Optimizing Heterogeneous Task Allocation for Edge Compute Micro Clusters Using PSO Metaheuristic

Yousef Alhaizaey
School of Computing Science
University of Glasgow
United Kingdom
y.alhaizaey.1@research.gla.ac.uk

Jeremy Singer
School of Computing Science
University of Glasgow
United Kingdom
jeremy.singer@glasgow.ac.uk

Anna Lito Michala
School of Computing Science
University of Glasgow
United Kingdom
annalito.michala@glasgow.ac.uk

***Abstract*— Optimised task allocation is essential for efficient and effective edge computing; however, task allocation differs in edge systems compared to the powerful centralised cloud data centres, given the limited resource capacities in edge and the strict QoS requirements of many innovative Internet of Things (IoT) applications. This paper aims to optimise heterogeneous task allocation specifically for edge micro-cluster platforms. We extend our previous work on optimising task allocation for micro-clusters by presenting a linear-based model and propose a metaheuristic Particle Swarm Optimisation (PSO) technique to minimise the makespan time and the allocation overhead time of heterogeneous workloads in batch execution. We present a comparative performance evaluation of metaheuristic PSO, mixed-integer programming (MIP) and randomised allocation based on the computation overhead time and the quality of the solutions. Our results show a crossover implying that mixed-integer programming is efficient for small-scale clusters, whereas PSO scales better and provides near-optimal solutions for larger-scale micro-clusters.**

***Index Terms*—Edge Micro-Clusters, Edge Systems, Edge Computing, Task Allocation, Resource Management, PSO, Optimisation.**

I. Introduction

The recent growth in Internet of Things (IoT) technology requires the distribution of computing resource across the network edge, rather than offloading computation to remote cloud data centres. This is mainly to minimise latency, eliminate network jitter, and enhance privacy of IoT-relevant services [1].

This transformation from a centralised model to a distributed model is captured by the trend in edge computing or fog computing. Several architectures and solutions have been proposed to facilitate effective edge computations. However, the majority of work focuses on edge computing systems where resources are considered sufficient for computing workloads locally. For example, cloudlet data centres deploy resource-rich computers or clusters to provide computation and storage services for nearby mobile devices [2], [3].

In this work, we focus instead on edge compute micro-clusters where the computing resources are extremely constrained. Edge compute micro-clusters demonstrate realistic infrastructures that can be deployed near the network edge to support smart IoT, based on micro-scale clusters composed of several small computing nodes with sufficient collective capacity to perform required computation.

Recent studies construct single board computer (SBC) clusters to process edge computing applications. Clustering resource-limited devices represents an effective way to offer sufficient computer power to handle workloads at the network edge. Although existing literature demonstrates the feasibility of such edge micro-clusters, most studies merely consider simple application deployment. Furthermore, there is an absence of evaluating complex multi-task scenarios, orchestration and management, and the required optimisation of such micro-clusters. In this paper we argue that resource management aspects, such as task allocation or load balancing, should be integrated within edge micro-cluster functionality to optimise overall performance and make these micro-clusters more efficient.

Resource management in edge computing can be complex, given resource heterogeneity, varying resource capacities, workload variation, and strict QoS requirements of IoT-based applications [4]. Resource management generally covers several technical concepts including, but not limited to, task allocation, task offloading, load balancing and resource provisioning [5]. These management aspects address specific problems and require advanced optimisation techniques. In this paper we assert that the optimisation techniques deployed in centralised cloud data centres need to be re-evaluated for compact edge micro-clusters.

This paper expands on our previous work on optimising task allocation for edge micro-clusters [6], through proposing a linear-model objective function and evaluating metaheuristic Particle Swarm Optimisation (PSO) [7]. PSO is a well-known optimisation technique used in many scientific areas, especially in cloud data centres to solve various resource management problems [8]. Our current study aims to evaluate the performance of different optimisation techniques in terms of *solution quality* and *allocation overhead*, particularly for edge micro-clusters. Although the No Free Lunch Theorem (NFLT) states that no optimisation technique can be most effective for all optimisation problems [2], [9], our evaluation reflects typical performance of selected optimisation techniques for micro-clusters and provide insights for edge computing researchers. Our results show that mathematical optimisation-

based allocation is appropriate for small-scale micro-clusters, whereas search-based metaheuristic allocation is a more viable technique for large-scale clusters. Accordingly, this paper seeks to address the following research questions:

1) *RQ1:* Which task allocation methods provide effective solutions that can optimise makespan time for edge micro-clusters in relation to heterogeneous workloads in batch modes?
2) *RQ2:* Which optimisation methods are efficient and appropriate (i.e., sufficiently lightweight) for edge micro-clusters?

This paper makes the following contributions. (1) The characterisation of micro-clusters nodes performance for processing concurrent workloads. (2) The contribution of a linear model based on empirical experiments to optimise the makespan time required to process heterogeneous workloads for edge micro-clusters. (3) The comparison of allocation overhead time and solution quality of different optimisation techniques for task allocation for edge micro-clusters. (4) The identification of a crossover point that refers to the efficiency of task allocation optimisation techniques for edge micro-clusters. (5) An experimental evaluation of different optimisation techniques for optimising task allocation for edge micro-clusters.

