Accepted Manuscript

Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory

Mahbuba Afrin, Jiong Jin, Ashfaqur Rahman, Yu-Chu Tian, Ambarish Kulkarni

PII: S0167-739X(18)32678-5
DOI: https://doi.org/10.1016/j.future.2019.02.062
Reference: FUTURE 4815

To appear in: *Future Generation Computer Systems*

Received date : 30 October 2018
Revised date : 24 January 2019
Accepted date : 22 February 2019

Please cite this article as: M. Afrin, J. Jin, A. Rahman et al., Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.02.062

Multi-Objective Resource Allocation for Edge Cloud based Robotic Workflow in Smart Factory

Mahbuba Afrin[1], Jiong Jin[1], Ashfaqur Rahman[2], Yu-Chu Tian[3], Ambarish Kulkarni[4]

Abstract

Multi-robotic services are widely used to enhance the efficiency of Industry 4.0 applications including emergency management in smart factory. The workflow of these robotic services consists of data hungry, delay sensitive and compute intensive tasks. Generally, robots are not enriched in computational power and storage capabilities. It is thus beneficial to leverage the available Cloud resources to complement robots for executing robotic workflows. When multiple robots and Cloud instances work in a collaborative manner, optimal resource allocation for the tasks of a robotic workflow becomes a challenging problem. The diverse energy consumption rate of both robot and Cloud instances, and the cost of executing robotic workflow in such a distributed manner further intensify the resource allocation problem. Since the tasks are inter-dependent, inconvenience in data exchange between local robots and remote Cloud also degrade the service quality. Therefore, in this paper, we address simultaneous optimization of makespan, energy consumption and cost while allocating resources for the tasks of a robotic workflow. As a use case, we consider resource allocation for the robotic workflow of emergency management service in smart factory. We design an Edge Cloud based multi-robot system to overcome the limitations of remote Cloud based system in exchanging delay sensitive data. The resource allocation for robotic workflow is modelled as a constrained multi-objective optimization problem and it is solved through a multi-objective evolutionary approach, namely, NSGA-II algorithm. We have redesigned the NSGA-II algorithm by defining a new chromosome structure, pre-sorted initial population and mutation operator. It is further augmented by selecting the minimum distant solution from the non-dominated front to the origin while crossing over the chromosomes. The experimental results based on synthetic workload demonstrate that our augmented NSGA-II algorithm outperforms the state-of-the-art works by at least 18% in optimizing makespan, energy and cost attributes on various scenarios.

Keywords: Resource Allocation; Multi-Robot System; Edge Cloud; Workflow Management; Smart Factory; Multi-Objective Evolutionary Algorithm.

1. Introduction

Robotic services are used in various real-world scenarios including fire-fighting [1], search and rescue [2]. These services follow individual workflow consisting of data hungry, latency sensitive and compute intensive tasks. In multi-robot systems, a group of robot, connected through wired/wireless communication, work collaboratively to execute these tasks. However, processing of tasks in such system subjects to several challenges. The robots are resource and energy constrained. Due to mobility, it is difficult to maintain a persistent communication among them. Lack of storage and memory also causes disruptions in exchanging and storing large scale data during robot-robot interaction. To overcome these limitations, the concept of Cloud Computing has been extended to multi-robot systems [3] [4]. In Cloud-enabled multi-robot systems, Cloud offers virtualized instances, platform and software services so that both local and remote resources can be utilized to process the tasks of robotic workflow [5] [6].

Conceptually, Cloud-enabled multi-robot systems promote Industry 4.0 applications such as smart factory [7], smart farm [8] and smart retail [9], where Internet of Things, Cloud and robotic technologies are amalgamated [10] [11]. In a smart factory, Cloud-robotic solutions can manage business processes including entire production and supply chain along with logistic support [12]. Safety, health and environmental regulations is one of the major concerns in smart factory. For example, safety assurance in smart factory during hazardous situation like fire occurrence is very crucial. In this case, both robot and Cloud resources should complement each other to process diversified tasks of emergency management service within a stringent deadline. These tasks are usually inter-dependent, latency sensitive and compute intensive. In addition, the resources are heterogeneous in terms of processing capability and energy consumption. Moreover, such system structure incurs supplementary cost while executing robotic tasks in a distributed manner over the local and virtual instances. Therefore, an optimal allocation of computing resources to the tasks with a view to minimizing service delay, energy consumption of resources and cost

[1]School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia.
[2]Data61, CSIRO, Sandy Bay, TAS 7005, Australia.
[3]School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD 4001, Australia.
[4]School of Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia.
Email: mafrin@swin.edu.au

of task execution is one of the key research challenges in managing robotic workflow for smart factory.

In this paper, we focus on optimal robot and Cloud-based resource allocation for the tasks of emergency management robotic service such as fire driven emergency management workflow in a smart factory. Tasks of this workflow require real-time response and data transmission among robots and Cloud, although higher inter-communication latency of resources resists the whole system to meet these requirements. To address the limitations of Cloud enabled multi-robot systems, the concept of Edge Cloud [13] infrastructure is introduced between the robot and Cloud. Basically, Edge Computing is an extension of the Cloud Computing paradigm providing data, compute, storage and application services to end-users on a so-called Edge layer [14]. It thus assists multi-robot systems by executing the latency sensitive and compute intensive services at closer proximity of the working environment. According to the proposed system model, the Edge Cloud is aware of the combined resource pool composed of both local and virtual resources to facilitate resource allocation for the tasks. However, the total cost of task execution over distributed resources and their uneven energy consumption during task execution significantly affect the system performance. In this context, computational resource allocation appears as a constrained multi-objective optimization problem when energy consumption of resources, overall makespan and total monetary cost for task execution are targeted to minimize simultaneously. Multi-objective evolutionary algorithms help to generate pareto-optimal solutions for such multi-objective optimization problems. Therefore, in this paper, we extend the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [15] to solve the resource allocation problem for its capability of finding diverse set of solutions. We augment the NSGA-II by defining a new chromosome structure, pre-sorted initial population based on the task size and processing speed of the resources. Besides, while crossing over the chromosomes, rather than selecting arbitrarily, the chromosome having minimum distant solution from the pareto-front to the origin is selected to balance the values of all objectives in subsequent generations. The results of our simulation experiments significantly improve existing multi-objective optimization approaches including benchmark NSGA-II, Multi-Objective Particle Swarm Optimization (MOPSO) [16], Strength Pareto Evolutionary Algorithm II (SPEA2) [17] and Pareto Archived Evolution Strategy (PAES) [18] in terms of meeting the stopping criteria, satisfying data dependency threshold, and minimizing the total energy consumption, the makespan and the total system cost for varying number of tasks and resources. To the best of our knowledge, this is the first work to design an Edge Cloud based multi-robot system optimizing three objectives concurrently. Moreover, performance comparison among different multi-objective evolutionary algorithms focusing on such system has been investigated in this work which has not been explored in the literature previously. The main contributions of this paper are listed as follows:

- An Edge Cloud based multi-robot framework is designed for emergency management robotic service in smart factory to overcome the limitations of remote Cloud based framework in executing latency sensitive tasks.
- The computational resource allocation problem for the tasks in a robotic workflow is modelled as a multi-objective optimization problem to simultaneously minimize the makespan of completing all tasks, the energy consumption of resources and the total cost of task execution.
- A multi-objective evolutionary algorithm NSGA-II is redesigned to solve the optimization problem with inclusion of pre-sorted initial population and minimum distant selection of chromosome that gives balanced solution for all objectives. Each pareto solution obtained in our approach provides an optimal mapping of tasks on computational resources.
- The experimental results support our adopted methodology. The performance study shows significant improvements in terms of meeting the stopping criteria, satisfying data dependency threshold, minimizing energy consumption, makespan and the total cost compared with state-of-the-art multi-objective optimization approaches.

The rest of the paper is organized as follows. In Section 2, relevant research works are reviewed. Edge Cloud based framework for executing robotic workflow in smart factory with motivating use case is described in Section 3. The problem formulation and our proposed multi-objective resource allocation mechanism are provided in Section 4. In Section 5, the efficacy of our proposed approach is validated through simulation. Finally, Section 6 concludes the paper with future directions.

2. Related Work

In literature, the integration of Cloud services and multi-robot systems has already been investigated. For instance, to virtualize execution of networked robotic services, Cloud infrastructure is introduced by Agostinho et al. [27] and Mouradian et al. [28]. Chen et al. [29] also provides the concept of Robot as a Service (RaaS) through a Service Oriented Architecture (SOA) that incorporates robot services in Cloud.

