Dynamic Task Allocation for Robotic Network Cloud Systems

Saeid Alirezazadeh\textsuperscript{1} and Luís A. Alexandre\textsuperscript{2}

\textsuperscript{1}C4-Cloud Computing Competence Center, Universidade da Beira Interior, C4 - Estrada Municipal, 506, 6200-284 Covilhã, Portugal
\textsuperscript{2}NOVA LINCS, Universidade da Beira Interior, Covilhã, Portugal
\textsuperscript{1}Email id: saeid.alirezazadeh@gmail.com

Abstract

Every robotic network cloud system can be seen as a graph with nodes as hardware with independent computational processing powers and edges as data transmissions between nodes. When assigning a task to a node we may change several values corresponding to the node such as distance to other nodes, the time to complete all of its tasks, the energy level of the node, energy consumed while performing all of its tasks, geometrical position, communication with other nodes, and so on. These values can be seen as fingerprints for the current state of the node which can be evaluated as a subspace of a hyperspace. We proposed a theoretical model describing how assigning tasks to a node will change the subspace of the hyperspace, and from that, we show how to obtain the optimal task allocation. We described the communication instability between nodes and the capability of nodes as subspaces of a hyperspace. We translate task scheduling to nodes as finding the maximum volume of the hyperspace.

Keywords — cloud, fog, edge, hyperspace, task scheduling.

1 Introduction

The use of robotic systems is increasing daily. They interact with many aspects of human life, such as industrial and manufacturing [13,26], military [22,29], domestic [25,36] among other [28]. Robotic systems can be classified as single robot or multi-robot. In this paper, we will focus the optimization in terms of the time required for a multi-robot system to conclude its tasks.

To solve a very hard problem, a natural process is to break the problem into so-called elementary problems that are easy to solve or, for which, solutions are already known. Then the solution for the main problem can be obtained collectively from the solutions of elementary problems. One of its instances is in the collaborative scientific literature. We can translate this method to robotic systems, which means that instead of a stand-alone robot performing a task, several robots can cooperate with each other to perform the task. Such a system is called a robotic network. Robotic networks have been widely studied and their main application in disaster management is described in [11,16,19–21,23,30], among others.
Task allocation problem:
\( T = (A_1, \ldots, A_m) \) and \( (T_i)_{i \in \mathbb{N}} = T_1, T_2, \ldots \)

Dynamic: Optimal performance for allocating \((T_i)_{i \in \mathbb{N}}\) (Our result)

Static: Optimal performance for allocating \(T\)

Centralized: Central unit provides task allocation, [5, 8]

Distributed: tasks disperse to all robots, and robots decide whether to perform tasks or not

Single robot, simultaneously time and memory optimization [1]

Multi-robot, time optimization [18]

Behavior-based, [6, 24]

Market-based, [10, 32, 34]

Combinatorial optimization-based, [9, 33]

Evolutionary algorithm-based, [2, 7, 31]

Figure 1: Diagram of studies on task allocation problem. The dashed arrow is only used to represent that the result in [1] for a single robot is not a specific instance of the result of [18].

Robots carry some level of intelligence to automatically perform several tasks. However, the capacity of a robotic network is higher than a single robot and is bounded by the collective capacity of all the robots [11]. In addition, by increasing the number of robots, we are able to increase the capacity, but at the same time, we increase the complexity of the model. Also, most of the tasks related to human-robot interaction, such as speech [15], face [14], and object [35] recognition are computationally demanding tasks.

Cloud robotics is described as a way to handle some of the computational limitations of robots by taking advantage of the internet and cloud infrastructure for delegating computation and also to perform real-time sharing of large data [17]. An important factor to identify the performance of cloud-based robotic systems is deciding whether to upload a newly arrived task to the cloud, processing it on a server (fog computing [4]) or executing it on any of the robots (edge computing [27]), the so-called, allocation problem. Our goal is to provide a theoretical framework to solve the allocation problem for a robotic network cloud system.

2 Related Works

In a multi-robot system, let \( T \) be a finite set of tasks that can be performed by the system. At the time segment, the system is performing a set of tasks, \( T_1 \), which is a subset of the set of all tasks. Simultaneously, a new set of tasks, \( T_2 \), arrives to be performed by the system. As can be seen in Figure 1, there are two types of allocating tasks:

- a **dynamic task allocation** answers the question of how to achieve the optimal performance
from the system by dynamically allocating tasks in the sequence of sets of arrived tasks by time, $(T_t)_{t\in\mathbb{N}}$.

