problem, web service location-allocation problem takes no account of service selection at the same location.

III. Problem Description

In this paper, we study the problem of Location-aware and Budget-constrained Application Replication and Deployment in Multi-cloud (LBARDM) with the aim to deploy one business application at multiple locations to minimize the average response time of the application within a given budget. This section introduces basic notations, concepts, and formulation of the problem.

A predefined business application can be represented as a Direct Acyclic Graphs (DAG). A DAG is denoted by $G(S, E)$ with S as a set of nodes where $s_l \in S$ represents application service and E as a set of directed edges where $e_{mn} \in E$ connects s_m with s_n. Based on e_{mn}, s_m is called the parent service of s_n and s_n is the child service of s_m. If a service s_l requires access to a dataset during processing, we denote this data-access service as $\dot{s_l}$. An application has exactly one dummy starting service that has no parent services and one dummy ending service that has no child services, which are denoted as s_{start} and s_{end} respectively. An example has been shown in Figure 2. In the example, service $\dot{s_5}$ is a data-access service (represented in a blue circle).

We refer to [7] and consider a set of user centres $\mathcal{U} = \{U_0, ..., U_k, ..., U_{|\mathcal{U}|-1}\}$, which represents the centres of global user groups. We assume that new application requests from user centres are generated according to Poisson distributions, following the common practice in many previous works [23]. The request rate γ_k denotes the average amount of requests from a user centre U_k during a unit period of time.

We assume that an application is replicated as one cohesive unit in different data centres among multiple cloud

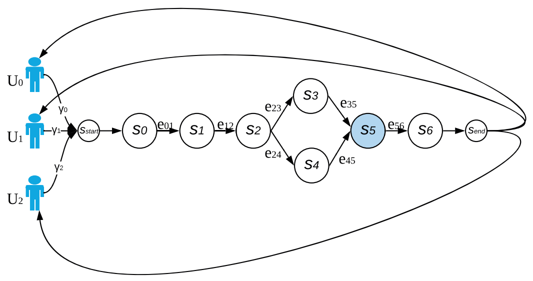

Figure 2. An example business application.

providers. That is, once a specific data centre is selected for replication, all services in the application are deployed onto the VMs provided by this data centre. To distinguish different services, we use *service instances* to represent the deployed services in application replicas. Deploying all the services of an application to the same data centre makes the inter-services network latency negligible [24].

We consider a set of data centre locations $\mathcal{A} = \{A_0, ..., A_i, ..., A_{|\mathcal{A}|-1}\}$ and use $R \subseteq \mathcal{A}$ to represent the application replication plan, where $|R| \geq 1$ and $A_r \in R$ specifies the location of one application replica according to the replication plan. We denote the application replica allocated at A_r as W_r. For each data centre location $A_i \in \mathcal{A}$ and each user centre $U_k \in \mathcal{U}$, we use lt_{ik} to represent the network latency between A_i and U_k. When there are multiple application replicas, we assume all the requests from U_k are assigned to the application replica with the lowest network latency following [22]. In this regard, the request rate from U_k to W_r can be determined as:

$$\theta_{rk} = \begin{cases} \gamma_k & \text{if } lt_{rk} = \underset{i:A_i \in R}{\text{argmin}}\{lt_{ik}\} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The *workload* of W_r is measured by the combined request rates from user centres:

$$\lambda_r = \sum_{k=0}^{|\mathcal{U}|-1} \theta_{rk}. \tag{2}$$

In W_r, the service instance of s_l, i.e., s_{rl}, maintains an independent queue for pending requests arriving from the corresponding user centres or its parent service instances. The *workload* of s_{rl} is identical to its corresponding application replica, i.e., $\lambda_{rl} = \lambda_r$.