The remainder of this paper is organised as follows. Section II outlines the systems specification and the task allocation problem formulations. Section III presents the adoption of PSO to optimise task allocation pertaining to the problems associated with edge micro-clusters. Section IV evaluates the results emerging from this study. Section V reviews related research into edge computing resource management and edge-based clusters. The paper ends with Section VI, in which the conclusion and suggestions for future research are presented.

II. Background

This section briefly discusses edge micro-cluster systems and presents a linear model developed specifically for modelling tasks allocation.

A. Edge Micro-Clusters and Workloads

1) ***Edge Micro-Clusters****:* Edge micro-clusters represent compact edge platforms that form stationary or portable solutions deployed near data sources. Such micro-clusters might connect to the same local area network (LAN) of IoT appliances and operate using renewable energy or battery. Edge micro-clusters can provide sufficient computing resources to other tiny IoT devices that are not equipped with processing and storage resources, which mandates offloading their workloads to edge or cloud data centres.

Incrementing to the micro-cluster setup configured in our earlier work [6], this micro-cluster testbed features several heterogeneous nodes to model the system. The current version of our micro-cluster comprises eight Raspberry Pi devices drawn from different generations to model the system's heterogeneity. The cluster nodes are mounted in a mini rack case and connected using an ethernet switch. Table I presents nodes

TABLE I
TYPICAL NODES DEPLOYED IN EDGE MICRO-CLUSTERS

Node Type	Cores	CPU(GHz)	RAM(GB)	OS	Qty
RPi (2B)	4	0.9	1	RaspberryPi OS	3
RPi (3B)	4	1.2	1	RaspberryPi OS	3
RPi (3B+)	4	1.4	1	RaspberryPi OS	1
RPi (4B)	4	1.5	4	RaspberryPi OS	1

TABLE II
EXEMPLAR HARDWARE SPECIFICATION OF TYPICAL NODES DEPLOYED IN EDGE-MICRO CLUSTERS AND IN CENTRALISED CLOUD DATA CENTRES

Machine	CPU	Cores	Memory	OS	Virtualisation Technology
RPi 3B	ARM Cortex-A72 with 1.2GHz	4	1GB	RaspberryPi OS	Containers (e.g.,Docker)
RPi 4B	ARM Cortex-A72 with 1.5GHz	4	4GB	RaspberryPi OS	Containers (e.g.,Docker)
Amazon EC2	Intel core i7 with 3.2 GHz	2	8GiB	Linux	Hypervisor (e.g.,Xen)

specifications used in the configured micro-cluster testbed. The testbed configuration overall represents micro-cluster heterogeneity in terms of resources like CPU and memory. Such edge micro-clusters platforms might include other equivalent SBC devices with different hardware specifications.

In addition, edge micro-cluster systems can be configured to receive and process workloads using lightweight deployment technologies like Docker containers, which can instantly initialise application instances without extra delay to the system. Conventional virtualisation solutions, such as hypervisor virtual machines, might not be compatible with micro-cluster devices as they require larger resource overhead in terms of CPU or RAM.

Micro-clusters can perform heterogeneous edge-relevant workloads, offloaded from surrounding IoT appliances, such as environmental sensors or drones. The anticipated workloads might include workloads like image processing and audio processing tasks. The subsequent section describes workloads in more detail.

2) ***Workloads****:* In terms of workloads, we assume that edge micro-cluster platforms receive and process heterogeneous tasks in batch-execution mode. Different IoT devices generate workloads and choose to offload to micro-clusters through any networking medium for processing and storage. Micro-clusters aggregate tasks into batches to be effectively allocated on cluster nodes.

Micro-clusters consist of several physical nodes, and each node can host a limited number of Docker containers of heterogeneous tasks. We consider that workloads are heterogeneous with respect to task types as different applications generate and offload certain kinds of tasks. For example, image-detection applications capture images and offload workloads of different sizes to an edge micro-cluster for processing. Batch execution consumes expensive parallel processing; thus, effective optimisation techniques are required. Section II-B presents task allocation for micro-clusters in further detail.

B. System Objective Function

1) ***Individual Node Performance****:* The hardware and software specifications of nodes in micro-clusters differ from nodes deployed in centralised cloud data centres. The nodes in

micro-clusters are typically integrated with small to moderate computing resources in terms of CPU, memory, and storage, whereas cloud data centres deploy high-performance server machines. Table II presents exemplar hardware specification of nodes in edge micro-clusters and nodes in cloud data centres.

To develop effective task allocation for micro-clusters, nodes performance need to be characterised at first. Below we present pre-experiments to evaluate the performance of nodes used in our micro-cluster when processing edge representative tasks. The main objective of these experiments is to capture performance characteristics of nodes when processing concurrent realistic workloads. This is to direct us to model a more realistic system objective function customised to micro-cluster platforms. Existing work characterises similar edge devices using micro-benchmarks like *Sysbench* and *Linpack* [10]. However, micro-benchmarks might not be sufficient to capture meaningful performance of micro-cluster nodes [11]. We used real-world holistic benchmarks to represent a balanced mixture of micro-behaviours for edge applications and capture relevant node performance metrics.

