Multi-objective Computation Offloading for Cloud Robotics using NSGA-II

Rihab Chaari
PRINCE Research Laboratory
University of Sousse
Sousse, Tunisia
rihab.chaari@coins-lab.org

Omar Cheikhrouhou
CES Laboratory
ENIS, University of Sfax
Sfax, Tunisia
omar.cheikhrouhou@isetsf.rnu.tn

Anis Koubâa
ISEP/IPP
Polytechnic Institute of Porto
Porto, Portugal
akoubaa@coins-lab.org

Habib Youssef
PRINCE Research Laboratory
University of Sousse
Sousse, Tunisia
habib.youssef@fsm.rnu.tn

Habib Hamam
Faculty of Engineering
University of Moncton, Canada
Canada
habib.hamam@umoncton.ca

***Abstract*—With the emergence of cloud robotics, computation offloading presents a new trend in cloud computing that has been applied to robots; to provide them with resources for performing computationally intensive tasks. In most scientific research, the main objectives behind computation offloading are reducing energy consumption and minimizing the execution time of robotics applications. However, these two metrics are conflicting, and optimizing them simultaneously is challenging. Reducing energy consumption may lead to a rise in the completion time, and vice-versa. In this paper, we consider the problem of optimization of energy consumption and completion time in a cloud robotic system. We formulated the offloading decision as a multi-objective optimization problem. We further adapted the Non-dominated Sorting Genetic Algorithm (NSGA-II) to find a set of Pareto-optimal solutions. Through simulations, we demonstrated that our offloading solution can save 80% of the robot's energy consumption; and reduce 70% of the application completion time. We proved also the adaptability of the model against bandwidth changes.**

***Index Terms*—Cloud Robotics, Computation offloading, Offloading decision, Multi-objective optimization, NSGA-II**

I. INTRODUCTION

In the last ten years, cloud robotics has emerged as a promising approach where the combination of the Internet and remote cloud resources are leveraged to improve the robots' applications performance by providing them with more capabilities and computation resources [1]. The cloud infrastructure and the Internet can propose a lot of potential opportunities to robots, including access to large volumes of data, accurate remote computation, collective learning, information sharing, to name a few.

Computation offloading is an approach to migrate the processing of computationally intensive tasks from the robot to external resources to reduce energy consumption and computation time. Although it can significantly tackle the limitations of robots in terms of resources, achieving an efficient offloading solution is still a challenging issue. An efficient offloading solution should consider two primary conflicting performance metrics for mobile systems:

- **Completion time:** From a cloud robotic perspective, the completion time is the duration between sending data to the cloud and receiving the result. For computation-intensive robotic applications, it is crucial to reduce the execution time and improve their responsiveness. When the execution time increases for computation-intensive tasks, the best strategy would be to offload these tasks to the cloud to leverage its high-speed processing capabilities. However, communication delays will also play an adversary role in achieving real-time performance.
- **Energy consumption:** Energy consumption is another aspect that must be considered in computation offloading. It is due to both computation and communication. In the case of a computation offloading scenario, communication will be the major component for energy dissipation due to the transmission of heavy tasks (e.g. video stream) from the robot to the cloud. In the case of local processing, the onboard computation will be the main source of energy consumption. It is therefore important to make the best tradeoff to maximize the overall benefit.

Both energy consumption and response time are the most concerning issues in robotic applications. By computation offloading, we expect that both energy and time are optimized. However, in some cases, it is not possible to achieve both objectives. The optimization of one objective can influence the other. For instance, cloud architectures distribute the computation on several data centers, which can save energy. However, the path traveled between data centers and robots can result in significant latency and can increase the delay of real-time applications. This work is motivated by our previous work [2], in which we designed a distributed cloud robotic architecture for static computation offloading. The performance evaluation of the architecture revealed a profit in resource utilization including CPU, memory, and battery. But, it has increased the latency because of communication delays between cloud data centers and mobile robots.