There exist some works concentrating task offloading for the execution of robotic services. For optimal task offloading Rahman et al. [24] deisgn a smart city based Cloud robotic framework for minimizing time and energy consumption of resources. However, in this work, a single robot is solely responsible for offloading decision making. Motion and connectivity-aware offloading for Cloud robotic services using evolutionary algorithm is further introduced by Rahman et al. [25]. In this work, multiple robots are responsible for offloading, however, they do not consider the cost of executing taks on Cloud resources. Later on, Rahman et al. [26] study on communication-aware and mobility-driven offloading for smart factory maintenance, where they only consider the energy consumption of resources. Considering limited robot to Cloud access, Wang et

Table 1: Summary of relevant works

Work	Solution Approach	Target Application	Objective Parameters			Resource Type			
			Energy	QoS	Cost	Network Resource	Local Robots	Cloud Resource	Edge Cloud Resource
Wang et al. [19]	Auction-based	Real-time applications		✓		✓			
Wang et al. [20]	Stackelberg game-based	Co-localization		✓		✓			
Liu et al. [21]	Reinforcement learning-based	Planning and maintenance	✓					✓	
Wu et al. [22]	Gini coefficient and market-based	Hazardous Situation	✓				✓		
Mouradian et al.[23]	Node and network level robots virtualization	Search and rescue		✓			✓		
Rahman et al. [24]	Genetic Algorithm-based	Smart City	✓	✓			✓		
Rahman et al. [25]	Genetic Algorithm-based	Smart City	✓	✓			✓	✓	
Rahman et al. [26]	Genetic Algorithm-based	Smart Factory	✓				✓	✓	
Multi-Objective Resource Allocation (Our Work)	Multi-Objective Evolutionary Strategy-based	Smart Factory	✓	✓	✓		✓	✓	✓

al. [19] present an auction-based bandwidth allocation mechanism for the robots. They only allocate the network resource, computational resource allocation strategies are not addressed here. Again, Wang et al. [20], discuss a Stackelberg game based resource allocation strategy where, the tasks are assigned according to bandwidth allocation cost. However, the uncertainty of the resource prices, energy consumption of resources is not investigated while allocating resources for robotic tasks.

Different resource and task allocation policies have also been studied for Cloud-based multi-robot systems. Liu et al. [21] discuss a resource allocation policy for Cloud using reinforcement learning to handle the robotic service requests, although their policy does not address the combined provisioning of both local and Cloud resources. Task allocation in resource-constrained multi-robot systems for search and rescue or other emergency and hazardous scenarios is discussed by Wu et al. [22]. Cost-efficient deployment of robots for a search and rescue use case during large-scale disaster management is explored by Mouradian et al. [23]. However, in the aforementioned works, no multi-robot task allocation policy targeting both local and Cloud resources is pursued.

To solve the multi-objective resource allocation problem in Cloud-enabled multi-robot system, different heuristic, market based, timed automata, swarm intelligence and linear programming model are used. Unfortunately, these approaches are not always adaptable to decentralized structures and are less supportive to deal with limited access of robots to Cloud due to real-world environmental constraints. For example, Zheng et al. [30] adopt linear programming model to monitor distributed robotic planning. However, this model requires high computational cost and further optimization to solve the problem at run-time. Nevertheless, considering the intrinsic constraints of Cloud and multi-robot assisted smart factory during the execution of emergency management services, neither the combined resource (local and virtual) allocation problem nor the solutions has been exploited in the literature so far. Table 1 provides a summary of existing relevant resource allocation policies in Cloud-enabled multi-robot system.

Therefore, in this paper, we consider the resource allocation problem for environment driven emergency management service in a smart factory. The concept of Edge Cloud is introduced in multi-robot system to address the communication latency constraint. We consider co-optimization of makespan of the tasks, energy consumption of the resources and monetary cost for task execution concurrently while allocating resources for the tasks of a robotic workflow. The resource allocation is modelled as a multi-objective optimization problem and a multi-objective evolutionary algorithm, NSGA-II, is augmented to minimize all the objective parameters simultaneously. In literature, basic NSGA-II algorithm is already recommended for multi-objective resource allocation in Mobile Cloud Computing to minimize time and energy by Ghasemi-Falavarjani et al. [31]. However, in this paper, a new chromosome structure is defined and a pre-sorted initial population is generated based on the task size and processing speed of the resources. We also extend the NSGA-II by selecting the minimum distant solution from the pareto-front to the origin during crossing over for balanced solution. The experimental results signify the improved performance in favor of our approach compared to other multi-objective optimization approaches.

3. Framework for Edge Cloud based Multi-Robot System

3.1. System Model and Assumptions

In this paper, Edge Cloud infrastructure between robot and Cloud forms a three-tier computational framework and helps to execute emergency management service for smart factory with less communication latency. In this Edge Cloud based system, the robots using Zigbee, Bluetooth and WiFi are able to communicate with each other [3] and collect the environmental data for service execution from its on board sensors. They also maintain communication with Edge Cloud and Cloud through gossip protocols [32]. Since the robots and Edge Cloud are localized in a smart factory, we assume that their mobility does not affect the robot-to-robot and robot-to-Edge Cloud communication during task execution.

In an Edge Cloud, the virtualized resources are managed by a *Resource Manager (RM)*. The RM also perceives the context of local computational resources residing in the robots. Moreover, it creates a combined resource pool with both local and remote resources while executing any robotic service as shown in Fig. 1. The RM is responsible to allocate computational resources efficiently for the tasks associated with a robotic workflow.

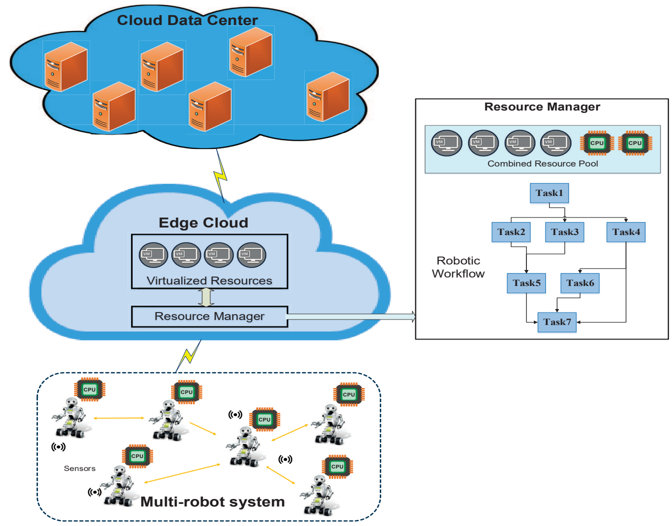


Figure 1: Resource Allocation for Edge Cloud based Robotic Workflow in Smart Factory

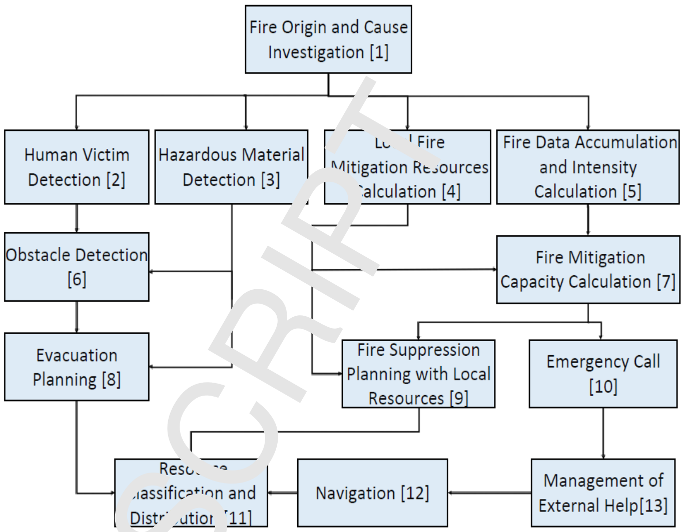


Figure 2: Workflow of fire emergency management service in a smart factory

While assigning tasks to the computational resources, it is required to consider data dependency of the tasks, communication latency among the entities (robots and Edge Cloud) and resource performance (energy consumption, processing time, total cost) simultaneously. Here, we assume that the corresponding tasks of a service and the meta-data of the tasks (inter-data dependency delay, Quality of Service requirement etc.) are pre-stored in the Edge Cloud. Whenever an event of interest triggers the initiation of the service, the RM residing in the Edge Cloud assigns the tasks to the combined resource pool.

3.2. Application Use Case: Fire Emergency Management Service in a Smart Factory

Through recent efforts of Internet of Things and Cloud robotics, Industry 4.0 applications can be run with limited human involvement, increased efficiency and reduced cost [33]. Based on this observation, smart factories with challenges of Health, Safety and Environment (HSE) are set as our motivation for robotic Inspection, Maintenance and Repair (IMR). In this scenario, a pool of heterogeneous sensors is deployed throughout smart factory to perceive environmental conditions. Cloud-aided robots can complement the sensing operation with action-oriented task such as inspection, fault diagnosis etc. All smart factory based applications require robots to continuously update intensive data in order to execute tasks in a coordinated manner [7] [33].

In this paper, we consider emergency fire management service in smart factory as an illustrative use case scenario. This kind of service requires timely execution to ensure the environmental safety within smart factory. This service consists of multiple tasks such as fire origin and cause identification, human victim and hazardous material detection, evacuation planning, navigation as well as management of external help. These tasks are interdependent and orchestrated through a Directed Acyclic Graph (DAG) based workflow model. An example workflow for fire emergency management service in smart factory is shown in Fig. 2. The number of tasks in such DAG based workflow can vary application to application. Here, our research problem is to allocate computational resources optimally for the tasks of such workflow to minimize the makespan for executing all the tasks, the total energy consumption of resources and the total cost require for executing the tasks on the computational resources. Relevant notations and definitions used in system model and problem formulation are represented in Table 2.

