1 GENETIC ALGORITHM

Genetic Algorithm (GA) has been successfully applied in solving container allocation problems. The solutions of GA are usually presented as string-like notations, i.e., chromosomes, to represent the allocation of containers on the PMs. The solution representation is a microservice array where the index is the microservice ID, and the value is a list, which contains the IDs of the physical machines (PMs). The same PM ID could not be repeated in the list as the containers of the same microservice can not be allocated in the same machine, i.e., the same PM ID can not be repeated in a service list. For example, in Figure 1, the parent 1 chromosome has four service types, and the list for each service is the PM ID to allocate containers, e.g., containers of service type s_1 are allocated in PMs (p_1, p_3, p_5, p_8).

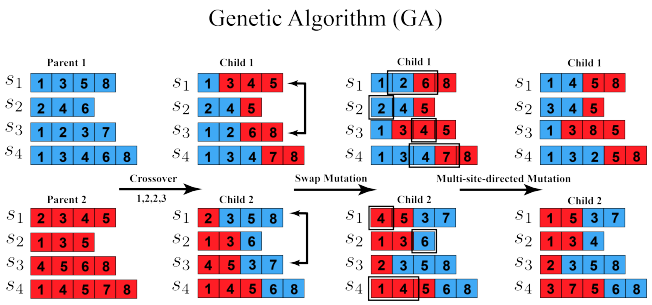

Figure 1: An example of Genetic Algorithm applied to a pair of parent chromosomes including crossover and mutation.

Similar to the objective function in the paper, we use the coefficient of variations of available resources are the fitness function, i.e., a smaller score means a better allocation solution. The reproduction of each generation follows two steps: (a) parents selection, (b) offspring generation. Algorithm 1 represents the overview of GA. After randomly generating a population of chromosomes, we adopt a strategy of selecting the best half of the chromosomes as the parents to generate children based on the fitness function (Line 2-3). The offspring generation contains three mutation operations, i.e., crossover, swap mutation, and multi-site-directed mutation. The crossover operator combines two parent chromosomes to generate two children by taking alternative pieces from these two parents and swap them (Line 5-8). Each crossover operator generates a random number that does not exceed the length of the related PM list, and the pieces to be swapped are picked after this crossover point. It is notable that for each service type, the number of PMs does not change. Therefore, the swap mutation (Line 10) only happens between PM lists with the same length (see swap mutation in Figure 1). Similarly, traditional growth and shrink mutation could be conducted since the length of the PM list is fixed. In this paper, we use multi-site-directed mutation (Line 12) to change multiple PM IDs randomly (see multi-site-directed mutation in Figure 1).

2 ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) simulates pheromone-based communication of ants to finding good paths to fetch food through graphs. In our scenario, each ant constructs a solution by starting to allocate

Algorithm 1: Genetic Algorithm (GA)

Input : A set of containers C^j of service type s_j chosen from microservice set $\mathcal{S}$ that are allocated on a set of machines $\mathcal{P}$
Output : A reallocation plan.
Init: Generate N random allocation plans and convert them to chromosomes.

1 **while** *no termination criterion is met* **do**
2 get score of each chromosome and sort them in ascending order by the score;
3 select the front half of these chromosomes as parents;
4 **for** *each pair of the selected parent chromosomes* **do**
5 randomly generate crossover points
6 **for** *each crossover point* **do**
7 swap genes of these two parent chromosomes after the cross point;
8 **end**
9 **for** *each parent chromosome* **do**
10 randomly pick two chromosome pieces, i.e., service array containing PM IDs, of the same length and conduct swap mutation;
11 randomly pick PM IDs and conduct multi-site-directed mutation;
12 **end**
13 **end**
14 merge generated children with previous selected front half parents to form a new population of chromosomes;
15 **end**
16 select the best child and convert it to the allocation plan.

containers to PM (Algorithm 2). The preference to select a PM to allocate a container is based on a pheromone-based heuristic function (Equation 1).