In particular, we examined the performance of different Raspberry Pi devices when processing concurrent tasks. We benchmark devices drawn from different generations that represent typical resource heterogeneity in edge micro-clusters, when processing concurrent and heterogeneous workloads from different applications, image-detection (Yolo), audio-text converting (PocketSphinx), and audio-text synchronisation (Aeneas). These applications are drawn from DeFog benchmark suite [12]. In addition, we modified and extended the DeFog benchmark to enable us to send multiple tasks concurrently.

Results are presented in Figure 1. The evaluation illustrates that nodes performance are linearly affected by the number of concurrent tasks. Results show linear growth in processing times for all applications benchmarks. Moreover, the experiments reveals nodes capabilities limits for handling concurrent tasks. For example, Raspberry Pi devices RPi 2B, RPi 3B, and RPi 3B+ were not able to accommodate multiple tasks that require intensive-processing (i.e., image detection). We noticed task failures when sending more than four concurrent tasks for image-processing workload. Furthermore, those nodes were not responding for more than eight concurrent tasks. Analysing task failures is beyond the scope of this paper and is subject to future work. Subsection II-B2 explains a linear-model system objective function for task allocation for edge micro-clusters.

*2) **Linear Model Objective Function**:* As mentioned earlier, we assume that micro-cluster processes workloads in a batch-execution mode, where tasks are offloaded from various IoT applications and aggregated into batches. The challenge is to effectively allocate tasks to micro-cluster nodes such that we minimise the makespan time of the entire batch and respect the cluster resources capacities.

The objective is to effectively allocate tasks to the cluster nodes in order to minimise the total execution time for a full batch of tasks (i.e. the *makespan* time) by effectively allocate tasks to nodes. Equation 1 describes the system

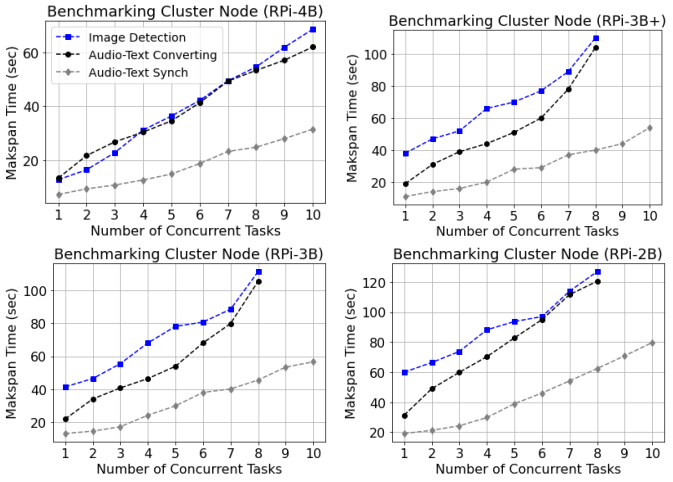

Fig. 1. Line graphs showing the performance trends of micro-cluster nodes for processing concurrent workloads.

objective function we developed based on results from the nodes performance experiments presented in section II-B1. Equation 1 evaluates candidate solutions provided by allocation techniques, where T is the makespan time, m_n is a constant that represents the gradient of node n, $numTask_n$ is the total number of tasks assigned to node n, and c_n is a constant that represents the y-intercept of node n. As the cluster processes tasks in parallel execution, T is effectively the maximum execution time among all the cluster nodes. Table III presents linear interpolation values of the gradients and the y-intercept derived for different applications.

$$T = \max_{n \in N} \left(m_n * numTask_n + c_n \right) \tag{1}$$

where:

T: denotes the Makespan Time
m_n: is a constant value representing the gradient of node n
$numTask_n$: is the total number of tasks allocated to node n
c_n: is a constant value representing y-intercept of node n

Similar to our earlier work [6], two constraints are introduced to model the micro-cluster heterogeneity and nodes capacities. First, we defined node capacity constraint (Equation 2), in which each node n has a defined capacity and should not accept more than its defined capacity. cap_n is a non-negative integer value that represents node n capacity, x_{nt} is a binary array of 0-1 indicating whether task t is allocated to node n. N and T represent a set of cluster nodes and a set of tasks, respectively. In addition, we defined a further constraint (Equation 3) to make sure that each task t is assigned to only one node n and avoid assigning the same task to multiple nodes.