4. Multi-objective Resource Allocation Strategy for Edge Cloud Robotic Workflow

4.1. Multi-objective Optimization Problem Formulation

The optimal resource allocation for the tasks of robotic workflow subjects to multiple constraints driven by their characteristics such as latency sensitivity, interdependency and resource requirements. It turns into a multi-objective optimization problem when overall makespan for completing the tasks, energy consumption of the resources and the total monetary cost for executing the tasks are simultaneous minimized.

In Edge Cloud-based multi-robot system, the total time m_r^t requires to complete a task $t \in T$ on a resource $r \in R$, is calculated using the processing speed ρ_r of the resource and the size of task λ_t as shown in Eq. 1.

$$m_r^t = \frac{\lambda_t}{\rho_r} \tag{1}$$

Similarly, the energy consumption e_r^t for executing task $t \in T$ on resource $r \in R$ is calculated in terms of the per unit time (sec) energy consumption $\hat{e}_r$ of that resource and the required time m_r^t to execute the task as presented in Eq. 2.

$$e_r^t = \hat{e}_r \times m_r^t \tag{2}$$

Table 2: Notations

Symbol	Definition
T	Set of all tasks in robotic workflow.
R	Set of all computational resources in the combined resource pool.
$x_{tr} \in \{0,1\}$	Equals to 1 if the task in $t \in T$ is assigned to any computing resource $r \in R$, 0 otherwise.
m_r^t	Time requires to complete a task $t \in T$ on a resource $r \in R$.
e_r^t	The energy consumption of executing a task $t \in T$ on a resource $r \in R$.
υ_r^t	Monetary cost for executing a task $t \in T$ on particular resource $r \in R$.
λ_t	The size of a task $t \in T$.
ρ_r	Processing speed of a resource $r \in R$.
$\hat{e}_r$	The per unit time (sec) energy consumption of resource $r \in R$.
$\hat{\upsilon}_r$	The cost requires by the resource $r \in R$ in per unit time (sec).
M_T	Maximum allowable delay to complete all the tasks of a robotic application workflow.
E_T	Maximum allowable total energy consumption of resources for execution of the workflow.
Υ_T	Total budget for execution of the workflow.
T_t'	Set of tasks on which task $t \in T$ depends.
$p_{r'}^{t'}$	Maximum processing time of the predecessor tasks of a task $t \in T$ executing on resource $r' \in R$.
$\eta_{r',r}$	The time requires to send data from computing resource, $r' \in R$ to $r \in R$.
δ_t	Tolerable inter-task data dependency delay between a dependent task $t \in T$ and a predecessor task $t' \in T_t'$.

Besides, the monetary cost υ_r^t for executing task $t \in T$ on resource $r \in R$ depends on the per unit time (sec) cost $\hat{\upsilon}_r$ of the resource and the required time m_r^t to execute the task as shown in Eq. 3.

$$\upsilon_r^t = \hat{\upsilon}_r \times m_r^t \tag{3}$$

$$min(M(T,R), E(T,R), \Upsilon(T,R)) \tag{4}$$

where

$$M(T,R) = \sum_{t \in T, r \in R} x_{tr} \times m_r^t \tag{5}$$

$$E(T,R) = \sum_{t \in T, r \in R} x_{tr} \times e_r^t \tag{6}$$

$$\Upsilon(T,R) = \sum_{t \in T, r \in R} x_{tr} \times \upsilon_r^t \tag{7}$$

Based on Eq. 1-3, the multi-objective optimization problem for proposed Edge Cloud based framework is expressed in Eq. 4. For a set of tasks T of robotic workflow and a set of available resources R, Eq. 4 simultaneously minimizes the total makespan $M(T,R)$, the total energy consumption $E(T,R)$ and the total cost $\Upsilon(T,R)$. However, Eq. 4 is subject to the following constraints:

- Assignment constraint: The solution of Eq. 4 refers the assignment of task $t \in T$ to resource $r \in R$ through non-zero value of a binary decision variable x_{tr}. The assignment constraint in this case ensures that a task $t \in T$ will get exclusive access to it's assigned resource (Eq. 8).

$$\sum_{t \in T, r \in R} x_{tr} = 1 \tag{8}$$

- QoS Constraint: The total task completion time will not exceed the maximum allowable delay $\hat{M}_T$ to execute the robotic workflow (Eq. 9)

$$\sum_{t \in T, r \in R} m_r^t \leq \hat{M}_T \tag{9}$$

- Energy Constraint: The total energy consumption of resources should be within the energy threshold $\hat{E}_T$ of the system (Eq. 10).

$$\sum_{t \in T, r \in R} e_r^t \leq \hat{E}_T \tag{10}$$

- Budget Constraint: The total cost for executing all tasks will not surpass the budget $\hat{\Upsilon}_T$ of system operator (Eq. 11).

$$\sum_{t \in T, r \in R} \upsilon_r^t \leq \hat{\Upsilon}_T \tag{11}$$

- Data Dependency Constraint: Eq. 12 signifies that the assignment of a dependent task $t \in T$ to a resource $r \in R$ and its tolerable inter-task data dependency delay δ_t will not be affected by the the assignment of all its predecessor tasks $t' \in T_t'$ on computing resources $r' \in R$. In this case, maximum processing time $p_{r'}^{t'}$ of the predecessor tasks and data exchange time $\eta_{r',r}, \forall t' \in T_t'$ is calculated.

$$\max(p_{r'}^{t'} + \eta_{r',r}) \leq \delta_t; \forall t' \in T_t' \tag{12}$$

A single solution for this multi-objective optimization problem is infeasible to optimize each objective separately. For such problem, a finite number of non-dominated or pareto-optimal solutions are generated [15]. A set of solutions S dominates another set S' if each solution $x \in S$ is no worse than solution $x' \in S', \forall x'$ in every objective and each solution $x \in S$ is strictly better than solution $x' \in S', \forall x'$ in at least one objective. If a solution set is not dominated by any other set, the elements of that set is called the non-dominated or pareto-optimal solutions [34]. Since all pareto-optimal solutions are considered equally acceptable, the goal of such constrained multi-objective optimization problem is to find a representative set of pareto-optimal solutions. From this set of pareto-optimal solutions, service provider selects a particular solution that satisfy the subjective preference of the system. Solving this problem using any optimizer tool like NIMBUS usually takes significant amount of time [13]. Therefore, to generate pareto-optimal solutions within a stringent time frame, multi-objective evolutionary algorithm is applied in this paper.

4.2. *Energy-Delay-Cost Co-optimization Using Multi-objective Evolutionary Algorithm*

Multi-objective evolutionary algorithms are widely adopted to solve multi-objective optimization problems as they optimize multiple objectives simultaneously. There exist several multi-objective evolutionary algorithms such as Non-dominated Sorting Genetic Algorithm (NSGA-II) [15], Strength Pareto Evolutionary Algorithm II (SPEA2) [17], Pareto Archived Evolution Strategy (PAES) [18] and bio-inspired multi-objective optimization strategy such as Multi-Objective Particle Swarm Optimization (MOPSO) [16]. However, NSGA-II performs better in terms of finding a diverse set of solutions and in converging to near the true pareto-optimal set compared with others [15].

In NSGA-II algorithm, the solution of a chromosome contains the values of different objectives obtained from fitness functions, and the solutions of a population are classified into different sets according to the ascending level of their domination. Each set of solutions represents a particular front on the solution space and the chromosomes generating those solutions are treated as the builder of the front. Moreover, the non-dominated set of solutions provides the optimal front (first front) on the solution space, termed as the pareto-front. In this paper, we extend the basic concept of NSGA-II algorithm and further refine it to develop an energy, delay and cost co-optimized resource allocation for emergency management service in Edge Cloud infrastructure. The detail on how NSGA-II has been redesigned in our paper is discussed below.

Population initialization. In the augmented NSGA-II, rather than creating randomized initial population, a pre-sorted initial population relying on the heuristics is generated. Here, the set of resources R in the combined resource pool is divided into k categories based on the ascending processing speed $\rho_r, \forall r \in R$. Similarly, the set of tasks T is also classified in k types according to the incremental size $\lambda_t, \forall t \in T$. Thereafter, taking the problem range and constraints (Eq. 8-12) into account, mathematical combination is used to conceptually assign the tasks of j type to the resources of j^{th} category for generating the chromosomes of initial population. The structure of chromosomes aligned with the steps of initial population creation is represented in Fig. 3, where a Gene Index symbolizes a particular task and the Gene refers to a specific resource.