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

Our main objective is to optimize both the energy consumption of robots and completion time, taking into account the trade-off between both. In our offloading decision solution, we aim to consider also the uniformity of the power consumption of all the robots; since this is beneficial for the durability and productivity of the set of robots working collaboratively as a team. The main contributions of this paper are as follow:

1) We model the offloading decision problem as a MOOP. We defined three objective functions: (1) reduce energy consumption, (2) minimize completion time, and (3) guarantee power uniformity among robots.
2) We propose an adaptive version of the Non-dominated Sorting Genetic Algorithm (NSGA-II) to search the list of Pareto front, that optimize both energy consumption and completion time.
3) We propose a new chromosome encoding to present the offloading decision that takes into consideration the heterogeneity of the available resources.

The remainder of this paper is organized as follows. In Section 2, we discuss the system model and the problem formulation. In Section 3, an adaptive NSGA-II is proposed to solve the multi-objective optimization problem. In Section 4, simulations were performed to prove the effectiveness of our proposed model. Finally, we conclude in Section 5.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we established the network model for a proxy-based cloud robotic architecture. Then, we introduce the energy and the time cost. Finally, we define the problem formulation as an optimization of the total energy consumption, the application completion time, and the remaining power of the robots.

A. Network Model

We consider a cloud robotic network that contains a team of robots and cloud servers working collaboratively to execute a defined application, which are organized in a proxy-based model, as depicted in Figure 1. The robots communicate with each other and with the proxy server through wireless communication, while the proxy communicates with the cloud servers through wired communication. Each robot defines a set of n tasks to be executed locally or offloaded to the cloud. The proxy performs the offloading decision and shares it with all the roots and cloud servers.

We define the infrastructure $I = (R, S)$, where R is the set of robots available to accomplish the mission, and S represents the set of servers available in the cloud. The set of robots R is presented as R=(R_1, R_2, ... , R_M) with M is the total number of available robots. The set of cloud servers is presented as S=(S_1, S_2, ... , S_K) with K is the total number of available cloud servers.

Each robot is defined as R_m=(Id_{R_m}, CPI_{R_m}, S_{R_m}, V_{R_m}, Pt_{R_m}) with Id_{R_m} is the identifier of the robot R_m, CPI_{R_m} is the number of clock cycles per instructions, S_{R_m} is the clock speed of the robot processor, V_{R_m} is the robot velocity, and Pt_{R_m} is the total available power.

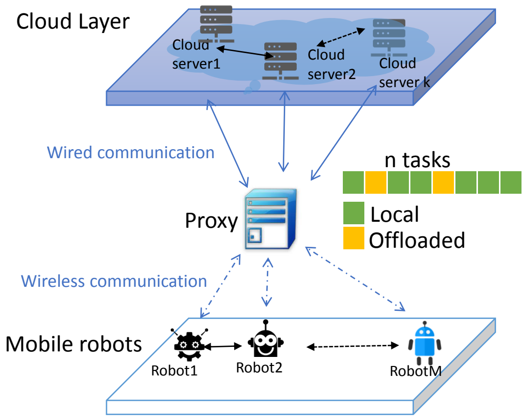


Fig. 1. The proposed system model of a cloud robotic network (with K cloud servers and M robots)

Each cloud server is defined as S_k=(Id_{S_k}, CPI_{S_k}, CP_{S_k}) with Id_{S_k} is the identifier of the server S_k. CPI_{S_k} is the number of clock cycles per instructions. CP_{S_k} present the number of available core processors.

We define also D the offloading decision vector presented by D=$\{d_1$, d_2, ... , $d_n\}$, where n is the total number of tasks. Each decision $d_i \in D$ is a tuple (t_i, r_j) where the task t_i will be executed on resource r_j with $r_j \in \{Id_{R_1}$, Id_{R_2}, ..., Id_{R_m}, Id_{S_1}, Id_{S_2}, ..., $Id_{S_k}\}$.