TABLE III
CALCULATED VALUES FOR GRADIENT (M) AND Y-INTERCEPT (C)

Workload	Image Detection		Audio-Text Converting		Audio-Text Synch	
	m-value	c-value	m-value	c-value	m-value	c-value
RPi 2B	10	50	15	20	5	15
RPi 3B	10	30	8	15	5	5
RPi 3B+	10	25	8	15	5	5
RPi 4B	5	5	5	5	3	5

$$\left(\sum_{t\in T} x_{nt}\right) <= cap_n \quad , \quad \forall \quad n \in N \tag{2}$$

$$\left(\sum_{n\in N} x_{nt}\right) = 1 \quad , \quad \forall \quad t \in T \tag{3}$$

III. TASK ALLOCATION USING PARTICLE SWARM OPTIMISATION

Particle Swarm Optimisation is a well-known metaheuristic optimisation technique inspired by the social collaboration of bird swarms. In PSO, various particles are generated to discover the search space. The PSO iteratively optimises candidate solutions for several number of iterations or upon meeting a pre-defined termination condition and gradually converges towards near-optimal solutions [7].

The velocity (Equation 4) and position (Equation 5) control particles in the search process. Velocity V_i^{t+1} computes the distance between the current position X_i^t and the new position X_i^{t+1} using information from current iteration and hyper-parameters values, defined in next paragraph. This prompts particles to re-locate in the direction of the best position. Particle best positions $PBest_i$ and global best positions of the swarm $GBest$ are frequently updated and compared during the search process to maintain the best found position.

PSO hyperparameters direct particles in their search for a global optimum. The inertia weight parameter W is used to balance exploration and exploitation, where a high W value increases exploration and a small value increases exploitation. Equation 6 balances exploration and exploitation by assigning a large W value at the beginning of the search process and gradually decreases it throughout the search process. $WMax$ and $WMin$ are constants representing upper and lower bounds, respectively, t is the current iteration, and $MaxIter$ is the total number of iterations. Equation 6 allows particles to discover the search space and then converge towards the best-found solution. A typically W value ranges from 0.4 to 0.9. In our experiments, we set $W = 0.79$ because we noticed that high value led to premature convergence whereas small value directed the particles to local optimal solutions.

The inertia weight W, the acceleration coefficients $c1$ and $c2$, and the random variables $r1$ and $r2$ help controlling the velocity of the particles. Coefficient components $c1$ and $c2$ also impact exploitation and exploration in PSO. The individual component $c1$ is primarily responsible for increasing exploration, while the coefficient social component $c2$ is used to increase the exploitation. Balancing between $c1$ and $c2$ helps PSO particles to converge toward near-optimal solutions. The random variables are used to add randomness to the particles in each iteration. Table IV summarises the hyperparameters used in PSO and the assigned values employed in our implementation.

$$\begin{aligned} V_i^{t+1} = W * V_i^t + c1 * r1 * (PBest_i - X_i^t) \\ +c2 * r2 * (GBest - X_i^t) \end{aligned} \tag{4}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{5}$$

$$W = WMax - t((WMax - WMin)/MaxIter) \tag{6}$$

In our PSO task allocation representation process, PSO particles represent candidate allocation solutions. A PSO particle is effectively a function that maps tasks to cluster nodes. We mapped the $length$ of particles in relation to the total number of tasks to be allocated and mapped the $range$ of particles to the total number of nodes in a micro-cluster. For example, suppose we have a batch of 10 tasks to be allocated on an 8-nodes cluster. In our representation, the $length$ of particles represents number of tasks, i.e., 10 and the $range$ of particles is randomly set between 0 to 7, where 0 represents node 1 and 7 represents node 8 as our edge micro-cluster contains eight nodes.

Table V presents illustration examples of different particles and the allocation representation. For example, in solution 1, the particle is represented as array [7, 7, 7, 4, 7, 5, 7, 3, 7, 6]. In our representation, tasks $t1$, $t2$, $t3$, $t5$, $t7$, $t9$ are allocated to node $n7$; task $t4$ is allocated to node $n4$, task $t6$ is allocated to node $n5$, and etc. Fitness function (Equation 1) evaluates fitness of particles using values derived from table III. Algorithm 1 represents our PSO-based task allocation.

Overall, PSO iteratively discovers the solution space during the search process. The particles gradually converge toward the best-found solution by updating their best positions using the velocity and position. PSO demonstrates fast convergence and is able to avoid local optima. Furthermore, PSO is easy to implement without requiring the complexity of mathematical optimisation, which renders PSO a promising lightweight optimisation technique for edge computing deployment. The following section evaluates the performance of PSO optimisation for edge micro-cluster platforms.

IV. PERFORMANCE EVALUATION

This section evaluates the performance of the task allocation techniques implemented in this work and analyses the results. The performance of the PSO-based technique is compared with that of the mixed-integer programming technique and with a random-based technique. Simple greedy techniques have been evaluated in our earlier work [6].