- a static task allocation answers the question of how to achieve the optimal performance from the system by allocating the tasks in the set of all the tasks, $T$.

More details about the works mentioned in Figure 1 can be seen in [1]. As it is shown in Figure 1 the problems we are addressing in this paper are focused in the dynamic allocation problem. We start by describing the mathematical tools used throughout the paper.

3 Preliminaries

We briefly explain the graph theory concepts used throughout the manuscript, see [3] for general graph theory results and [1] oriented to the cloud robotic systems, for more details.

**Definition 1.** A directed graph $G = (V, \overrightarrow{E})$ is defined by the set of vertices of the graph that is the set of algorithms $V = \{A_1, \ldots, A_n\}$ and the set of directed edges that is, a subset of ordered pairs of elements of $V$,

$$\overrightarrow{E} = \{(A_i, A_j) \mid A_j \text{ is using the output of } A_i\}.$$  

**Definition 2.** In a directed graph $G = (V, \overrightarrow{E})$ for $v \in V$:

- the out-degree of $v$ is the number of directed edges with $v$ as the first component.
- the in-degree of $v$ is the number of directed edges with $v$ as the second component.

In a directed graph, we say that two vertices $A_i$ and $A_j$ are connected (or adjacent), if at least one of the edges $(A_i, A_j)$ or $(A_j, A_i)$ are in $\overrightarrow{E}$.

**Definition 3.** A subgraph of a graph $G$ is the graph obtained by removing some vertices and edges from $G$ such that for all the remaining edges their vertices are not removed.

**Definition 4.** A path is a graph that can be represented as a sequence of its vertices such that all consecutive vertices are adjacent, all vertices except the first in the sequence have in-degree 1, and all vertices except the last in the sequence have out-degree 1.

**Definition 5.** A cycle is a closed path. Equivalently, a cycle can be represented as a sequence of its vertices such that all consecutive vertices are adjacent and all vertices have both in-degree and out-degree 1.

We can display the graph of all algorithms in a way that all the edges are directed downward. The constructed graph with downward edges can be seen as a union of its connected components. Besides, by adding virtual vertices 0 and 1 to each of connected components of the graph with vertex 1 on the top of the first layer with edges from it to all vertices in the first layer and the vertex 0 on the bottom of the last layer with edges from all vertices in the last layer to it, this process will transform the graph to a union of semi-lattices, denote by $\mathcal{SL}(G)$. We abuse notation slightly and denote the virtual vertices of all of the connected components of the graph by 0 and 1.

Denote by $\text{ExecutionFlows}(G)$ the set of all execution flows from 1 to 0 in $\mathcal{SL}(G)$, where an execution flow is a subgraph of $\mathcal{SL}(G)$ which is a path.

Assume that a finite set of robots $\mathcal{R} = \{R_1, \ldots, R_n\}$ and the robots are designed to perform a dynamic sequence of tasks from a finite set $\mathcal{T} = \{T_1, \ldots, T_m\}$. For the set of tasks, we also consider a
task which we call forced Idle task, which is used later when a robot is waiting for a specific amount of time. We try to solve the dynamic task allocation problem for a robotic network with robots with scheduled tasks. From now on by $A|_B$ we mean a restriction of $A$ to $B$.

What we are planning to do is to translate each new task $T \in \mathcal{T}$ to a subset, $S_R(T)$, of a hyperspace $S$ with respect to the robot $R$, and each robot $R \in \mathcal{R}$ to a subset, $(S_T(R), t^R_s(T))$ of a hyperspace $S \times \mathbb{R}$, where $t^R_s(T)$ is the time that the robot $R$ starts to perform the task $T$. And then the task $T$ will be scheduled for the robot $R$, if the value of the operator

$$SS(T, R) = \int_{S_T(R) \cap S_R(T)} dS$$

(which is the size of the subspace $S_T(R) \cap S_R(T)$ of $S$) is maximum and the following conditions satisfying: For the robot $R$, let for a task $T$, $t^R_s(T)$ denotes the completion time of the task $T$ and let $T^R_1, \ldots, T^R_k$ be the sequence of tasks, with the same order, scheduled for the robot $R$, then:

- Performing the task $T$ does not affect any elements of the set

  $$\{t^R_s(T^R_1), \ldots, t^R_s(T^R_k)\}.$$

- Performing the task $T$ affects

  $$\{t^R_s(T^R_{\alpha_1}), \ldots, t^R_s(T^R_{\alpha_l})\} \subseteq \{t^R_s(T^R_1), \ldots, t^R_s(T^R_k)\},$$

  where

  $$1 \leq \alpha_1 \leq \ldots \leq \alpha_l \leq k,$$

  and change their start time respectively to

  $$\text{new} J^R_s(T^R_{\alpha_i}), \ldots, \text{new} J^R_s(T^R_{\alpha_l}).$$

Note that, the preceding changes cause

$$\text{new} J^R_s(T^R_{\alpha_i}) > t^R_s(T^R_{\alpha_i}), \quad i = 1, \ldots, l.$$

Then for all $i = 1, \ldots, l$, considering

$$(\text{new} J^R_s(T^R_{\alpha_i}), (S_{T^R_{\alpha_i}}, R), t^R_s(T^R_{\alpha_i})),$$

and $S_R(T^R_{\alpha_i})$, where $\text{new} J^R_s(T^R_{\alpha_i})$ is the new subset of the hyperspace $S \times \mathbb{R}$ which is changed and the change is caused by performing the task $T$, we have one of the following conditions:

- The robot $R$ is such that, for all $i$:

  $$\int_{\text{new} S_{T^R_{\alpha_i}}(R) \cap S_R(T^R_{\alpha_i})} dS \geq SS(T^R_{\alpha_i}, R).$$

- Let the robot $R$ is such that, for some $i$:

  $$\int_{\text{new} S_{T^R_{\alpha_i}}(R) \cap S_R(T^R_{\alpha_i})} dS < SS(T^R_{\alpha_i}, R). \quad (1)$$
Denote by $NV_R$ the set of all $i$ such that the inequality (1) holds. Then the robot $R$ has the property that for any robot $B \in R \setminus \{R\}$, the inequality (2)

$$
\sum_{i \in NV_R} \left( SS(T_{\alpha_i}, R) - \int_{new,S_{T_{\alpha_i}}(R) \cap S_R(T_{\alpha_i})} dS \right) < \sum_{j \in NV_B} \left( SS(T_{\beta_j}, B) - \int_{new,S_{T_{\beta_j}}(B) \cap S_B(T_{\beta_j})} dS \right)
$$

holds, where the index $\beta$ is to identify the tasks scheduled in $B$ that allocation of task $T$ change them.

Now we describe the hyperspace $S$. The hyperspace can be obtained by numerical evaluation of tasks and specifications of robots.

We first provide a simple example of subspace of the hyperspace regarding only whether a robot node is compatible with a task (whether it is able to perform it) or not.

Let $X$ be a subspace of $S$ then $SS(T, R) |_X$ means restriction of the operator to the subspace $X$.

## 4 Compatibility

The compatibility subspace, denoted by $CMPT$, is the two element set $\{\varnothing, 1\}$ subspace of $S$. For a task $T$ and a robot $R$ define:

$$
S_T(R) |_{CMPT} = S_R(T) |_{CMPT}
$$

$$
= \begin{cases} 
1 & \text{, If the robot } R \text{ is compatible with the task } T, \\
\varnothing & \text{, Otherwise.}
\end{cases}
$$

If the robot $R$ is not compatible with the task $T$ (not able to perform that task), then

$$
S_T(R) |_{CMPT} = S_R(T) |_{CMPT} = \varnothing
$$

and

$$
SS(T, R) |_{CMPT} = 0.
$$

Now we provide more complex scenarios.

## 5 Communication instability

Let a task $T$ with the time windows $[a, b]$ is scheduled to be performed by a robot $R$. Assume that the robot $R$, to perform the task $T$, needs to communicate with the other robots several times. Denote by $CommTime_{R,R_i}(T)$ the total communication time between the robot $R$ and the robot $R_i$ while the robot $R$ is performing the task $T$. The number of times that the robot $R$ is communicating with the
robot $R_i$ is finite. Let $\text{CommTime}^j_{R,R_i}(T)$ be the minimum time that it takes for the robot $R$ to send a request to the robot $R_i$ and to receive a response for the $j$-th times while the robot $R$ is performing the tasks $T$. Because of communication instability, $\text{CommTime}^j_{R,R_i}(T)$’s for all $j = 1, \ldots, k_T$ is a positive value random variable. Note that,

$$0 \leq \text{CommTime}^i_{R,R_i}(T) = \sum_{j=1}^{k_T} \text{CommTime}^j_{R,R_i}(T) \leq b,$$

where $k_T$ is the total number of requests sent by the robot $R$ to the robot $R_i$ while performing the task $T$.