In this paper, the robot's application is modeled into tasks T=$\{t_1, t_2, ... , t_n\}$ with n the total number of tasks. We present the set of tasks as a Directed Acyclic Graph (DAG) $G = (V, E)$. Vertices V represent the set of tasks and edges E present the execution dependency between tasks. Each edge $e_i \in E$ is the link between task t_i and task t_{i+1}. We assume that:

- Task preemption is not allowed: once an agent (whether a robot or a cloud server) starts the execution of a task, it must finish it without interruption.
- Each task should be accomplished by a single computing unit (a robot or a cloud server). If a task requires some services from other agents, it must be divided into more tasks according to these requirements.
- All the tasks are available on the cloud servers as Application Programming Interface APIs or can be downloaded from another site or server through a high network speed.
- Each robot is connected to a cloud server. In turn, a single cloud server can serve the offloading request of more than one robot. The robot is a mono-task computing unit, while cloud servers are multi-task.

B. Cost Model

All the parameters required for the definition of the energy and completion time cost models are depicted in Table I.

TABLE I
NOTATIONS USED IN ENERGY AND COMPLETION TIME MODELS

Notation	Description
CPI	The number of clock cycles per instruction
P_{ex}	The computation power
$N(t_i)$	The number of instructions for task t_i
Sr	The clock speed of the robot processor
P_{mov}	The movement power
Vr	The robot velocity
$d_c(t_i)$	The amount of data collected for task t_i
P_{Data}	The data collection power
T_r	The data transmission rate
P_{idle}	The power when the robot is idle
S_c	The clock speed of the cloud server
P_{trans}	The power used for data transmission
$dt(t_i)$	Data to be transmitted to the cloud for completion of task t_i
B	The network bandwidth
$P(t_i)$	The power required to execute task t_i
$P(r_j)$	The power available on resource r_j

The total energy consumed by the robot for executing a task $t_i \in T$ is denoted as $E_{total}(t_i)$, which can be presented by either $E^{local}_{total}(t_i)$ or $E^{cloud}_{total}(t_i)$. A task should be executed by a single resource: whether on the robot or offloaded to the cloud.

$$E_{total}(t_i) = \begin{cases} E^{local}_{total}(t_i) & \text{if } \mathrm{d}_i \in \{\mathrm{Id}_{R_1}, Id_{R_2}, ..., Id_{R_m}\} \\ E^{cloud}_{total}(t_i) & \text{if } \mathrm{d}_i \in \{\mathrm{Id}_{S_1}, Id_{S_2}, ..., Id_{S_k}\} \end{cases}$$

$E^{local}_{total}(t_i)$ is the overall energy consumed by the robot for the completion of task t_i. It contains the energy for computation $E^{local}_{comp}(t_i)$ (Equation (2)), the energy for movement $E^{local}_{mov}(t_i)$ (Equation (3)), and the energy for data collection $E^{local}_{Data}(t_i)$ (Equation (4)).

$$E^{local}_{total}(t_i) = E^{local}_{comp}(t_i) + E^{local}_{mov}(t_i) + E^{local}_{Data}(t_i) \tag{1}$$

$$E^{local}_{comp}(t_i) = CPI * P_{ex} * N(t_i)/Sr \tag{2}$$

$$E^{local}_{mov}(t_i) = P_{mov} * \sqrt{(x1 - x2)^2 + (y1 - y2)^2}/Vr \tag{3}$$

$$E^{local}_{Data}(t_i) = d_c(t_i) * P_{Data}/T_r \tag{4}$$

If the task t_i requires from the robot to move from one location to another, the energy consumed for movement is presented by Equation (3), where the robot is moving from the initial point $P_1(x_1, y_1)$ to the destination point $P_2(x_2, y_2)$. Here, we suppose that all the points are in the same zone area. Besides, the energy cost of data collection is presented by Equation (4). In this equation, we consider $d_c(t_i)$, T_r, and P_{Data} to be the amount of data collected for task t_i, the data transfer rate, and the power used for data collection respectively.