The experiments were carried out using both modelling and empirical evaluations. Specifically, the allocation techniques were implemented via modelling using a MacBook Pro machine (2.4 GHz dual-core i5, 4GB RAM). We empirically

Algorithm 1: PSO-based Task Allocation

```
Initialise Particles X_i^t Randomly;
Initialise Velocity V_i^t Randomly ;
Initialise hyperparameters using table IV;
for t in Iteration do
    for i in Particles do
        Compute fitness using (Equation 1);
        Compute New Velocity V_i^{t+1} using (Equation 4);
        Compute New Position X_i^{t+1} using (Equation 5);
        #Update Best Position (PBest)
        if fitness < PBest_i then
            PBest_i=fitness
            X_i=PBest_i
        end
        # Update Global Position (GBest)
        if fitness < GBest then
            GBest=fitness
            X_i=GBest
        end
    end
end
return Best Allocation Decision
```

TABLE IV
PSO HYPERPARAMETERS VALUES IMPLEMENTED IN THIS WORK

Parameters	Description	Assigned Value
W	inertia weight	0.79
c1	coefficient for individual component	1.49
c2	coefficient for social component	1.49
r1	random coefficient	0-1
r2	random coefficient	0-1
Particles	Number of particles in each iteration	50
Iteration	Number of generations	100

conducted experiments using a physical micro-cluster testbed to evaluate the obtained solutions. The allocation techniques effectively should be executed on a cluster node capable of performing the orchestration and management aspects; however, due to dependencies requirements, we implemented the optimisation techniques via modelling on a MacBook Pro machine. The tasks are then allocated to cluster nodes according to allocation decisions.

We noted that the majority of work in the literature follows simulation-based experiments using simulation tools such as CloudSim or MATLAB. We used an empirical evaluation approach using a physical micro-cluster testbed to capture more realistic performance. PSO-based optimisation and the random-based techniques were implemented in Python, while the mixed-integer programming-based method was implemented using Google open source software suite for optimisation OR-Tools [13].

TABLE V
EXEMPLAR WORKLOAD MAPPING ON EIGHT-NODE MICRO-CLUSTER WITH ESTIMATED MAKESPAN TIME

Workloads	$t1$	$t2$	$t3$	$t4$	$t5$	$t6$	$t7$	$t8$	$t9$	$t10$	**Est Makespan**
Image-Detection	$n7$	$n7$	$n7$	$n4$	$n7$	$n5$	$n7$	$n3$	$n7$	$n6$	40 sec
Audio-Text Synch	$n6$	$n7$	$n7$	$n3$	$n3$	$n7$	$n6$	$n5$	$n4$	$n4$	15 sec
Audio-Text Convert	$n6$	$n7$	$n7$	$n4$	$n5$	$n7$	$n3$	$n7$	$n4$	$n3$	31 sec
Mixed Workloads	$n7$	$n0$	$n4$	$n7$	$n3$	$n5$	$n7$	$n3$	$n6$	$n7$	25 sec

A. Optimisation Techniques

1) ***Mixed-Integer Programming Optimisation****:* Without loss of generality, the mixed-integer programming technique optimises the fitness function and allocates workloads to cluster nodes according to the optimal found allocation. Similar to PSO-based technique, it estimates each batch makespan time using values derived from Table III and takes into account the system constraints (see Equation 2 and Equation 3. We implemented this method using Google open source software suite for optimisation, OR-Tools [13].

2) ***Randomised Allocation****:* The random-based allocation method is a straightforward baseline allocation technique where workloads are randomly distributed to cluster nodes without knowledge about nodes capabilities or workloads types.

B. Evaluation Metrics

In this research, we evaluated the proposed allocation technique for edge micro-clusters based on two edge-relevant performance metrics. First is the ***allocation overhead time***, which is the time required by an allocation technique to complete the required computation and produce an allocation decision. The second metric is ***makespan time***, which reflects the quality of the solutions provided by the allocation techniques.

The makespan time is defined as the total time required for the micro-cluster testbed to process workloads in batch execution mode. The makespan time includes the time required by the allocation techniques to generate an allocation decision, the time required for offloading tasks to cluster nodes, the time required for the nodes to process the allocated workloads, and finally, the time required to receive results back from cluster nodes. As the micro-cluster processes tasks in parallel execution, the makespan time effectively is the maximum execution time required by the cluster node that takes the maximum time to complete its allocated workloads.

Table V presents examples of allocation solutions for allocating batches of 10 tasks and the estimated makespan based on the linear-model objective function. As per example, we explain the first solution for image-detection. According to the allocation decision, tasks $t1$, $t2$, $t3$, $t5$, $t7$, and $t9$ are allocated to $n7$; $t4$ is allocated to $n4$; $t6$ is allocated to $n5$; $t8$ is allocated to $n3$; and finally, $t10$ is allocated to $n6$. All tasks will be processed in parallel in the cluster and require an estimated makespan time of 40 sec, which is the time required by either $n3$, $n4$ or $n5$. These nodes are RPi3B and the estimated $MakespanTime$ = 40 sec, translated as: m = 10; $numTask$ = 1; and c = 30 ($10 * 1 + 30 = 40$ sec).