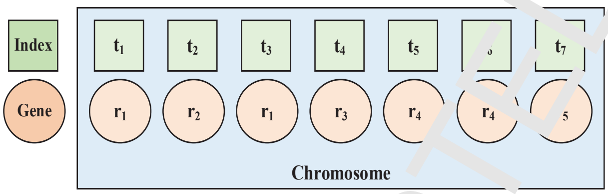


Figure 3: Chromosome structure for the evolutionary algorithm

The aforementioned operation not only sorts tasks and resources on the basis of their characteristics, but also implicitly promotes the assignment of larger tasks to computationally enriched resources and comparatively small sized tasks to computationally constrained resources. Since, there exists linear relation between the processing speed and the energy consumption of the resources, as well as between the processing speed and cost of the resources, solutions of the initial population eventually maintain the task processing time, energy consumption and cost of the resources within the lower bound. For maximum G number of generations, the initial population is noted as parent population P_1 for the first generation and the corresponding child population C_1 is generated by applying genetic operators on P_1. To conduct different operations of evolutionary algorithm on the first generation populations, both P_1 and C_1 are merged together in a combined population U_1.

Domination count and ranking. The solution space of a population is determined by the outcome of its member chromosome's fitness value on makespan, energy and cost objectives using Eq. 5-7. On the combined population U_i of any generation g_i; $i \in \{1...G\}$, Algorithm 1 performs domination count of the solutions generated by its member chromosomes and ranks them in different fronts within corresponding solution space based on the level of domination.

Algorithm 1 Algorithm for domination counting and front ranking

```
1:  procedure SolutionsSorting(U_i)
2:      F_1 ← ∅
3:      for c := U_i do
4:          Ω_c ← ∅
5:          ω_c ← 0
6:          for c′ := U_i do
7:              if c.M < c′.M & c.E ≤ c′.E & c.ϒ ≤ c′.ϒ || c.M ≤ c′.M & c.E < c′.E & c.ϒ ≤ c′.ϒ || c.M ≤ c′.M & c.E ≤ c′.E & c.ϒ < c′.ϒ then
8:                  Ω_c ← Ω_c ∪ c′
9:              else if c′.M < c.M & c′.E ≤ c.E & c′.ϒ ≤ c.ϒ || c′.M ≤ c.M & c′.E < c.E & c′.ϒ ≤ c.ϒ || c′.M ≤ c.M & c′.E ≤ c.E & c′.ϒ < c.ϒ then
10:                 ω_c ← ω_c + 1
11:         if ω_c = 0 then
12:             F_1 ← F_1 ∪ c′
13:     τ ← 1
14:     while F_τ ≠ ∅ do
15:         F_{τ+1} ← ∅
16:         for c := F_τ do
17:             for c″ := Ω_c do
18:                 ω_{c″} ← ω_{c″} − 1
19:                 if ω_{c″} = 0 then
20:                     F_{τ+1} ← F_{τ+1} ∪ c″
21:         τ ← τ + 1
```

In Algorithm 1, the *SolutionsSorting* procedure consists of two parts and takes the combined population U_i of generation g_i as an argument. During the first part, the procedure initializes the builder set of chromosomes for the pareto-front F_1 (line 2). For a chromosome $c \in U_i$, a dominates list Ω_c is initialized that contains the chromosomes whose solutions are dominated by the solution of c (line 4). A dominated counter ω_c is also initialized to get the number of chromosomes whose solutions dominate the solution of c (line 5). If the makespan objective value $c'.M$, the energy consumption objective value $c'.E$ and the cost objective value $c'.\Upsilon$ of any individual $c' \in U_i$ is dominated by the objective values of c, c' is added to Ω_c. Conversely, if the objective values of c' dominate the objective values of c, then ω_c is incremented with 1 (line 6-10). After exploring all the chromosomes, if ω_c is found in its initial value, the solution of c is considered non-dominated and c is added to the builder chromosome set of pareto-front $F1$. The first part of the procedure is executed for each chromosome of U_i.

In the second part of the procedure, each front is taken into account to identify builder chromosome set for the next front. Initial value of the front counter τ (line 13) helps to start this operation with builder chromosome set of the pareto-front. The value of τ also symbolizes the rank of corresponding front. However, the builder chromosome set of the front $F_{\tau+1}$, next to the current considering front F_τ is initialized with an empty set (line 14-15). The dominates list Ω_c of each builder chromosome $c \in F_\tau$ is explored and the dominated counter $\omega_{c''}$ of each chromosome $c'' \in \Omega_c$ is decremented by 1 (line 16-18). In performing this operation, the chromosome whose dominated counter becomes 0 is added to the builder chromosome set $F_{\tau+1}$ of the next front (line 19-20). After studying all the builder chromosomes of F_τ, τ is incremented by 1 so that $F_{\tau+1}$ of this round can be considered as the F_τ for the following round (line 21). The iteration of such rounds continues until the F_τ is empty.

Selection of population. Since the size of combined population U_i for any generation g_i is $2N$, for iterative refinement, it is very important to select the best N number of chromosomes from U_i to form the parent population P_{i+1} for the next generation g_{i+1}. Algorithm 2 represents a procedure named *PopulationSelection* that helps to select parent population for the next generation from builder chromosome set of the fronts identified through *SolutionsSorting* procedure. The procedure receives the set of all f number of fronts as argument.

At first, the procedure initializes parent population P_{i+1} for the next generation, a size counter κ to track how many chromosomes have been selected for P_{i+1} and a front counter τ (line 2-4). The fronts are traversed in ascending order of their rank and their builder chromosomes are added to the P_{i+1} (line 5-8). For a particular front F_τ, if addition of its builder chromosomes to P_{i+1} exceeds the population size N, *CrowdingDistance* procedure is called for that front. The called procedure selects a set of chromosomes ϕ from the builder set of the front to be added with P_{i+1}. The cardinality χ of ϕ is determined by the difference of population size N and size counter κ of P_{i+1} (line (9-12)).

Algorithm 2 Algorithm for selecting parent population for next generation

1: **procedure** PopulationSelection($\{F_1, F_2, F_3, ..., F_f\}$)
2: $\quad P_{i+1} \leftarrow \emptyset$
3: $\quad \kappa \leftarrow 0$
4: $\quad \tau \leftarrow 1$
5: $\quad$ **while** $\kappa + |F_\tau| \leq N$ **do**
6: $\qquad P_{i+1} \leftarrow P_{i+1} \cup F_\tau$
7: $\qquad \kappa \leftarrow \kappa + |F_\tau|$
8: $\qquad \tau \leftarrow \tau + 1$
9: $\quad$ **if** $\kappa < N$ **then**
10: $\qquad \chi \leftarrow N - \kappa$
11: $\qquad \phi \leftarrow$ *CrowdingDistance*(F_τ, χ)
12: $\qquad P_{i+1} \leftarrow P_{i+1} \cup \phi$

Crowding distance calculation. In selecting N number of chromosomes for the parent population P_{i+1} of next generation g_{i+1}, sometimes the available slot in P_{i+1} can be less enough to accommodate the entire builder chromosome set F_τ of a particular front. In this case, crowding distance of the solutions are calculated to identify the compatible builder chromosomes of that front to fill the available slot in P_{i+1}. Algorithm 3 represents the procedure *CrowdingDistance* to calculate the crowding distance of solutions for a particular front F_τ and select χ number of compatible chromosomes for P_{i+1}.

The procedure at first identifies the cardinality μ of F_τ and initializes the crowding distance $c.D, \forall c \in F_\tau$ to 0 (line 2-4). The chromosomes of F_τ is sorted in ascending order of their makespan objective value M (line 5). Based on this sorted F_τ, crowding distance of the chromosomes having maximum and minimum makespan objective value is set to infinity (line 6-7). Then for the rest of the chromosomes, the procedure calculates the normalized difference between the makespan objective values of next and previous individuals and adds to the crowding distance of the considering chromosome (line 8-9). The similar operations are also conducted on the chromosomes of F_τ for the energy objective value E (line 10-14) and the cost objective value Υ (line 15-19) accordingly. After completing the aforementioned steps for all the objectives, the set of selected chromosomes ϕ is initialized and the chromosomes of F_τ are sorted in descending order of their crowding distance D. Thereafter, the first χ number of chromosomes from F_τ are added to ϕ. Then, ϕ is returned to the calling procedure for adding to the P_{i+1}.