$E^{cloud}_{total}(t_i)$ is the energy consumption for offloading task t_i to the cloud. Offloading a task to the cloud brings additional energy consumption cost: the energy of data transmission to the cloud $E^{cloud}_{trans}(t_i)$ and the energy when the robot is idle during cloud computation $E^{cloud}_{idle}(t_i)$.

$$E^{cloud}_{total}(t_i) = E^{cloud}_{idle}(t_i) + E^{cloud}_{trans}(t_i) \tag{5}$$

$$E^{cloud}_{idle}(t_i) = CPI * P_{idle} * N(t_i)/S_c \tag{6}$$

$$E^{cloud}_{trans}(t_i) = P_{trans} * dt(t_i)/B \tag{7}$$

The total time required to complete task t_i is presented by $T_{total}(t_i)$, which can be equal to either the robot completion time $T^{local}_{total}(t_i)$ or the cloud completion time $T^{cloud}_{total}(t_i)$ according to the offloading decision.

$$T_{total}(t_i) = \begin{cases} T^{local}_{total}(t_i) & \text{if } \mathrm{d}_i \in \{\mathrm{Id}_{R_1}, Id_{R_2}, ..., Id_{R_m}\} \\ T^{cloud}_{total}(t_i) & \text{if } \mathrm{d}_i \in \{\mathrm{Id}_{S_1}, Id_{S_2}, ..., Id_{S_k}\} \end{cases}$$

The robot completion time $T^{local}_{total}(t_i)$ includes the time for computation of the task on the robot $T^{local}_{comp}(t_i)$, the time for data collection $T^{local}_{Data}(t_i)$, and the time for movement $T^{local}_{mov}(t_i)$. Computation in the cloud comes with additional time: the communication time $T^{local}_{trans}(t_i)$.

$$T^{local}_{total}(t_i) = T^{local}_{comp}(t_i) + T^{local}_{Data}(t_i) + T^{local}_{mov}(t_i) \tag{8}$$

$$T^{cloud}_{total}(t_i) = T^{cloud}_{comp}(t_i) + T^{cloud}_{trans}(t_i) \tag{9}$$

The execution time of task t_i on the robot and on the cloud are respectively presented by Equation (10) and Equation (11).

$$T^{local}_{comp}(t_i) = N(t_i) * CPI/S_r \tag{10}$$

$$T^{cloud}_{comp}(t_i) = N(t_i) * CPI/S_c \tag{11}$$

Equations (12),(13), and (14) are respectively the movement time, data collection time, communication time.

$$T^{local}_{mov}(t_i) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}/Vr \tag{12}$$

$$T^{local}_{Data}(t_i) = Data(t_i)/T_r \tag{13}$$

$$T^{cloud}_{trans}(t_i) = dt(t_i)/B \tag{14}$$

We consider a cloud robotic network, where every single robot has limited power that we denote as P_m for the robot R_m with $m \in \{1, 2, ..., M\}$. An appropriate offloading decision should balance the power consumption between the robots. The standard deviation (SD) [3] is the commonly used measure of dispersion. SD will be used in our model to evaluate the power level of the robotic network. Equation (15) presents the formula of the power level, with $\bar{P}$ is the mean value of the power of all the robots in the network.

$$P^{SD} = \sqrt{(1/M - 1) * \sum_{m=1}^{M} (P_m - \bar{P})^2} \tag{15}$$

C. Problem Formulation

Our main objective is to minimize the total energy consumption, reduce the total completion time, and balance the remaining power of the robots. The search for an optimal offloading decision is constrained by multiple conditions, including interdependence between tasks and their requirements in terms of resources, the available bandwidth, etc. Taking into consideration these constraints, the resource allocation for tasks in a cloud robotic network turns into a multi-objective optimization problem.

$$\text{minimize} \quad (E_{total}, T_{total}, P^{SD}) \tag{16}$$

$$\text{where} \quad E_{total} = \sum_{t_i \in T} (E_{total}(t_i)) \tag{17}$$

$$\text{and} \quad T_{total} = \sum_{t_i \in T} (T_{total}(t_i)) \tag{18}$$

$$\text{subject to.} \quad T_{total} <= T_{deadline} \tag{19}$$

$$T_{t_i} <= \delta \quad \forall t_i \in T \tag{20}$$

$$P(t_i) <= P(r_j) \quad if \quad d_i = Id_{R_j} \tag{21}$$

The formulation of the multi-objective optimization problem for the proposed cloud robotic model is expressed in Equation (16). Having a set of available resources (robots and cloud servers) and a workflow, we aim to simultaneously minimize the total energy consumption, the completion time, and the power level. However, this is subject to some constraints:

- Time constraints: The execution time of a task should not exceed a tolerable execution time δ (Equation (20)). Besides, the total completion time of the workflow should not surpass a tolerable deadline (Equation (19)).
- Power constraint: A task is assigned to a robot for execution only if its power is sufficient to execute it (Equation (21)).

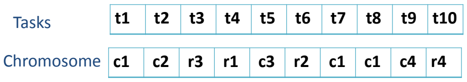
III. PROPOSED SOLUTION

NSGA-II was proposed by Deb et al. in [4]. It is the most popular multi-objective evolutionary algorithm. It is mainly based on two concepts: the nondominated sorting and the crowding distance. The non-dominated sorting concept is used to rank the solutions. While the crowding distance estimates the density of solutions around the set of Pareto front.

The NSGA-II algorithm can be summarized in the following steps: an initial population is generated based on the problem constraints. Then, the individuals are sorted in descending order according to their objective functions. Once the sorting is completed, individuals are selected based on their crowding distance and rank. Next, a binary tournament selection is applied to select individuals to perform genetic operators. Finally, the offspring population and the current population are merged and the individuals of the next generation are collected by selection. The algorithm stops until some stopping criteria are reached.

Seen the heterogeneity of our system model, we adopted a new chromosome encoding. In what follows, we present the proposed new chromosome encoding for our model.

Chromosome encoding. The chromosome encoding, used in this system model, is composed of two fields: a set of tasks $\{t_i | 1 <= i <= n\}$ and a set of available resources (robots or cloud servers). The tasks are mapped to the associated resource according to the encoding string. An example chromosome is presented in che, which present a problem with ten tasks. We adopted this chromosome presentation to guarantee uniformity of energy consumption among robots.



Tasks	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
Chromosome	c1	c2	r3	r1	c3	r2	c1	c1	c4	r4

Fig. 2. Chromosome structure for a problem with ten tasks

Fitness function: In this paper, we consider three objective functions: the energy consumption, the total completion time and the power level. These fitness values are calculated for each individual in the population with respect to the constraints explained by Algorithm 1.

Algorithm 1: Fitness functions calculation

```
Result: $E_{total}$, $T_{total}$, $P^{SD}$
initialization $E_{total}$, $T_{total}$, $P^{SD}$ ;
define $P_c$, $PM$ ;
for each task $t_i \in T$ do
    find $id$ resource allocation $d_i \in$ (R ∪ S) ;
    if $d_i \in R$ then
        $E^{local}_{total}(t_i) = E^{local}_{comp}(t_i) + E^{local}_{mov}(t_i) + E^{local}_{dc}(t_i)$;
        $T^{local}_{total}(t_i) = T^{local}_{comp}(t_i) + T^{local}_{dc}(t_i) + T^{local}_{mov}(t_i)$ ;
        $E_{total}$=$E_{total}$+$E^{local}_{total}(t_i)$ ;
        $T_{total}$=$T_{total}$+$T^{local}_{total}(t_i)$ ;
        measure $Pc_{t_i}$ as the power consumed by task $t_i$ ;
    else if $d_i \in S$ then
        $E^{cloud}_{total}(t_i) = E^{cloud}_{idle}(t_i) + E^{cloud}_{trans}(t_i)$ ;
        $T^{cloud}_{total}(t_i) = T^{cloud}_{comp}(t_i) + T^{cloud}_{trans}(t_i)$ ;
        $E_{total}$=$E_{total}$+$E^{cloud}_{total}(t_i)$ ;
        $T_{total}$=$T_{total}$+$T^{cloud}_{total}(t_i)$ ;
        measure $Pc_{t_i}$ as the power consumed by task $t_i$ ;
    else
        continue ;
    for each robot $R_j \in R$ do
        if $id == j$ then
            update $(Pt_{R_j})$ the power of robot $R_j$ ;
            for each task $t_i \in T$ do
                measure $Powermedian = \sum_{R_j \in R}(Pt_{R_j})$ ;
                append $Powermedian$ to $PM$ ;
                append $Pc_{t_i}$ to $P_c$ ;
            end
    end
end
measure $P^{SD} = \sqrt{(1/M - 1) * \sum_{t_i \in T}(P_m(i) - PM(i)^2}$ ;
```