C. Results Evaluation

1) ***Accuracy of the Linear Model****:* This subsection evaluates the linear model objective function. We assessed the

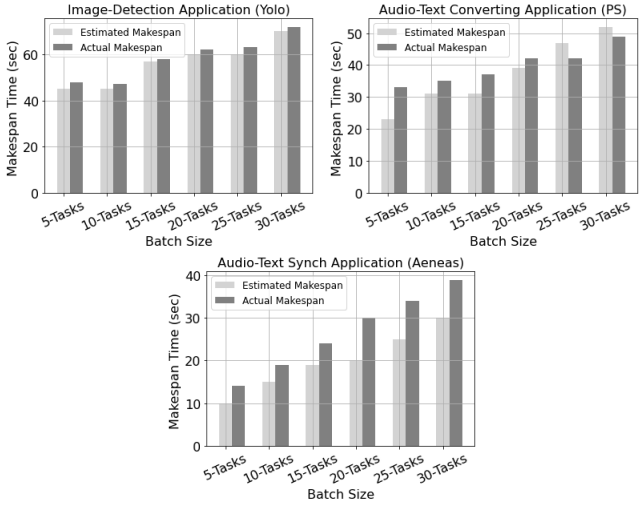

Fig. 2. Comparison between the estimated makespan time computed by the linear-based model and the actual makespan time running on the edge micro cluster.

accuracy of the proposed linear model using three application benchmarks by comparing the estimated makespan time generated by the linear model and the actual makespan time on an eight node micro-cluster testbed. Figure. 2 shows the results for different applications, where the x-axis represents different batch sizes, and the y-axis represents makespan time in sec. Overall, the results demonstrate that the linear model is useful for estimating the actual makespan time required for batch execution for edge micro-cluster. The data for the gradient and y-intercept values used for these experiments are presented in Table III.

*2) **Allocation Overhead Time**:* The allocation overhead time represents the runtime an allocation technique requires to search the solution space and converge to produce a task allocation decision. Allocation overhead time is a critical metric for edge computing because orchestration techniques run on resource-constrained devices, such as Raspberry Pi devices [14]. In relation to edge micro-clusters, we anticipate that the allocation technique will be deployed on one cluster node, thereby representing a cluster head node.

We consider two different scenarios to evaluate the performance of the optimisation techniques specifically for edge micro-clusters, as shown in Figure 3. For the first scenario, tasks are all the same types, representing homogeneous workloads to be effectively allocated to the cluster. In this scenario, the tasks are classified into different batches according to task types. For the second scenario, batches comprise heterogeneous workloads, where tasks are aggregated in mixed batches collected from different applications.

The PSO-based overhead time was compared to the mixed-integer programming allocation overhead time. Figure 3 shows practical crossover graphs identifying conventional overlapping between different task allocation optimisation techniques for edge micro-clusters. The allocation overhead time for the random-based technique never exceeds a few msec and was

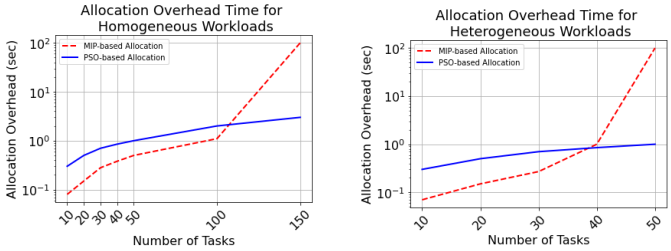

Fig. 3. Crossover graphs representing the allocation overhead time required by the exponential complexity MIP-based and linear complexity PSO-based for different batch sizes and workload types. The Random-based allocation is excluded from this comparison as its overhead time is always minimal and never exceeds a few msec.

excluded from the figures.

The results demonstrate that the mixed-integer allocation technique is lightweight for homogeneous batches, as it converges very quickly and obtains the optimal allocation solution in less than 1 sec for up to 100 homogeneous tasks. However, the overhead time grows exponentially and dominates the total execution time for large homogeneous batches. Moreover, the problem becomes more complex for the heterogeneous workloads given the increased number of constraints, which makes exponentially complex MIP-based allocation an unfeasible technique for heterogeneous workloads.

On the other hand, PSO-based technique proves linear-complexity growth for both scenarios and achieves near-optimal solutions quality for homogeneous and heterogeneous batches, as we will explain in the following subsection. The PSO-based technique shows faster computation time than the mixed integer programming method and effectively produces near-optimal solutions for large-scale batches.

*3) **Makespan Time and Solution Quality**:* The task allocation techniques in this study are designed to minimise the makespan time metric, defined as the total execution time required for the edge micro-cluster to complete processing batches of workloads. This objective is achieved by identifying an appropriate allocation solution that permits effective distribution of tasks across cluster nodes.

We evaluated the performance of the task allocation techniques to find the most effective allocating decision that minimises the makespan time of different batch sizes. We assessed the performance of the allocation techniques with regard to different batch sizes and workloads types, i.e., homogeneous batches or heterogeneous batches.