We consider a collection of VM types $\mathcal{V} = \{V_0, ..., V_j, ..., V_{|\mathcal{V}|-1}\}$ available in $A_i \in \mathcal{A}$. Based on the related study in [7] and industry practice [25], we assume that each service must have exclusive use of one VM to minimize interferences between any two service instances. Refer to [7], the average resource consumption per request over a long sequence of requests for the same service instance is considered highly stable. Therefore, we can obtain the average time rt_{rl} for s_{rl} to process a single request excluding data access time. For the instances of data-access services, we assume that request processing begins with data access, i.e., once the requested data becomes available, the service instances will perform necessary processing on the data. The *capacity* of a service instance, measuring the amount of requests processable by s_{rl} per time unit, can be calculated by:

$$\mu_{rl} = \frac{1}{rt_{rl} + dt_{rl}}, \tag{3}$$

where dt_{rl} is the data access time required for s_{rl} to access its datasets remotely. If s_{rl} is the instance of a non-data-access service, $dt_{rl} = 0$.

We follow [7] and model the operation of each individual service instance as an independent $M/M/1$ queue. According to Little's Law [26], the service instance's *average* request processing time is:

$$pt_{rl} = \frac{1}{\mu_{rl} - \lambda_{rl}}, \tag{4}$$

where $\mu_{rl} > \lambda_{rl}$, as the guarantee that the workload of any service instance will not exceed its capacity. In the remaining of this paper, we use *capacity feasible* VMs to refer the VM types on which service instances' capacities are greater than the requested workloads.

We calculate the average response time of each application replica W_r, i.e., ART_r, as follows:

$$ART_r = 2ut_r + MS(W_r), \tag{5}$$

where $2ut_r$ is the round-trip average network latency between W_r and the corresponding user centres, and $MS(W_r)$ is the execution time of W_r, also called makespan denoting the time for request processing to reach s_{end} of W_r. Concretely, ut_r can be determined by:

$$ut_r = \frac{\sum_{k=0}^{|\mathcal{U}|-1} \theta_{rk} \cdot lt_{rk}}{\lambda_r}. \tag{6}$$

Note that ART_r is affected by the network latency lt_{rk} between user centres and data centres and the data access time dt_{rl} required by data-access service instances. They are usually determined by several non-changeable factors in the communication network, such as the physical distance among the datasets, user centres, and data centres.

We use DC_{rl} to represent the deployment cost of s_{rl}. The total deployment cost (TDC) and total response time (TRT) can be calculated as follows:

$$TDC = \sum_{r:A_r \in R} \sum_{l=0}^{|\mathcal{S}|-1} DC_{rl}, \tag{7}$$

$$TRT = \frac{\sum_{r:A_r \in R} \lambda_r \cdot ART_r}{\sum_{k=0}^{|\mathcal{U}|-1} \gamma_k}. \tag{8}$$

Therefore, LBARDM problem aims to minimize TRT of all the application replicas, as defined in eq. (9), subject to budgetary constraint.

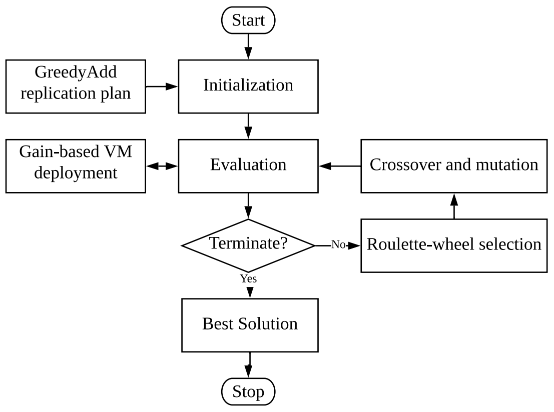

Figure 3. The overall process of GA-ARP.

$$\begin{aligned} \min \quad & TRT, \\ \text{subject to} \quad & TDC \leq b, \end{aligned} \tag{9}$$

where b is the budget specified by the application provider. We refer to [19] and define b in eq. (10):

$$b = C_{cheapest} + f \cdot (C_{highest} - C_{cheapest}), \tag{10}$$

where $C_{highest}$ and $C_{cheapest}$ are the total costs of deployments by using either the most expensive VM type or the cheapest capacity feasible VM types for all the services respectively. $f \in [0,1]$ is the budget factor to determine how tight the budgetary control is. The larger f is, the more budget the application provider has.