Genetic operators: As with any evolutionary algorithm, an initial population is first generated. Then, genetic operators are used for better search in the space of solutions. For that, some individuals from the initial population are selected for crossover. In the crossover phase, we proceed as follows. We randomly select two individuals from the population. Then, we generate a sequence of different crossover points, whose number is determined by the length of the chromosome (see Equation (22)) to adapt the crossover operation to different workflow size. We select crossover points, provided that two consecutive points are separated by at least 3 genes. Then, we make the crossover as explained in Figure 3.

$$NumbrecrossoverPoints = chromosome - length \div 10 + 2 \tag{22}$$

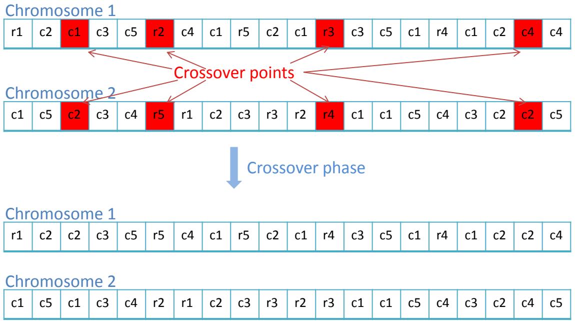

Chromosome 1
r1 c2 c1 c3 c5 r2 c4 c1 r5 c2 c1 r3 c3 c5 c1 r4 c1 c2 c4 c4
Crossover points
Chromosome 2
c1 c5 c2 c3 c4 r5 r1 c2 c3 r3 r2 r4 c1 c1 c5 c4 c3 c2 c2 c5
Crossover phase
Chromosome 1
r1 c2 c2 c3 c5 r5 c4 c1 r5 c2 c1 r4 c3 c5 c1 r4 c1 c2 c2 c4
Chromosome 2
c1 c5 c1 c3 c4 r2 r1 c2 c3 r3 r2 r3 c1 c1 c5 c4 c3 c2 c4 c5

Fig. 3. Crossover phase in the proposed NSGA-II solution

Once the crossover phase is completed, an individual is randomly selected for mutation. We count the number of tasks executed on the robots and those offloaded to the cloud. If the number of tasks executed locally is higher than the number of offloaded tasks, we change the chromosome by changing the execution from the robot to the cloud-based on the mutation probability.

IV. SIMULATION RESULTS AND DISCUSSION

In this section, we evaluate the performance of our multi-objective offloading decision approach. We conducted our experiments in Matlab using the parameters presented in Table II. These parameters were carefully chosen based on research works [5], [6], [7]. We used robots with 3.2 GHz clock speed, 20 cycles per instruction, and 8 bits per instruction. The power consumed by the robot is distributed as follow: $P_{ex} = P_{idle} = 50$ watt, $P_{dc} = 35$ watt, $P_{dt} = 80$ watt, and $P_{mov} = 80$ watt. The data transmission rate is set to 11 Mbit/s according to the IEEE 802.11 standard.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Population size	50
Crossover probability	0.6
Mutation probability	0.4
Bandwidth	0.512(Mbit/s)
The number of tasks in the workflow	30-60
The number of available computing resources	10-30
The number of instructions in each task	$5{*}10^6 - 10^7$
The clock speed of the cloud servers	50-100 Ghz
The allowable completion time interval	[4000-8000]ms