Algorithm 3 Algorithm for calculating crowding distance

1: **procedure** CrowdingDistance(F_τ, χ)
2: $\quad \mu \leftarrow |F_\tau|$
3: $\quad$ **for** $c := F_\tau$ **do**
4: $\qquad c.D \leftarrow 0$
5: $\quad$ *SortAscending*(F_τ, M)
6: $\quad F_\tau[1].D \leftarrow \infty$
7: $\quad F_\tau[\mu].D \leftarrow \infty$
8: $\quad$ **for** $i = 2......\mu - 1$ **do**
9: $\qquad F_\tau[i].D \leftarrow F_\tau[i].D + \frac{F_\tau[i+1].M - F_\tau[i-1].M}{F_\tau[\mu].M - F_\tau[1].M}$
10: $\quad$ *SortAscending*(F_τ, E)
11: $\quad F_\tau[1].D \leftarrow \infty$
12: $\quad F_\tau[\mu].D \leftarrow \infty$
13: $\quad$ **for** $i = 2......\mu - 1$ **do**
14: $\qquad F_\tau[i].D \leftarrow F_\tau[i].D + \frac{F_\tau[i+1].E - F_\tau[i-1].E}{F_\tau[\mu].E - F_\tau[1].E}$
15: $\quad$ *SortAscending*(F_τ, Υ)
16: $\quad F_\tau[1].D \leftarrow \infty$
17: $\quad F_\tau[\mu].D \leftarrow \infty$
18: $\quad$ **for** $i = 2......\mu - 1$ **do**
19: $\qquad F_\tau[i].D \leftarrow F_\tau[i].D + \frac{F_\tau[i+1].\Upsilon - F_\tau[i-1].\Upsilon}{F_\tau[\mu].\Upsilon - F_\tau[1].\Upsilon}$
20: $\quad \phi \leftarrow \emptyset$
21: $\quad$ *SortDescending*(F_τ, D)
22: $\quad$ **for** $i = 1.....\chi$ **do**
23: $\qquad \phi \leftarrow \phi \cup F_\tau[i]$
24: $\quad$ return ϕ

Extension of genetic operator. For a particular generation g_i, while generating the child population C_i from the parent population P_i, genetic operators such as fitness calculation, selection, crossover and mutation are applied. The fitness of the population is determined through the objective functions discussed in Eq. 1-3. In addition, to imply mutation on the population, binomial distribution [35] and to make crossover of a particular chromosome with the fittest chromosome of the population, simulated binary [36] approaches are used. To select the fittest chromosome from the population, *FitChromosomeSelection* procedure in Algorithm 4 is applied. The procedure takes the pareto-front builder chromosome set $F_1^{P_i}$ of parent population P_i as the argument. After initializing necessary variables (line 2-3), for each chromosome $c \in F_1^{P_i}$, the distance $c.d$ of makespan, energy consumption and cost objective value $(c.M, c.E, c.\Upsilon)$ from the theoretical lowest value of makespan, energy and cost $(0, 0, 0)$ in three dimensional space is calculated (line 4-5). The chromosome having minimum distance value from the origin is selected as the fittest chromosome of the population for making crossover (line 6-9). In minimizing all objectives, since the pareto-front of a population reflects concave up, decreasing trend in the solution space, the resultant chromosome of this procedure provides a balanced solution for all objectives. Therefore, it is feasible to spread the characteristics (tasks to resource assignment) of the resultant chromosome to others for further refinement of multi-objective optimization in the following generations.

Algorithm 4 Algorithm for selecting fittest chromosome for crossover

```
1: procedure FitChromosomeSelection(F_1^{P_i})
2:     d_min ← ∞
3:     c_fit ← null
4:     for c := F_1^{P_i} do
5:         c.d ← √(c.M² + c.E² + c.ϒ²)
6:         if c.d < d_min then
7:             c_fit ← c
8:             d_min ← c.d
9:     return c_fit
```

Moreover, after running this extended NSGA-II algorithm with pre-sorted population for G number of generations, the solution set for the optimization problem might point towards multiple chromosomes. In this case, *FitChromosomeSelection* will also be used for selecting the best chromosome for task to resource assignment. However, the selection of final solution and characteristics of the corresponding chromosome solely depend on the intention of service providers. They are able to define the expected percentage of time, energy and cost from the ultimate solution space, and select the solution and chromosome accordingly.

5. Simulation Results and Discussion

In this section, the performance of our proposed multi-objective resource allocation approach is compared with some of the existing multi-objective optimization strategies in the literature. Benchmark NSGA-II algorithm as adopted by Ghasemi-Falavarjani et al. [31] is implemented for the comparison with our proposed augmented NSGA-II approach. The benchmark strategy follows NSGA-II algorithm with randomized initial population and arbitrary chromosome selection during crossover. Conversely, our proposed augmented NSGA-II operates on pre-sorted initial population and selects a chromosome for crossover that has the minimum distant solution at the pareto-front from the origin for a particular generation. Furthermore, we compare the performance of our augmented NSGA-II with benchmark Multi-Objective Particle Swarm Optimization (MOPSO) [16], Strength Pareto Evolutionary Algorithm II (SPEA2) [17] and the Pareto Archived Evolution Strategy (PAES) [18] algorithms. All the strategies are simulated in Matlab. We also analyze the impact of Edge Cloud on the dependent tasks of a robotic workflow. The simulation setup, scenarios and results are discussed below.

Table 3: Simulation parameters

Parameter	Value
Population size	50
No of generations	50-500
Mutation rate	0.5
Crossover rate	0.5
Number of tasks in a workflow	35-65
Number of computing resources	15-45
Processing speed of virtual resources	10000-30000 MIPS
Processing speed of local resources	8000-15000 MIPS
Per unit time (sec) energy consumption of virtual resources	50-150 Watt
Per unit time (sec) energy consumption of local resources	40-90 Watt
Tasks data Size	5000-10000 MI
Per unit time (sec) monetary cost of virtual resources	0.50-0.90 Cents
Per unit time (sec) monetary cost of local resources	0.25-0.40 Cents
Allowable completion time of all tasks	4000-5000 ms
Maximum allowable energy consumption of workflow	1500-2500 Watt
Total Budget	150-200 Cents
Data dependency threshold	1200-1500 ms
Communication bandwidth	128-512 Kbps
Ratio of local and virtual resources in resource pool	1/3
Ratio of dependent and independent tasks	2/5

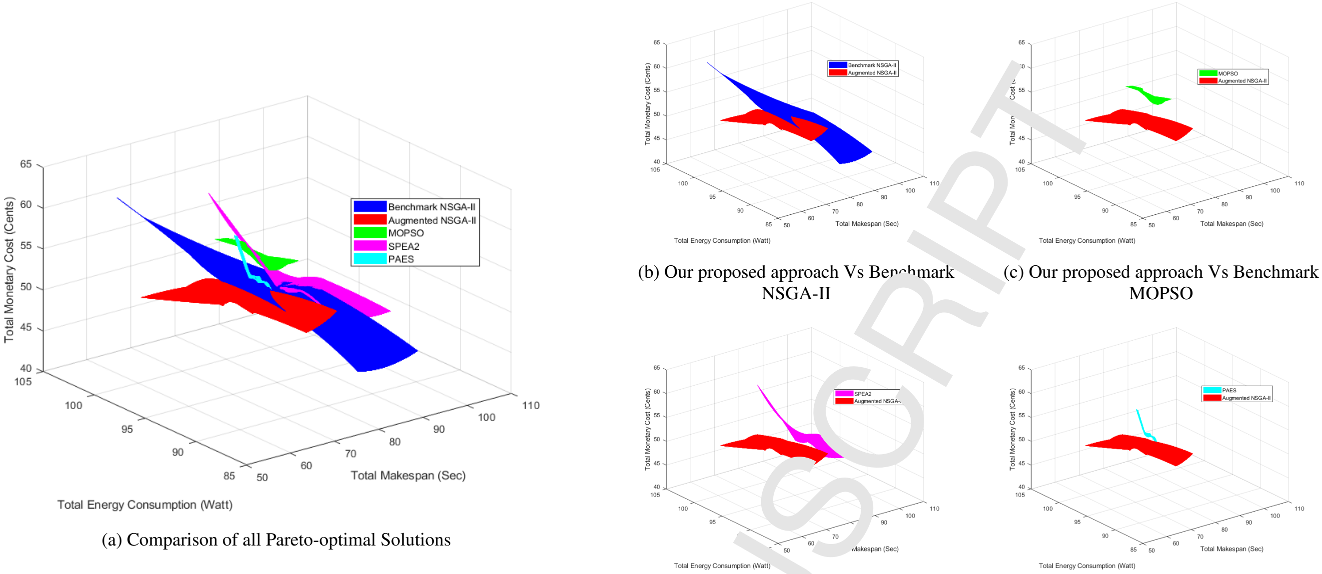
5.1. Simulation Environment

To conduct our experiments in Matlab which is commonly implemented, we use synthetic workload, driven from real-world references, and select system parameters carefully on different trials for fair evaluation [20] [25] [37]. The parametric values for the simulation environment are summarized in Table 3. The value of simulation parameters within a specific range is set by discrete uniform distribution. Apart from proximity based classification of computing resources (local and virtual), to reflect the resource heterogeneity, we consider three types of resources (low, medium and high) based on their processing speed for ease of the simulation. Per unit energy consumption of resources vary time to time according to their speed and computational processes running on them.

5.2. Simulation Scenarios and Result Analysis

While comparing the proposed policy with the existing multi-objective optimization approaches, pareto front solutions, the number of generations require to meet stopping criteria, impacts of varying number of tasks and resources on the objective parameters are considered as performance metrics. In addition, impact of Edge Cloud architecture in the system is analyzed. Impact of the complexity of different tasks are also addressed by analyzing how the system deals with varying number of inter-dependent tasks of robotic workflow and maintain the data dependency threshold. Evaluation results acquired through simulating different scenarios show that our augmented NSGA-II algorithm outperforms the state-of-the-art works well across by at least 18% in optimizing all objectives.