IV. The proposed approach: GA-ARP

In multi-cloud environment, there are a large number of VM types available at different data centre locations, resulting in extremely large search spaces. Considering the application replication, LBARDM problem becomes more complex. We first design problem-specific solution representation and fitness measurement to optimize application replication and VM types selection in parallel. The new designs can substantially reduce the size of the search space. In recent years, problem-specific seeding strategy demonstrates promising performance for GA-based algorithms [13]. Consequently, we add a heuristic-based application replication plan to the initial population of our GA approach. The plan starts with deploying the first copy of the application in the cheapest region and repeatedly adds new replicas guided by the criteria of reducing TRT as far as possible. This process will continue until no improvement can be found.

The overall process of GA-ARP is shown in Figure 3. After encoding replication plans, an initial population is generated randomly with a heuristic-based plan (GreedyAdd described in Subsection IV-C). For each encoded plan, our algorithm optimizes VM types selection of all application replicas and calculates TRT of the deployment solution as the fitness value (described in Subsection IV-B). After the roulette-wheel selection, the crossover and mutation operations are performed to generate new plans. The population is evolved iteratively until the predefined maximum number of generations is reached. Finally, the best chromosome is decoded to return the final replication and deployment decision. In the following subsections, we provide detailed description of our algorithm, including representation, fitness measurement, population initialization, and genetic operators.

A. Chromosome Representation

We use chromosomes to encode application replication plans evolved by GAs. Our chromosome is a bits string, and the length of the string is $|A|$. For the example in Figure 4, there are totally 10 data centre locations, and 3 application replicas are deployed at A_0, A_2, and A_8.

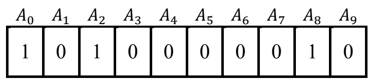

Figure 4. An example chromosome

B. Fitness Measurement

For the randomly generated and newly evolved chromosomes, i.e., replication plans, we first optimize VM types selection with the minimal TRT within b by the following steps:

(1) Select the cheapest capacity feasible VM types for all service instances;

(2) Calculate the *benefit* for each service instance by upgrading its current VM type to one with higher capacity, e.g., from AWS m5.large to m5.xlarge. Based on Gain [19], the well-known DAG scheduling heuristic considering the budget constraint, the *benefit* can be calculated as follows:

$$benefit = \frac{TRT_{old} - TRT_{new}}{TDC_{new} - TDC_{old}}, \tag{11}$$

where TRT_{new} and TDC_{new} are the response time and deployment cost after upgrade;

(3) Upgrade the service instance with the largest benefit iteratively until the TDC reaches the given budget b.

Then we assign the chromosome a fitness value, i.e., TRT of the final application deployment solution, based on eq. (8). The fitness value of a chromosome indicates its chance of survival and reproduction in the next generation.

C. Population Initialization

Generally, GAs generate the initial population randomly to ensure diversity. However, the strategy cannot produce high quality solutions reliably in our experiments. Therefore, we propose a heuristic-based method, i.e., GreedyAdd, to generate a solution as one of the initial population discussed below. The remaining population will be initialized randomly. That is, we randomly select some data centre locations as the application replica locations. The seeding strategy has been shown to improve the quality of evolved solutions substantially.

GreedyAdd for population initialization: We first generate the cheapest replication plan, i.e., deploying the first copy of the application to the cheapest data centre. Progressively, one additional replica is allocated to one of the remaining data centre locations in an attempt guided by the objective of choosing the right replica with the highest reduction in TRT until no better plans can be found within the given budget. The calculation of TRT follows the procedure as described in Subsection IV-B.