$$\eta = \frac{1}{CV^{mem} + \alpha CV^{cpu}} \tag{1}$$

The selection contains both exploration and exploitation. We sample a random number r between 0 and 1, if r is smaller than a threshold when we select PM with exploitation according to Equation 2.

$$i = \begin{cases} arg\ max_{i \in \mathcal{PM}}\{\tau_i^\alpha \eta_i^\beta\}, & r \le r_0 \\ P_i, & otherwise \end{cases} \tag{2}$$

Otherwise, we take the exploration strategy to select PM using roulette wheel according to Equation 3.

$$P_i = \begin{cases} \frac{\tau_i^\alpha \eta_i^\beta}{\sum_{i \in \mathcal{PM}} \tau_i^\alpha \eta_i^\beta}, & i \in \mathcal{PM} \\ 0, & otherwise \end{cases} \tag{3}$$

On the other hand, the PM IDs should meet the requirements that the allocated container should not exceed PM's available resources. Meanwhile, containers with the same type can not be allocated to the same machine (Equation 4).

Algorithm 2: Ant Colony Optimization (ACO)

Input : A set of containers C^j of service type s_j chosen from microservice set $\mathcal{S}$ that are allocated on a set of machines $\mathcal{P}$

Output : A reallocation plan.

Init: Initialize the pheromone on each PM.

1 **while** *no termination criterion is met* **do**
2 **for** *each ant in the ant population* **do**
3 **for** *each service type* **do**
4 get the heuristic probability of each PM;
5 **for** *each container of this service type* **do**
6 generate a random number r between 0 to 1;
7 **if** *r is less then selection probability threshold* **then**
8 select the machine with the largest heuristic value;
9 **else**
10 use roulette wheel selection to pick the machine for allocating the container;
11 **end**
12 update pheromone locally;
13 **end**
14 **end**
15 **end**
16 select the best ant based on objective function;
17 update pheromone globally;
18 **end**
19 select the best ant and convert its path to an allocation plan.

$$\mathcal{PM} = \{i | (\sum_{j=1}^{m} \sum_{k=1}^{K_j} \zeta^{res}(c_k^j) x_i^{kj} \leq r_i^{res}) \cup (\sum_{k=1}^{K_j} x_i^{kj} = 1)\}$$
$$\forall j = 1, \cdots, m, \quad x_i^{kj} \in \{0, 1\}, \quad res \in \{cpu, mem\} \tag{4}$$

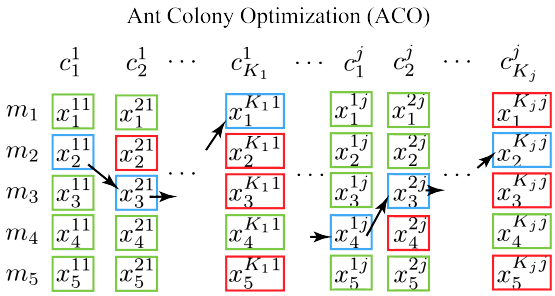

Figure 2: An example of the usage of Ant Colony Optimization on an allocation scenario. The blue box represents the machine is chosen to allocate the container. The green box is the candidate machines to be selected, and the red box means the machine can not allocate this container.

As shown in Figure 2, the ant selects the next machine to allocate containers from a list of PMs in the green box, meanwhile avoiding red PMs which do not enough resources or have the same container allocated previously.

Once finish allocation, each ant leaves a pheromone trail for the allocation plan, and the heuristic value of each machine changes accordingly. The amount of pheromone for each machine in the trail is determined by Equation 5, where L is the number of ants.

$$\Delta\tau = \frac{1}{C \sum_{l=0}^{L} \eta_l} \tag{5}$$

In the meanwhile, the pheromone over all machines will be updated both locally and globally by evaporating parts of the pheromone on them (Line 12 and Line 17). After a few iterations, a reallocation plan with the most pheromone is selected as the solution.