Comparison of pareto-optimal solutions. After 200 generations, the pareto-optimal solutions of our augmented NSGA-II along with benchmark NSGA-II, MOPSO, SPEA2 and PAES algorithm on fixed number of heterogeneous tasks



(a) Comparison of all Pareto-optimal Solutions

(b) Our proposed approach Vs Benchmark NSGA-II

(c) Our proposed approach Vs Benchmark MOPSO

(d) Our proposed approach Vs Benchmark SPEA2

(e) Our proposed approach Vs Benchmark PAES

Figure 4: Comparison of pareto-optimal solutions.

(50) and resources (30) are depicted in Fig. 4. In this scenario, each pareto-optimal solution of our optimization approach provides better outcome for three objectives, compared to other benchmark strategies as shown in Fig. 4 (a). The initial population of proposed approach that is determined based on the task's size and processing speed of resources, inherently minimizes the overall makespan, energy consumption and total cost. Selection of the chromosome having minimum distant solution from the origin for crossover further improves its performance. Therefore, after a fixed number of generations, it provides significantly better solution than the random population used in Benchmark NSGA-II as represented in Fig. 4 (b). In the selection of chromosome for generating first child population from parent population, our augmented NSGA-II approach selects the chromosome with the best fitness value. It also complements finding better solutions compared to Benchmark NSGA-II, where all chromosomes are considered for applying genetic operators.

In MOPSO algorithm, potential solutions fly through the problem space by following the current optimum particles [16]. It follows a one-way information sharing mechanism, and the evolution only looks for the best solution. In comparison to benchmark NSGA-II, MOPSO multiplies the chances to keep individuals' changes, making it easier to maintain diversity in the solution space. However, in terms of pareto-optimal sets quality, our augmented NSGA-II as well as benchmark NSGA-II algorithm give shorter distinct non-dominated set while generating pareto-optimal front. As benchmark NSGA-II, MOPSO also contains random process such as random initial population. Even, while doing the trade-off between the global and local best solution, it uses a random weight factor that can affect the performance of the algorithm due to number of tasks, number of constraints and inter-relation among the constraints. Our augmented NSGA-II is relatively more robust as it is less dependent on a random technique as shown in Fig. 4 (c).

However, compared to SPEA2 and PAES algorithm, MOPSO is faster and require less algorithm parameters to handle. An archive of the non-dominated set is kept separate from the population of candidate solutions in the evolutionary process of SPEA2. Solutions by SPEA2 have significantly better diversity than NSGA-II ones due to the better diversity maintenance strategy of SPEA2. However, SPEA2 is more computationally expensive to run than NSGA-II and is thus more time consuming. Moreover, in SPEA2 the search populations are randomly initialized, where as in our augmented NSGA-II a pre-sorted population is initialized heuristically. For these reasons, our augmented NSGA-II gives better pareto-optimal solutions than SPEA2 as depicted in Fig. 4 (d). On another side, PAES uses local search from one current solution to generate a new candidate and concurrently compares the current solution with the candidate solution, and also maintain an archive to keep the best solutions found so far. To maintain the archive, computational complexity of PAES algorithm is the highest among others. This algorithm is also based on random initial population and for larger number of objectives it cannot achieve good results as our augmented NSGA-II and other studied approaches that is reflected in Fig. 4 (e).

Therefore, pair wise comparing the fronts produced by the different algorithms suggests that our augmented NSGA-II has superior performance over others. It converges fast while maintaining a good quality and diversity of obtained solutions on this multi-constrained problem.

Required generations to meet stopping criteria. The simulation results on Fig. 5 represents the efficacy of our proposed approach in achieving no further optimization state

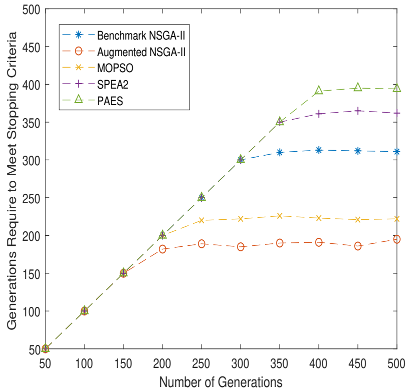


Figure 5: Number of generations require to meet stopping criteria.

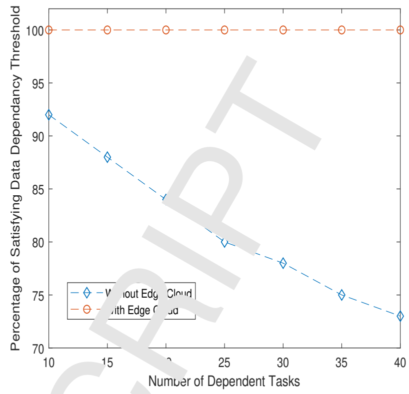


Figure 6: Percentage of satisfying data-dependancy threshold.

within less generations compared to the benchmark NSGA-II, MOPSO, SPEA2 and PAES algorithms. This scenario is simulated with fixed number of tasks (50) and resources (30). The simulation is run for at most 500 generations and it shows that our approach meets the stopping criteria within 200 generations. For using pre-sorted population, compared to other benchmarks, our augmented NSGA-II implicitly advances a certain number of generations in meeting the stopping criteria. Pre-sorted parent population and selection of chromosome with the best fitness value while generating child population, help to meet stopping criteria and converge to the pareto-optimal solutions faster than others.

As in MOPSO the evolution only looks for the best solution, it outperforms other benchmark algorithms (NSGA-II, SPEA2, PAES). Algorithmically, it deals with less number of parameters, which complements to meet the stopping criteria within less number of generations. Because of the random initial population, it cannot outperform our augmented NSGA-II. Again, due to the maintenance of archive, although SPEA2 and PAES can converge to the pareto-optimal solutions, for updating the archive they require additional time that take larger number of generations to meet stopping criteria than benchmark NSGA-II. However, in PAES, a single parent generates a single offspring in combination with a historical archive that records the non-dominated solutions previously found. This process increases the computational complexity of the algorithm compared to SPEA2 and others. Fig. 5 also demonstrates that all studied algorithms are able to find a good approximation of the pareto-optimal set of solutions, but differ in the rate of convergence to the optimal solutions in terms of meeting stopping criteria.

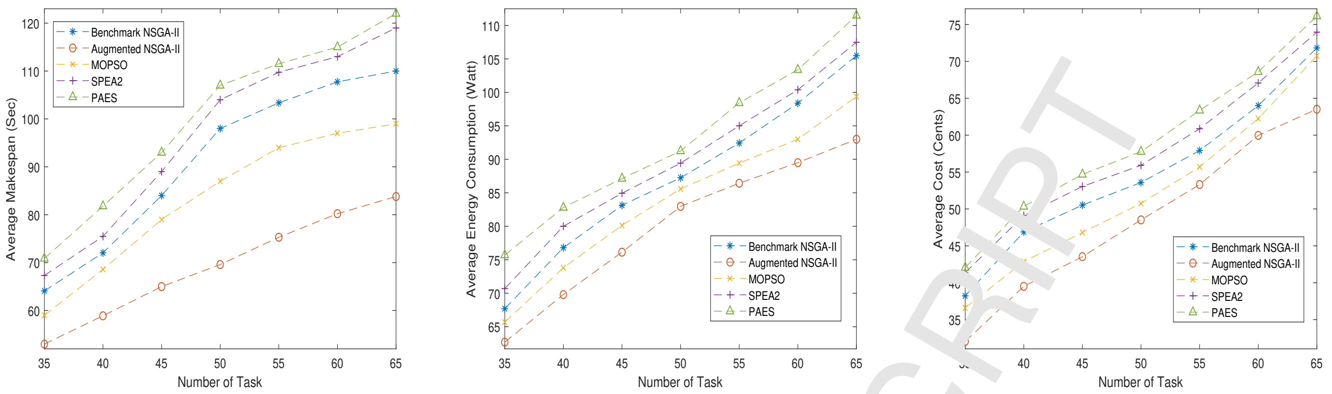
It is worth mentioning that, while dealing with real-time systems, the latency of running large number of generations to find the optimal solutions is not acceptable. Therefore, on basis of the results, we can claim that for real-time systems, our proposed policy performs better than the other benchmarks and PAES is the least applicable for such systems.

Impact of Edge Cloud on inter-dependent tasks. On fixed number of generations (200) and computing resources (30), the impact of Edge Cloud in dealing with varying number of inter-dependent tasks of robotic workflow is represented in Fig. 6. Compared to Cloud based multi-robot system, the Edge Cloud based system performs significantly better. Since, Edge Cloud brings virtual computing resources closer to the robots, distributed placement of inter-dependent tasks on both local and virtual resources does not violate the data-dependency threshold of the dependent tasks. For validating the impact of Edge Cloud on inter-dependent tasks, we use the percentage of satisfying data-dependency threshold $\hat{\delta_T}$ as performance metric. It is calculated by the ratio of number of tasks of a workflow that maintain the data-dependency threshold s_T and the total number of dependent tasks on that workflow d_T as Eq. 13. The higher percentage denotes higher efficiency of the system to handle inter-dependent tasks of a robotic workflow.

$$\hat{\delta_T} = \frac{|s_T|}{|d_T|} \times 100\% \tag{13}$$

Fig. 6 depicts that for increasing number of dependent tasks, the percentage of satisfying data-dependency threshold remains constant with almost zero violation. As the Cloud is deployed geographically far from the Edge Cloud, with the increasing number of dependent tasks, violation of data-dependency threshold increases due to higher communication latency.