D. Crossover Operator

We apply the crossover operator to evolve replication plans. During crossover operation, the parent chromosomes are divided into two parts by one random cut-off point. Their offsprings are generated by exchanging tail parts of the two parents as the example in Figure 5. Note that to avoid generating an invalid plan, i.e., the amount of application replicas is 0, the crossover is only performed on bit locations with 1s in the parent chromosomes. Accordingly, those bit locations with 0s can be easily filled up after crossover.

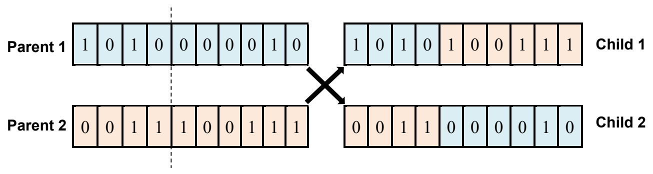

Figure 5. An example of crossover operation

E. Mutation Operator

Mutation operator can introduce local improvement to previously evolved plans by making a small change on an offspring in a random way. In particular, a bit is selected randomly and inverted, i.e. if the bit is 1, it is changed to 0 and vice versa. Similarly, the operation may generate an invalid plan without application replicas. In this case, we invert a new random bit.

V. Experiment

We conduct a series of simulation studies to examine the performance of GA-ARP by comparing them with the common application placement strategies, i.e., deploying the application at the data centre where the dataset is stored (NearData for convenience) and deploying application replicas at all datacentre locations to ensure all users close to the application (NearUsers for convenience) [15]. Also, we choose our recently proposed hybrid GA-based approach for budget-constrained composite applications deployment, i.e., H-GA [7], to benchmark the performance of GA-ARP.

A. Datasets

We used 10 business applications obtained from real-world studies reported in [27]. These workflows provide realistic depictions of many practical business applications, ranging from online shopping to travel planning. According to the latest worldwide IaaS public cloud services market share [2], the real VM type descriptions and pricing schemes

Table I
APPROACH PERFORMANCE COMPARISON FOR LBARDM PROBLEM (TRT IN MS.)