Experiment 1: Benefits of the proposed solution

In the following experiments, we define three computation policies: i.) Computation in the Cloud Policy (CCP) where the computation of all tasks will be offloaded to the cloud, ii.) Computation on the Robot's Policy (CRP) where all tasks will be executed on robots, and iii.) Computation Offloading Policy (COP) where tasks computation follow the proposed offloading strategy. We compared the average fitness value of the energy consumption and the completion time of our computation offloading policy with (CRP) and (CCP) policies. In this experiment, we set the number of iteration to meet stopping criteria to 300 iterations, and the number of tasks to 40 tasks. We used 20 robots in the computation on the robot's policy, 20 cloud servers in the computation in the cloud policy, and 10 robots and 10 cloud servers in the computation offloading policy.

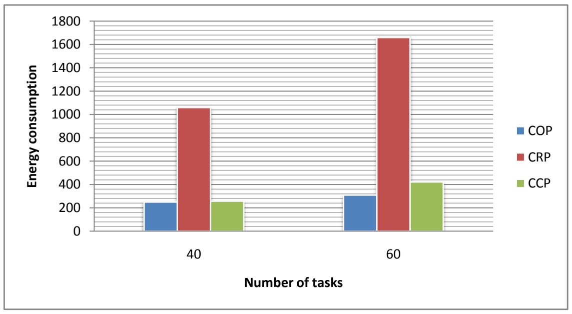


Fig. 4. Comparison of average energy consumption with CCP, CRP, and COP

From Figure 4, we can observe that computation on the robots consumes more energy compared to the CCP and COP. Computation in the cloud and computation using our proposed solution can significantly reduce the energy consumption since computation is done remotely and the energy consumed by the robots for cloud transmission is much less than the energy consumed for computation. Besides, our proposed solution has further improvements in energy consumption since it takes into consideration the completion time of the tasks and the power uniformity among robots to optimize energy consumption. Using our offloading solution, we can approximately optimize 80% of the total energy consumption of the robots.

Figure 5 proves that even the completion time is optimized using the offloading decision approach. We can see that the completion time in the COP policy is less than the completion time of the other policies. Using our offloading solution, the total completion time can be reduced to up to 70%. To process and transfer tasks in the cloud take less time than computation on the robots, due to the powerful processing capabilities of the cloud.

Experiment 2: Adaptability to network bandwidth

In this experiment, we tested the adaptability of our solution to the changes in bandwidth. In this experiment, we used 5 robots and 5 cloud servers, and 40 tasks. We varied the

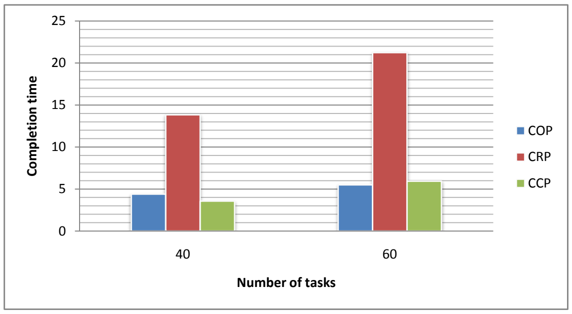


Fig. 5. Comparison of average completion time with CCP, CRP, and COP

bandwidth value, and we measured the number of offloaded tasks.

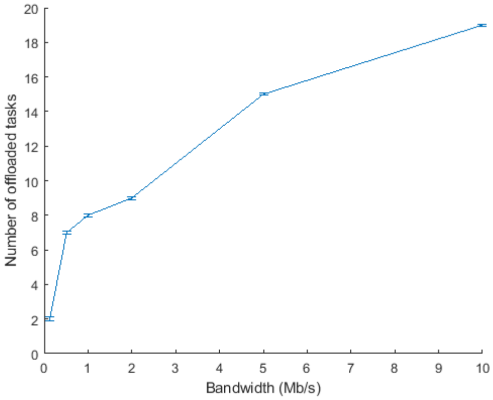


Fig. 6. The adaptability of the NSGA-II based solution to bandwidth changes

The results of the adaptability to the changes in bandwidth in Figure 6 shows a clear progression in the number of offloaded tasks with the rise in the bandwidth. This involves that better bandwidth allows offloading more tasks to the cloud; therefore increase in the total energy consumption.