Impacts of varying number of tasks. The impact of varying number of tasks on the average makespan, energy consumption of resources and monetary cost of resource usage for our proposed policy and other benchmark strategies are demonstrated in Fig. 7 (a)-(c) respectively. Each data point on the graphs represents the mathematical average of the pareto-optimal solutions for all objectives. The number of generations (200) and computing resources (30) are remained unchanged during simulating this scenario. Since the resource number is fixed, the graphs of Fig. 7 (a). depict that the average makespan sharply increases in all the studied approaches with the increasing number of tasks in a robotic workflow. However, the rate of increase is obviously less in our augmented NSGA-II in comparison with other benchmarks. This is mainly owing to pre-sorted population that provides improved solutions in our approach. In the pre-sorted population, the computing



(a) (b) (c)

Figure 7: For varying number of tasks (a) Average makespan (b) Average energy consumption of resources (c) Average cost of system

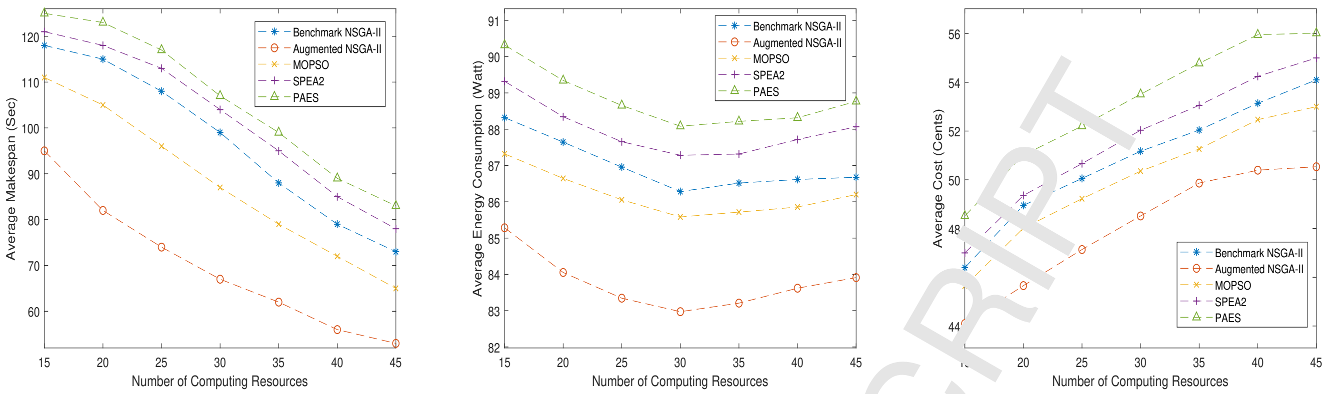
resources are categorized based on their processing speed for assigning the tasks to a particular type of computing resources according to their size. As a consequence, it generates efficient initial population and this population is consistently refined after each generation for increasing number of tasks than the others. It is also illustrated that MOPSO outperforms other benchmark algorithms due to its limited number of parameters in the algorithm. Although benchmark NSGA-II takes slightly more time compared with MOPSO to find pareto-optimal solution, the quality of its solution is better than SPEA2 and PAES. Albeit, SPEA2 is able to faster finding the best pareto front than NSGA-II, but the quality of this pareto-optimal solutions degrades for increased number of tasks in such scenario. Because of the computational complexity PAES does not achieve so good results as SPEA2 achieves. The same reasons are also reflected on other objective values (energy consumption of resources and the monetary cost of the system for executing the tasks on the computational resources) as shown in Fig. 7 (b) and Fig. 7 (c) respectively.

Finally, it is revealed from the results that while optimizing all objectives, our augmented NSGA-II puts good impact on achieving better performances compared with its counterpart. Though all the studied approaches give gradual increasing objective values for the increasing number of tasks, the increase rate is lower in our approach compared to others.

Impacts of varying number of computing resources. The impacts of varying number of resources on the objective parameters (average makespan and energy consumption of resources and cost of system) are represented on the graphs of Fig. 8. In this scenario, number of generations (200) and tasks (50) remain fixed. For fixed number of tasks, as the number of resources increases, the average makespan gets decreased for all approaches as depicted in Fig. 8 (a). The proposed approach exhibits better solution for increasing number of computing resources, as it initially selects better population for evolution. Whereas, the benchmark NSGA-II and other approaches do not consider this issue at all. For the same reasons, the proposed one also performs better in terms of energy consumption as shown in Fig. 8 (b). For increasing number of resources with fixed number of tasks, in all studied approaches, at first energy consumption of resources gets significantly minimized as the tasks are completed in reduced time. In the later part, the number of resources becomes relatively higher than the demand, which consequently rises the energy consumption. However, the rate of increase is lower in approach compared to benchmark NSGA-II, MOPSO, SPEA2 and PAES.

Inherently, the graphs in Fig. 8 (c) show that our proposed approach can optimize cost better compared to other approaches. However, as the number of resources increases, the total cost also gets increased with less increasing rate in our augmented NSGA-II. In comparison, MOPSO cannot minimize as much as our approach as it deals with random population. The increasing rate of average cost of resource usage for executing the tasks is almost similar in MOPSO, NSGA-II and SPEA2, although the rate is higher in PAES for the computational complexity required for maintaining an archive.

In summary, some of the most important factors involved in the superiority of our augmented NSGA-II is its pre-sorted initial population and the selection process of chromosome for generating child population by extending the genetic operators. Though MOPSO algorithm performs better in our considered problem domain than the benchmark NSGA-II, however, the crowding distance operator for NSGA-II during the selection performs well in terms of finding diversified solution and quality of pareto-optimal solutions, leading us to select benchmark NSGA-II to be augmented. In our augmented NSGA-II, the population diversity in different generations is well preserved by dividing the tasks and resources into several categories. It is numerically identified that our proposed approach can optimize the three objectives by 18% better on average than all the studied approaches in different scenarios. In a nutshell, our proposed augmented NSGA-II is best suited to be used optimizing multi-objective system parameters while allocating computational resources for robotic workflow in smart factory.



(a) (b) (c)

Figure 8: For varying number of computing resources (a) Average makespan (b) Average energy consumption of resources (c) Average cost of system

6. Conclusion and Future Directions

In this paper, we propose an Edge Cloud robotic framework that is used for allocating computing resources for multi-robotic workflow of smart factory applications. To overcome the limitations of remote Cloud, a middle tier Edge Cloud is incorporated between local robots and remote Cloud that satisfies inter-task dependency within a workflow. The resource allocation problem is formulated as a constrained multi-objective optimization problem that optimizes the makespan for completing the tasks, the energy consumption of resources and the total cost for executing the tasks simultaneously. To solve this problem, the classical multi-objective evolutionary approach, namely NSGA-II is redesigned in our system with pre-sorted population and the fittest chromosome selection during crossover. Our proposed approach is compared with existing optimization approaches (benchmark NSGA-II, MOPSO, SPEA2, PAES) and it gives better performance in different simulation scenarios.

We are confident that our solution can be applied in other real-world scenarios in future. This work can also be extended to deal with the dynamic environment (mobility of robots, link failures, limited bandwidth) of smart factory. Utilization of Edge resources for real-time applications will be a good extension of our work as well.

Acknowledgement

The authors would like to thank Data61, CSIRO, Australia for supporting the work.