	f	0.1		0.2		0.3		0.4		0.7	
Workflow	Approach	TRT	NoR	TRT	NoR	TRT	NoR	TRT	NoR	TRT	NoR
Workflow1	NearData	281.26	1	276.32	1	273.51	1	272.86	1	272.39	1
	NearUsers	N.A.	15	174.04	15	118.34	15	96.81	15	84.52	15
	H-GA	281.26±0	1	276.2±0	1	273.51±0	1	272.86±0	1	272.14±0	1
	GA-ARP	180.08±0	3↑	122.28±0.55	5	95.19±0.53	6↓	80.8±0.18	7↓	72.99±0.26	8↓
Workflow2	NearData	279.49	1	277.63	1	272.88	1	272.47	1	272.19	1
	NearUsers	N.A.	15	137.17	15	107.62	15	84.94	15	77.36	15
	H-GA	279.49±0	1	274.32±0	1	272.88±0	1	272.38±0	1	271.83±0	1
	GA-ARP	165.95±1.52	5↑	108.49±0.58	7↑	85.15±0.52	7	76.04±0	7↓	69.62±0	9
Workflow3	NearData	282.48	1	282.17	1	282.11	1	282.11	1	282.11	1
	NearUsers	N.A.	15	511.47	15	487.4	15	482.98	15	481.29	15
	H-GA	279.74±0	1	274.46±0	1	272.97±0	1	272.49±0.03	1	271.86±0.02	1
	GA-ARP	189.35±0	2	149.27±0	5↑	142.02±0.62	5.9↑	140.76±0.04	6.2↑	134.42±3.63	5.2↑
Workflow4	NearData	280.72	1	276.77	1	273.53	1	272.71	1	272.06	1
	NearUsers	236.52	15	128	15	94.97	15	84.49	15	76.87	15
	H-GA	280.72±0	1	274.5±0	1	273.3±0.01	1	272.73±0.06	1	272.28±0	1
	GA-ARP	153.17±0	4↑	102.37±0	6	83.23±0	8	73.86±0.11	7.1↓	69.08±0.23	7.8↓
Workflow5	NearData	281.13	1	277.19	1	274.03	1	272.67	1	271.96	1
	NearUsers	229.3	15	117.32	15	94.6	15	85.87	15	78.24	15
	H-GA	280.62±0.13	1	274.12±0	1	272.99±0.03	1	272.52±0.04	1	272.04±0.03	1
	GA-ARP	148.74±0.08	4.1↑	97.97±0.61	6.7↑	79.59±0.51	6↓	71.61±0.23	7↑	65.76±0	9
Workflow6	NearData	281.22	1	274.91	1	272.58	1	271.92	1	271.57	1
	NearUsers	163.82	15	97.91	15	80.85	15	71.43	15	67.27	15
	H-GA	276.92±0	1	273.72±0.02	1	272.62±0.05	1	272.2±0.05	1	271.85±0.04	1
	GA-ARP	128.82±0.03	6↑	88.06±0	8	72.88±0.02	9.2↓	64.62±0.06	8.2↓	61.33±0.08	9.7↓
Workflow7	NearData	284.9	1	281.83	1	279.39	1	275.19	1	272.61	1
	NearUsers	N.A.	15	470.81	15	391.4	15	373.03	15	118.93	15
	H-GA	283.02±0	1	276.71±0.01	1	273.49±0	1	272.97±0.06	1	272.36±0.15	1
	GA-ARP	186.76±1.31	2.9↑	136.82±0	4	105.47±0.42	5.2↑	91.08±0.12	5.9↓	82.96±0.16	6.8↑
Workflow8	NearData	282.89	1	281.7	1	277.41	1	275.29	1	272.54	1
	NearUsers	N.A.	15	209.55	15	132.08	15	112.93	15	101.31	15
	H-GA	280.65±0	1	276.31±0	1	273.54±0.02	1	272.95±0.06	1	272.48±0.14	1
	GA-ARP	187.95±0.95	2.9↑	134.2±1.71	6.5↑	101.49±0.79	5.5↓	91.37±0.58	7.1↓	82.59±0.62	7.4↑
Workflow9	NearData	283.71	1	278.13	1	277.12	1	272.95	1	272.3	1
	NearUsers	243.33	15	179.72	15	108.21	15	93.46	15	83.08	15
	H-GA	280.67±0	1	274.38±0	1	273.17±0.07	1	272.74±0.05	1	272.18±0.02	1
	GA-ARP	157.48±1.98	5.7↑	116.84±0.34	7.2↑	91.76±1.04	8.1↑	81.76±0.09	8.5↓	75.5±0.29	9.2↓
Workflow10	NearData	280.21	1	276.66	1	275.59	1	272.94	1	272.27	1
	NearUsers	N.A.	15	409.97	15	373.31	15	361.71	15	107.49	15
	H-GA	278.81±0	1	274.63±0	1	273.15±0.08	1	272.7±0.12	1	272.16±0.02	1
	GA-ARP	178.31±0.05	3.5↑	120.9±0.97	7↑	94.63±0	5	82.01±0	5	78.85±0.43	6↓

of three leading cloud providers, i.e., Amazon Web Service (AWS), Microsoft Azure and Alibaba Elastic Compute Service (ECS)[1], are collected. 24 different VM types (8 from AWS, 8 from Azure, and 8 from Alibaba) have been included in the experiments. We also consider a total of 15 locations for major AWS, Azure and Alibaba data centres. To simplify problem, we assume the application-dependent dataset is only stored in Virginia (US East) and postpone data replication and synchronization to the future. Furthermore, we adopt 82 user centres from 35 countries on 6 continents in Sprint IP Network[2] to simulate the global user community.

To evaluate the network latency between users/data and deployed applications, we use real-world observation of network latency from Sprint[3] IP backbone network databases.