Figure 4 illustrates the quality of solutions of the allocation techniques, namely: mixed integer-based allocation, PSO-based allocation, and random-based allocation. We examined for different batches of homogeneous and heterogeneous workloads to be allocated on an 8-node heterogeneous micro-cluster. For homogeneous workloads, results demonstrate that mixed-integer programming-based and PSO-based techniques can achieve comparable solutions quality, and their allocation overhead times do not significantly impact the total makespan time (see Figure 3). These findings overall prove that our

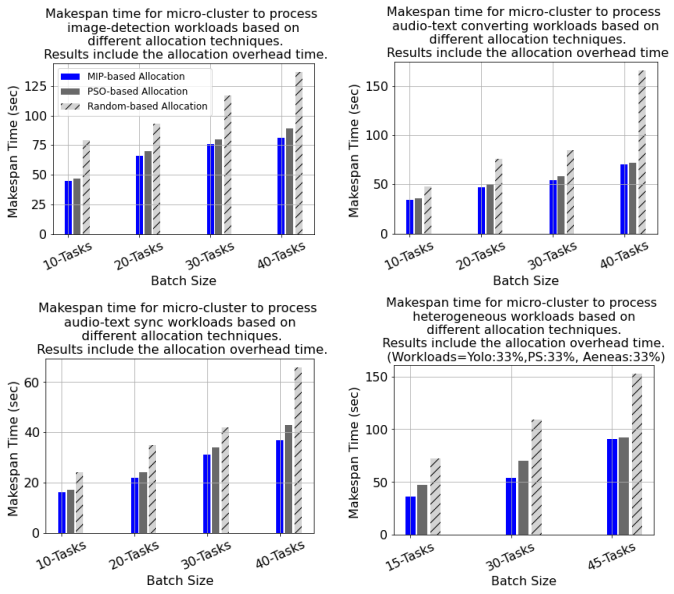

Fig. 4. Quality of solutions obtained by MIP-based allocation, PSO-based allocation, and Randomised-based allocation for different batches before MIP reaches the 100 task threshold.

PSO-based technique is effective in producing near-optimal solutions compared with optimal solutions obtained by MIP-based technique.

For the heterogeneous batches, we examined three different heterogeneous batches where workloads are aggregated from different applications. The workload descriptions of batches is as follows. The image-detection workloads constitute 33%, audio-text synchronisation workloads constitute 33%, and audio-text converting workloads constitute 33%. For this scenario, results show that the allocation overhead time is growing exponentially for MIP-based allocation, as shown in Figure 3, and significantly influences the overall makespan time for large batch size (i.e., batch size 45 tasks). We observe that MIP-based technique takes 18 sec to produce the optimal allocation decision. The allocation overhead time of the PSO-based does not impact the makespan time as it grows linearly in relative to the batch sizes. Overall, the PSO-based technique demonstrates a viable and efficient technique that maintains both near-optimal allocation decision and minimal overhead time.

Finally, the results show that the random-based allocation is impractical for edge micro-cluster systems because the makespan time is always high for both homogeneous and heterogeneous batches; although the allocation overhead time remains constant and never exceeds a few msec. Furthermore, we noticed task execution failures when nodes exceeded their capacities. The analysis and the measurements of avoiding task failures are beyond the scope of this paper. This might involve techniques for load balancing or task migration between the cluster nodes.

*4) **Discussion**:* We conclude that there might be a necessary balance between obtaining high-quality solutions and securing much faster ones. It might be necessary to compromise between solution quality and computational complexity in order obtain faster allocation decisions. This applies when the allocation overhead to find high-quality allocation solutions exceeds the expected total computation time of any batch, particularly for large-scale clusters and large batch sizes.

*5) **Threats to Generality**:* We consider the generalizability of our findings. First, the devices deployed in the current version of our edge micro-cluster testbed were limited to Raspberry Pi devices. The characteristics of Raspberry Pi devices like cost, power consumption and advanced hardware specifications make Raspberry Pi devices a compelling selection for deployment in edge micro-cluster platforms. However, such micro-cluster platforms technically might involve other comparable SBC devices like Beaglebone or Odroid. Secondly, the cluster size is arguably smaller than the real-world scenarios. Yet, our linear-based model can be generalised easily to evaluate for larger clusters. Expanding the cluster size and involving other SBC types are planned for future work.

V. Related Work

Edge micro-cluster platforms comprise several heterogeneous but resource-constrained computing nodes aggregated over a local area network and coordinated by a controller node [2] [15]. These micro-clusters demonstrate compact solutions for practical edge computing. Recent studies established the need for such micro-clusters and show the feasibility and suitability of using single board computers (SBCs), such as Raspberry Pi and Odroid devices, to build edge micro-clusters that can host edge-related applications [15].

Qureshi et al. [16] presented a SBC-based cluster for smart parking management system in context of smart city. Specifically, they constructed a small edge cluster using different SBC devices to monitor parking plots and analyse daily data. The work serves to motivate that such small SBC-based clusters are effective solutions for edge use cases. Rausch et al. [17] further investigated the energy characteristics of portable edge clusters and highlighted the usability of such clusters for portable and fieldwork edge use cases. The authors built an edge cluster using Raspberry Pi devices and general-purpose hardware devices. Sagkriotis et al. [18] investigated energy usage for SBC-clusters and developed energy monitoring application for profiling nodes energy consumption while hosting Docker containers. Nevertheless, further work is still required to optimise different resource management aspects for such computing clusters. Existing work only utilised micro-clusters for basic applications deployment without integrating the complexity of multi-task scenarios, orchestration, and different management aspects.