References

[1] K. Baba, S. Ino, Y. Takami, Y. Motoki, M. Mori, K. Kido, Y. Arimura, Fire-fighting robot, Google Patents, US Patent 7,182,144 (2007).
[2] A. Davids, Urban search and rescue robots: from tragedy to technology, IEEE Intelligent Systems 17 (2) (2002) 81–83.
[3] G. Hu, W. P. Tay, Y. Wen, Cloud robotics: architecture, challenges and applications, IEEE Network 26 (3) (2012) 21–28.
[4] Z. Du, L. He, Y. Chen, Y. Xiao, P. Gao, T. Wang, Robot cloud: Bridging the power of robotics and cloud computing, Future Generation Computer Systems 74 (2017) 337 – 348.
[5] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, A. V. Vasilakos, Cloud robotics: Current status and open issues, IEEE Access 4 (2016) 2797–2807.
[6] B. Kehoe, S. Patil, P. Abbeel, K. Goldberg, A survey of research on cloud robotics and automation, IEEE Transactions on Automation Science and Engineering 12 (2) (2015) 398–409.
[7] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, B. Yin, Smart factory of industry 4.0: Key technologies, application case, and challenges, IEEE Access 6 (2018) 6505–6519.
[8] S. B. Tom Duckett, Simon Pearson, B. Grieve, Agricultural robotics: The future of robotic agriculture, Technical Report, UK-RAS Network White Papers, ISSN 2398-4414 (2018).
[9] S. Kumar, C. S. Joshi, Robotics-as-a-service: Transforming the future of retail, Technical Report, Tata Consultancy Services Limited (2014).
[10] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Business & Information Systems Engineering 6 (4) (2014) 239–242.
[11] Y. Lu, Industry 4.0: A survey on technologies, applications and open research issues, Journal of Industrial Information Integration 6 (2017) 1–10.
[12] H. Yan, Q. Hua, Y. Wang, W. Wei, M. Imran, Cloud robotics in smart manufacturing environments: Challenges and countermeasures, Computers & Electrical Engineering 63 (2017) 56 – 65.
[13] M. R. Mahmud, M. Afrin, M. A. Razzaque, M. M. Hassan, A. Alelaiwi, M. Alrubaian, Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure, Software: Practice and Experience (2016).
[14] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646.
[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[16] C. A. C. Coello, M. S. Lechuga, Mopso: a proposal for multiple objective particle swarm optimization, Proceedings of Congress on Evolutionary Computation (2002) 1051–1056.
[17] E. Zitzler, M. Laumanns, L. Thiele, Spea2: Improving the strength pareto evolutionary algorithm, Technical Report (2001).
[18] J. Knowles, D. Corne, The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation, Proceedings of Congress on Evolutionary Computation (1999).
[19] L. Wang, M. Liu, M. Q. H. Meng, A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems, IEEE Transactions on Cybernetics 47 (2) (2017) 473–484.
[20] L. Wang, M. Liu, M. Q.-H. Meng, A pricing mechanism for task oriented resource allocation in cloud robotics, Robots and Sensor Clouds, Springer International Publishing (2016) 3–31.
[21] H. Liu, S. Liu, K. Zheng, A reinforcement learning-based resource allocation scheme for cloud robotics, IEEE Access 6 (2018) 17215–17222.
[22] D. Wu, G. Zeng, L. Meng, W. Zhou, L. Li, Gini coefficient-based task allocation for multi-robot systems with limited energy resources,

IEEE/CAA Journal of Automatica Sinica 5 (1) (2018) 155–168.

[23] C. Mouradian, S. Yangui, R. H. Glitho, Robots as-a-service in cloud computing: Search and rescue in large-scale disasters case study, CoRR (2017).

[24] A. Rahman, J. Jin, A. Cricenti, A. Rahman, D. Yuan, A cloud robotics framework of optimal task offloading for smart city applications, in: IEEE Global Communications Conference, 2016, pp. 1–7.

[25] A. Rahman, J. Jin, A. Cricenti, A. Rahman, M. Panda, Motion and connectivity aware offloading in cloud robotics via genetic algorithm, in: IEEE Global Communications Conference, 2017, pp. 1–6.

[26] A. Rahman, J. Jin, A. Rahman, T. Cricenti, A. Kulkarni, Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance, IEEE Transactions on Industrial Informatics, in press (2018).

[27] L. Agostinho, L. Olivi, G. Feliciano, F. Paolieri, D. Rodrigues, E. Cardozo, E. Guimaraes, A cloud computing environment for supporting networked robotics applications, in: 9th IEEE International Conference on Dependable, Autonomic and Secure Computing, 2011, pp. 1110–1116.

[28] C. Mouradian, F. Z. Errounda, F. Belqasmi, R. Glitho, An infrastructure for robotic applications as cloud computing services, in: IEEE World Forum on Internet of Things, 2014, pp. 377–382.

[29] Y. Chen, Z. Du, M. García-Acosta, Robot as a service in cloud computing, in: 5th IEEE International Symposium on Service Oriented System Engineering, IEEE Computer Society, Washington, DC, USA, 2010, pp. 151–158.

[30] X. Zheng, C. Julien, R. Podorozhny, F. Cassez, T. Rakotoarivelo, Efficient and scalable runtime monitoring for cyber–physical system, IEEE Systems Journal 12 (2) (2018) 1667–1678.

[31] S. Ghasemi-Falavarjani, M. Nematbakhsh, B. S. Ghahfarokhi, Context-aware multi-objective resource allocation in mobile cloud, Computers and Electrical Engineering 44 (2015) 218–240.

[32] D. Shah, Gossip Algorithms (Foundations and Trends in Networking) (2007).

[33] R. Y. Zhong, X. Xu, E. Klotz, S. T. Newman, Intelligent manufacturing in the context of industry 4.0: a review, Engineering 3 (5) (2017) 616–630.

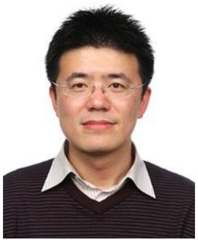
[34] A. Rahman, B. Verma, Cluster based ensemble classifier generation by joint optimization of accuracy and diversity, International Journal of Computational Intelligence and Applications 12 (04).

[35] M. Hamdan, A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms, International Journal on AI Tools 20 (01) (2011) 209–219.

[36] K. Deb, M. Goyal, A combined genetic adaptive search (geneas) for engineering design, Computer Science and Informatics 26 (1996) 30–45.

[37] J. Huang, R. J. Kauffman, D. Ma, Pricing strategy for cloud computing, Decis. Support Syst. 78 (C) (2015) 80–92.

Mahbuba Afrin is currently pursuing her PhD degree at Swinburne University of Technology, Australia, with Tuition Fee Scholarship (TFS). She has also been awarded Data61 PhD Scholarship to work in collaboration with Commonwealth Scientific and Industrial Research Organisation (CSIRO). She received her M.Sc. (2017) and B.Sc. (2015) in Computer Science and Engineering from University of Dhaka, Bangladesh. Before starting PhD programme, she worked as a Lecturer at Department of Computer Science and Engineering in United International University, Bangladesh (2015-2017) and as a Research Assistant in Green Networking Research Group, University of Dhaka (2013-2015). Her research interests include Cloud Networked Robotics, Multi-Robot Systems, Edge Computing, Cloud Computing, Cloud Radio Access Network, Mobile Cloud Computing.



Jiong Jin (IEEE M'11) received the B.E. degree with First Class Honours in Computer Engineering from Nanyang Technological University, Singapore, in 2006 and the Ph.D. degree in Electrical and Electronic Engineering from the University of Melbourne, Australia, in 2011. He is currently a Senior Lecturer in the School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia. Prior to it, he was a Research Fellow in the Department of Electrical and Electronic Engineering at the University of Melbourne from 2011 to 2013. His research interests include network design and optimization, fog and edge computing, robotics and automation, Internet of Things and cyber-physical systems as well as their applications in smart manufacturing, smart transportation and smart cities.

Ashfaqur Rahman is a Senior Research Scientist and team leader at Data61 division of CSIRO. He is a data scientist and working as a researcher for nearly 15 years. His key research areas are machine learning and Data mining. More specifically Ensemble learning and fusion, Evolutionary Algorithm based network optimization, Distributed Machine Learning for Big Data, Feature selection/weighting methods, Image segmentation and classification. Dr. Rahman received his Ph.D. degree in Information Technology from Monash University, Australia in 2008. He has published around 80 peer-reviewed journal articles, book chapters and conference papers.

Yu-Chu Tian (M'00) received the B.S. degree in electrical engineering from Wuhan Institute of Engineering, Wuhan, China, in 1982, the M.S. degree in electrical engineering from East China University of Science and Technology, Shanghai, China, in 1987, and the Ph.D. degree in industrial automation from Zhejiang University, Hangzhou, China, in 1993. Over the last many years, he has worked in Zhejiang University; Hong Kong University of Technology, Hong Kong, China; Curtin University of Technology, Perth, Western Australia; and the University of Maryland at College Park, Maryland, USA. Since 2002, he has been with the Queensland University of Technology, Brisbane, Queensland, Australia, as a Professor of Computer Science. His current research interests include big data computing, cloud computing, real-time computing, computer networks and control theory and engineering. He is the editor-in-chief of Springer's book series Handbook of Real-Time Computing, and an Associate Editor for a few international journals.

Ambarish Kulkarni is a Lecturer-Computer Aided Engineering at Swinburne University of Technology. He specializes in mechanical design with focus on finite element, virtual and augmented reality techniques. His research focuses on product and process development activities. He is currently working on design of medical, automotive process/products improvements using virtual and augmented reality tools. His current projects are on electric vehicle drivetrains, battery packaging, bushfire shelter and therapeutic sleep system designs. He is specialized in using augmented virtual design tools with significant field and industry experience. Further applications of his research have included bio-mechanical modelling and virtual sub-system designs. He was in industrial R&D career and became a full-time academic since 2009. He is also actively involved in teaching finite element methods, augmented virtual reality, graphics design and supervising students. He is a Fellow member of the Institution of Engineers-Australia, member-Society of Automotive Engineers-International and American Society of Mechanical Engineers-USA.

Highlights of "**Multi-Objective Resource Allocation for Edge Cloud based Robotic Workflow in Smart Factory**"

- An Edge Cloud based multi-robot framework for robotic workflow in smart factory.
- Simultaneous optimization of makespan, energy consumption and total cost to allocate computational resources of robotic workflow.
- Augmentation of NSGA-II algorithm with pre-sorted population and redesign of genetic operators.