B. Simulation Settings and Algorithm Parameter Settings

Refer to [7], we apply Facebook subscribers statistics to simulate the distribution of application requests from user centres γ_k. By June 2017, there are approximately 1.33 billion Facebook subscribers from all of 35 countries considered in our simulation[4]. We consider the user scale for business applications as 1/1000th of Facebook subscribers to simulate the application providers with million users, because we have more applications at this scale in the multi-cloud market, e.g., Xero[5]. Each user makes 25 requests on average daily following [28]. Accordingly, the application request rate is close to 400 requests per second (that is, approximately 32 millions requests daily). Refer to [29], we implement 5 budget factors, i.e., 0.1, 0.2, 0.3, 0.4, and 0.7 for each application.

For the two GA-based approaches, our parameter settings include: population size is 100, maximum generation is 50, which are sufficient for the search process to converge. The crossover rate and mutation rate are 0.9 and 0.1 respectively following common practice in the literature [30]. To compare the results, we consider the mean and standard deviation of TRT after running each experiment 30 times. All the experiments[6] for running four competing approaches are performed on the same computer with Intel Core i7-8700 CPU (3.2 GHz and 15.5 GB of RAM).

[1]Amazon EC2: https://aws.amazon.com/ec2/instance-types/; Microsoft Azure: https://azure.microsoft.com/en-us/services/virtual-machines/; Alibaba ECS: https://www.alibabacloud.com/product/ecs.

[2]https://www.sprint.net/network_maps.php

[3]https://www.sprint.net/performance/

[4]https://www.internetworldstats.com/facebook.htm

[5]https://www.xero.com/nz/

[6]The source code can be accessed at: https://github.com/qingdaost/LBARDM

C. Budget Compliance

We first analyze the approaches in terms of their capabilities of meeting the pre-specified budget requirements. For the tightest budget, i.e., $f = 0.1$, NearUsers cannot generate adequate solutions for 6 out of 10 applications (shown as N.A. in Table I). While as f increases, all four approaches are capable of deploying applications within b, even for the application with 14 different services (workflow 10).

From above results, we can conclude that it is unpractical to replicate applications at all available regions when the allowed budget is low. In such a situation, even by selecting the cheapest capacity feasible VMs for all service instances, the TDC cannot be brought under the specific budget.

D. TRT Evaluation

The TRT generated by the four approaches for ten applications with five budget factors are shown in Table I. From the simulation results, we observe that NearUsers outperforms NearData for 34 out of 50 cases. It shows the potential to reduce TRT by deploying application replicas at more data centre locations, rather than the data source location alone. For example, NearUsers significantly reduces TRT of workflow 4, 5, 6, and 9 comparing with NearData at all the budget levels.

However, NearUsers cannot always produce short TRT. Except the over-budget scenarios, NearUsers also suffers the worst performance when deploying workflow 3 ($f = 0.2, 0.3, 0.4, 0.7$), workflow 7 ($f = 0.2, 0.3, 0.4$), and workflow 10 ($f = 0.2, 0.3, 0.4$). These results suggest that placing application replicas close to all users is not effective enough, because some application replicas only serve a small number of users, which causes the leased VMs heavily under-utilized.

In most cases, H-GA can achieve shorter TRT than NearData by the population-based solution improvement framework for VM deployment. As NearData, however, H-GA cannot deploy more application replicas close to users. GA-ARP can generate the best replication and deployment solutions in terms of TRT among all four competing approaches with respect to all workflows. From Table I, we can calculate its average improvement over NearData, e.g., 69.82% for workflow 6 (highest) and 46.37% for workflow 3 (lowest), over NearUsers, e.g., 71.1% for workflow 3 (highest) and 11.93% for workflow 6 (lowest), and over H-GA, e.g., 69.66% for workflow 6 (highest) and 44.97% for workflow 3 (lowest). More importantly, the TRT achieved by GA-ARP on all workflows are consistently within 200 ms, even under the tightest budget. The performance can maintain good user experience [17]. We also find that GA-ARP has very small standard deviation, confirming its stability and reliability for LBARDM problem.