In edge computing, task allocation generally represents techniques applied to produce an optimal or near-optimal mapping of IoT-related workloads to edge devices. The overall objective is to generate effective allocation decisions capable of meeting different QoS requirements. This typically involves minimising edge-related metrics, such as makespan time, power consumption, and latency. Task allocation is a complex

optimisation problem in edge computing due to resource heterogeneity and the distribution of edge infrastructures. Researchers have proposed task allocation solutions based on various optimisation approaches. Solutions can be generally classified into three categories: mathematical-based solutions, heuristic-based solutions, and metaheuristic-based solutions [2] [5]. Skarlat et al. [19] provided a genetic algorithm-based solution to optimise fog service placement in fog computing. They compared the greedy first fit, exact optimisation, and cloud solutions through simulation methods. Taneja et al. [20] developed a heuristic-based algorithm to deploy applications in fog-cloud paradigm. The proposed solution iterates from fog nodes toward cloud nodes, thereby placing application modules on the most eligible fog nodes. If no fog nodes are available, contact is made with cloud nodes. Azimi et al. [21] compared two metaheuristic optimisation algorithms, namely: Particle Swarm Optimisation (PSO) and BAT algorithm to optimise multi-clustering management. Gill et al. [22] proposed a ROUTER resource scheduling management based on PSO technique for different QoS metrics for smart homes, including energy consumption, latency, bandwidth, and response times. They implemented the proposed solution using iFogSim and compared it with two algorithms, namely: round robin and first come first serve scheduling. Compared to [22], this work evaluates PSO-based task allocation designed to optimise batch execution for micro-clusters composed of several resource-limited devices and investigates the effectiveness and the efficiency for the proposed techniques. In comparison to existing research, this paper seeks to evaluate different task allocation techniques for optimising batch execution for edge micro-cluster platforms.

VI. Conclusion

This paper presents a comparative evaluation of different optimisation techniques for optimising heterogeneous workloads in batch execution for micro-cluster platforms. The performance of PSO-based allocation technique was compared with that of mixed-integer programming allocation and with the randomised-based allocation techniques. Our evaluation demonstrates a crossover point, which provides typical indicative limits for the application of each optimisation technique for edge micro-clusters. Our results indicate that PSO and MIP both achieve comparable solutions for small-scale edge clusters. However, PSO scales better and finds effective solutions for large-scale micro-clusters, with linear computation overheads compared to the exponential MIP for large-scale clusters. Based on our evaluation, we argue that mathematical-based optimisation is only recommended for task allocation for small-scale clusters, while PSO-based optimisation demonstrates a lightweight and effective optimisation approach for large-scale edge clusters. For future directions, we intend to extend this work by optimising other edge-related QoS metrics, like energy consumption and cost, to address complex multi-objective optimisation problems. In addition, we aim to evaluate other metaheuristic optimisation techniques, such as genetic algorithms.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[2] S. O. Ogundoyin and I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," *Swarm and Evolutionary Computation*, vol. 66, p. 100937, 2021.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[4] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–37, 2019.

[5] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: a comprehensive review," *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, 2020.

[6] Y. Alhaizaey, J. Singer, and A. L. Michala, "Optimizing task allocation for edge micro-clusters in smart cities," in *22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2021, pp. 341–347.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[8] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simulation Modelling Practice and Theory*, p. 102353, 2021.

[9] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[10] R. Morabito, "Virtualization on internet of things edge devices with container technologies: A performance evaluation," *IEEE Access*, vol. 5, pp. 8835–8850, 2017.

[11] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[12] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "Defog: fog computing benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 47–58.

[13] L. Perron and V. Furnon, "OR-tools," Google, 2022, v9.3. [Online]. Available: https://developers.google.com/optimization/

[14] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet of Things*, p. 100273, 2020.

[15] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018.

[16] B. Qureshi, K. Kawlaq, A. Koubaa, B. Saeed, and M. Younis, "A commodity SBC-edge cluster for smart cities," in *2019 2nd International Conference on Computer Applications & Information Security*. IEEE, 2019, pp. 1–6.

[17] T. Rausch, C. Avasalcai, and S. Dustdar, "Portable energy-aware cluster-based edge computers," in *2018 IEEE/ACM Symposium on Edge Computing*. IEEE, 2018, pp. 260–272.

[18] S. Sagkriotis, C. Anagnostopoulos, and D. P. Pezaros, "Energy usage profiling for virtualized single board computer clusters," in *2019 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–6.

[19] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.

[20] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management*, 2017, pp. 1222–1228.

[21] S. Azimi, C. Pahl, and M. H. Shirvani, "Particle swarm optimization for performance management in multi-cluster iot edge architectures." in *CLOSER*, 2020, pp. 328–337.

[22] S. S. Gill, P. Garraghan, and R. Buyya, "Router: Fog enabled cloud based intelligent resource management approach for smart home IoT devices," *Journal of Systems and Software*, vol. 154, pp. 125–138, 2019.