Experiment 3: Comparison with the original version of NSGA-II

In this experiment, we compare the proposed version of NSGA-II with the original one. For that, we used two metrics: Best Frontier Members (BFM) and Average Frontier Fitness (AFF). BFM presents the best fitness value for each OF in the set of the Pareto frontiers. AFF is measured as the average fitness value in the list of Pareto frontiers for each OF. In this experiment, we used 10 robots and 10 cloud servers. We set the number of iteration to meet stopping criteria to 300 iterations.

From Table III, we can notice that the modified NSGA-II surpasses the original version of NSGA-II according to the BFM metric. The deviation of the original NSGA-II increases as long as the number of tasks increases. For the first objective function, the average deviation of the original NSGA-II is 5.6%, while the deviation is insignificant for the second OF. Besides, AFF in the proposed version of NSGA-II surpasses the original NSGA-II for the first OF. The average deviation of the first OF is 5.6%, and 0.36% for the second OF.

TABLE III
COMPARATIVE RESULTS BETWEEN THE ORIGINAL AND THE PROPOSED NSGA-II

Number of tasks	BFM				AFF			
	Original NSGA-II		Modified NSGA-II		Original NSGA-II		Modified NSGA-II	
	Energy	Time	Energy	Time	Energy	Time	Energy	Time
40	208.82	0.82	208.89	0.81	228.94	2.27	222.18	2.06
50	230	1.04	228.97	1.04	253.19	2.77	244.41	1.97
60	256.5	1.94	251.39	1.61	271.34	2.9	270.3	2.88
70	370.21	2.38	353.48	2.06	383.41	3.79	377.14	3.33
80	385.85	2.65	375.65	2.63	402.74	4.35	397.42	4.02

V. CONCLUSION

The main objective of this paper is to optimize the robot's energy consumption and application completion time. Because Energy consumption and completion time are contradictory objectives, we formulated the offloading decision as a multi-objective optimization problem. Accordingly, we developed an adaptive version of the NSGA-II algorithm to find the set of non-dominated solutions. Simulation results proved that the proposed offloading solution optimizes both metrics (the energy consumption and the completion time) when compared with computation on robots or clouds.

Security is an important dimension of performance in all networked robotic scenarios. Communication and transferring data between robots and remote computing resources can result in some security threats, namely: alteration, denial of service, and information disclosure. In our future work, we aim to propose an offloading decision that optimizes security threads in addition to the energy consumption and completion time.

REFERENCES

[1] Chaâri, R., Ellouze, F., Koubâa, A., Qureshi, B., Pereira, N., Youssef, H., Tovar, E. (2016). Cyber-physical systems clouds: A survey. Computer Networks, 108, 260-278.
[2] Chaari, R., Cheikhrouhou, O., Koubâa, A., Youssef, H., Hmam, H. (2019, June). Towards a distributed computation offloading architecture for cloud robotics. In 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC) (pp. 434-441). IEEE.
[3] Högel, J., Schmid, W., Gaus, W. (1994). Robustness of the standard deviation and other measures of dispersion. Biometrical journal, 36(4), 411-427.
[4] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182-197.
[5] Afrin, M., Jin, J., Rahman, A., Tian, Y. C., Kulkarni, A. (2019). Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory. Future Generation Computer Systems, 97, 119-130.
[6] Rahman, A., Jin, J., Cricenti, A. L., Rahman, A., Kulkarni, A. (2018). Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance. IEEE Transactions on Industrial Informatics, 15(5), 2500-2511.
[7] Rahman, A., Jin, J., Cricenti, A., Rahman, A., Panda, M. (2017, December). Motion and connectivity aware offloading in cloud robotics via genetic algorithm. In GLOBECOM 2017-2017 IEEE Global Communications Conference (pp. 1-6). IEEE.