E. Replica Amount

Next, we investigate the effectiveness of GA-ARP by comparing the number of replicas of the final application replication plan with that of the GreedyAdd heuristic. The average number of replicas across 30 runs, NoR, is also shown in Table I.

There are 39 cases where NoR is different for plans found by GA-ARP and GreedyAdd. Concretely, GA-ARP selects more replicas in 23 cases shown as $\uparrow$ in Table I, and fewer replicas in 16 cases shown as $\downarrow$. In comparison to the GreedyAdd plans, basically, the NoR of a better solution tends to be bigger when the budget is tight ($f = 0.1, 0.2$) while smaller when the budget is loose ($f = 0.4, 0.7$). This is because GreedyAdd gets trapped to some sub-optimal solutions.

We take a closer look at some cases where the replication plans generated by GA-ARP and GreedyAdd have the same NoR, e.g. workflow 1 with $f = 0.2$. GA-ARP can further reduce TRT by 4.51%. By observing the replication plan generated by GreedyAdd in this example, the five application replicas are progressively placed in Virginia, Singapore, Sao Paulo, London, and Seoul. However, GA-ARP can escape from this local optimal point, and replace the data centres in Singapore and Seoul with Tokyo and Mumbai. The change of replica locations improve the application performance.

F. Further Analysis

The average computation time of GA-ARP with respect to different budget level is within 3000 seconds. Therefore, GA-ARP is more practical for the scenarios where applications have stable or predictable demand, e.g., user distribution and request rates.

We plot the evolution of TRT corresponding to some cases (Figure 6 and 7) from the four competing approaches. As evidenced in these figures, GA-ARP can converge within 50 generations with evident improvement.

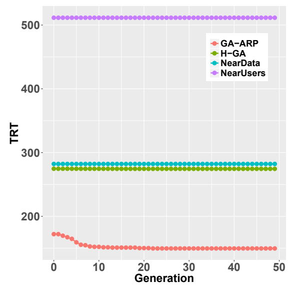

Figure 6. Workflow 3, k = 0.2

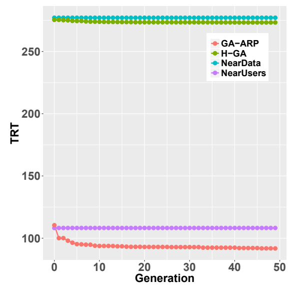

Figure 7. Workflow 9, k = 0.3

VI. Conclusions and Future Work

In this paper, we studied the business application replication and deployment in multi-cloud that considers the average response time and budget control. To address the problem, we propose a novel GA-based approach which can select VMs from multi-cloud for service instances of application replicas within a given cost budget so that the average response time is minimized. Due to the problem complexity, GA-ARP is designed with problem-specific solution representation, fitness measurement, and population initialization, which are effective for breeding excellent offsprings. The experiments based on the datasets collected from real world cloud providers, network environment, and business applications show that GA-ARP outperforms some existing application placement strategies and our recently proposed GA-based approach for LBARDM.

Future work will consider multi-objective evolutionary algorithms to solve the multi-cloud application replication and deployment problem.

References

[1] L. Columbus. (2018) 83% of enterprise workloads will be in the cloud by 2020. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/

[2] C. Stamford. (2019) Gartner forecasts worldwide public cloud revenue to grow 17% in 2020. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020

[3] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 7, 2014.

[4] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame *et al.*, "Optimis: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[5] R. Xu, Y. Wang, H. Luo, F. Wang, Y. Xie, X. Liu, and Y. Yang, "A sufficient and necessary temporal violation handling point selection strategy in cloud workflow," *Future Generation Computer Systems*, 2018.

[6] G. Jung and H. Kim, "Optimal time-cost tradeoff of parallel service workflow in federated heterogeneous clouds," in *2013 IEEE 20th International Conference on Web Services*. IEEE, 2013, pp. 499–506.

[7] T. Shi, H. Ma, G. Chen, and S. Hartmann, "Location-aware and budget-constrained service deployment for composite applications in multi-cloud environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1954–1969, 2020.

[8] G. Shipley. (2018) Best practices for deploying your apps in the cloud. [Online]. Available: https://developer.ibm.com/articles/cl-best-practices-deploying-apps-in-cloud/

[9] S. Carter, N. J. MacDonald, and D. C. Cheng, *Basic finance for marketers*. Food & Agriculture Org., 1997, vol. 1.

[10] M. Guzek, P. Bouvry, and E.-G. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 53–67, 2015.

[11] L. Heilig, E. Lalla-Ruiz, and S. Voß, "A cloud brokerage approach for solving the resource management problem in multi-cloud environments," *Computers & Industrial Engineering*, vol. 95, pp. 16–26, 2016.

[12] C. Guerrero, I. Lera, B. Bermejo, and C. Juiz, "Multi-objective optimization for virtual machine allocation and replica placement in virtualized hadoop," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2568–2581, 2018.

[13] G. Fraser and A. Arcuri, "The seed is strong: Seeding strategies in search-based software testing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. IEEE, 2012, pp. 121–130.

[14] T. Shi, H. Ma, and G. Chen, "A genetic-based approach to location-aware cloud service brokering in multi-cloud environment," in *2019 IEEE International Conference on Services Computing (SCC)*. IEEE, 2019, pp. 146–153.

[15] H. Khalajzadeh, D. Yuan, J. Grundy, and Y. Yang, "Improving cloud-based online social network data placement and replication," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 678–685.

[16] L. Heilig, R. Buyya, and S. Voß, "Location-aware brokering for consumers in multi-cloud computing environments," *Journal of Network and Computer Applications*, vol. 95, pp. 79–93, 2017.

[17] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 3, pp. 689–702, 2015.

[18] F. Xie, J. Yan, and J. Shen, "A data dependency and access threshold based replication strategy for multi-cloud workflow applications," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 281–293.

[19] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, "Scheduling workflows with budget constraints," in *Integrated research in GRID computing*. Springer, 2007, pp. 189–202.

[20] L. Zeng, B. Veeravalli, and X. Li, "Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*. IEEE, 2012, pp. 534–541.

[21] T. Loukopoulos and I. Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *Journal of Parallel and Distributed Computing*, vol. 64, no. 11, pp. 1270–1285, 2004.

[22] H. Huang, H. Ma, and M. Zhang, "An enhanced genetic algorithm for web service location-allocation," in *International Conference on Database and Expert Systems Applications*. Springer, 2014, pp. 223–230.

[23] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Pias: Practical information-agnostic flow scheduling for commodity data centers," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 4, pp. 1954–1967, 2017.

[24] D. Trossell. (2016) How to reduce data and network latency where others fail. [Online]. Available: https://www.datacenterknowledge.com/archives/2016/11/03/reduce-data-network-latency-others-fail

[25] Skytap. (2019) Help and documentation. [Online]. Available: https://help.skytap.com/Accessing_VMs_with_Published_Services.html

[26] J. D. Little and S. C. Graves, "Little's law," in *Building intuition*. Springer, 2008, pp. 81–100.

[27] K.-C. Huang and B.-J. Shen, "Service deployment strategies for efficient execution of composite SaaS applications on cloud platform," *Journal of Systems and Software*, vol. 107, pp. 127–141, 2015.

[28] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*. IEEE, 2010, pp. 446–452.

[29] H. Arabnejad and J. G. Barbosa, "A budget constrained scheduling algorithm for workflow applications," *Journal of grid computing*, vol. 12, no. 4, pp. 665–679, 2014.

[30] K.-F. Man, K.-S. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.