Then

$$\text{CommTime}_{R}(T) = \max\{\text{CommTime}^i_{R,R_i}(T) \mid i = 1, \ldots, n\},$$

is the maximum overall communication time between the robot $R$ and all the other robots to perform the task $T$. Note that

$$0 \leq \text{CommTime}_{R}(T) \leq b.$$

If $\text{CommTime}_{R}(T)$ is smaller, then the robot $R$ is more accessible by other robots to perform the task $T$. Hence, the task $T$ needs to be allocated to a robot that is more accessible by other robots. Note that, in case the deadline is not provided, then $b = \infty$.

Define

$$\text{InvCommTime}_{R}(T) = \frac{1}{\text{CommTime}_{R}(T)}$$

to be the inverse communication time between the robot $R$ and all the other robots to perform the task $T$. By construction, the value of $\text{InvCommTime}_{R}(T)$ is non-negative. Note that, if $\text{CommTime}_{R}(T) \to \infty$ then $\text{InvCommTime}_{R}(T) \to 0$. Also note that, if $\text{CommTime}_{R}(T) = 0$, which means that the robot $R$ does not communicate with the other robots to perform the task $T$, then $\text{InvCommTime}_{R}(T) = \infty$, which means that

$$\text{CommTime}_{R}(T) \to 0 \implies \text{InvCommTime}_{R}(T) \to \infty$$

continuously. Hence, maximizing $\text{InvCommTime}$ is equivalent to minimizing $\text{CommTime}_{R}(T)$.

Let the communication subspace of $S$, denote by $\text{Communication}$, be be the non-negative real line with infinity, $\mathbb{R}_{\geq 0} \cup \{\infty\}$. Let

$$S_R(T) |_{\text{Communication}} = \mathbb{R}_{\geq 0} \cup \{\infty\},$$

and let

$$S_T(R) |_{\text{Communication}} = [0, \text{InvCommTime}_{R}(T)].$$

Hence,

$$S_T(R) |_{\text{Communication}} \cap S_R(T) |_{\text{Communication}} = [0, \text{InvCommTime}_{R}(T)].$$

We have then

$$SS(T, R) |_{\text{Communication}} = \text{InvCommTime}_{R}(T).$$
Instability Distribution:

Here, we explore the communication instability between neighbour robots. We assumed that \( \text{CommTime}^{j}_{R,R_i}(T) \) is a positive random variable for each \( j \). Now assume that the robots \( R \) and \( R_i \) are neighbours and

\[
\text{CommTime}^{j}_{R,R_i}(T) = C(T \mid R, R_i) + \varepsilon^j_{R,R_i},
\]

where \( C(T \mid R, R_i) \) is a constant equal to the minimum communication time between the neighbour robots \( R \) and \( R_i \) and \( \varepsilon^j_{R,R_i} \) is the non-negative random variable delay and we are assuming

\[
\varepsilon^j(R, R_i) \sim \text{Exponential}(\lambda_{R,R_i}).
\]

Note that, we are assuming that the delay variables \( \varepsilon^j_{R,R_i} \) are independent for all \( j \)’s. Hence, for two neighbouring robots:

\[
\text{CommTime}^{j}_{R,R_i}(T) = 2kT C(T \mid R, R_i) + \sum_{j=1}^{2kT} \varepsilon^j_{R,R_i},
\]

where, by [12],

\[
\sum_{j=1}^{2kT} \varepsilon^j_{R,R_i} \sim \text{Erlang}(2kT, \lambda_{R,R_i}).
\]

Now assume that \( \lambda \) be a robot which is not a neighbour of the robot \( R \), then there is a path from \( R \) to \( \lambda \) through robots \( B_1, \ldots, B_k \) such that the robots \( B_u \) and \( B_{u+1} \) are neighbours for \( u = 0, \ldots, k \).

To enforce uniform notation for the path from \( R \) to \( \lambda \), let \( B_{k+1} = \lambda \) and \( B_0 = R \). Then

\[
\text{CommTime}^{j}_{B_0,B_{k+1}}(T) = 2kT \sum_{u=0}^{k} C(T \mid B_u, B_{u+1}) + \sum_{u=0}^{k} \sum_{j=1}^{2kT} \varepsilon^j_{B_u,B_{u+1}},
\]

where

\[
\sum_{u=0}^{k} \sum_{j=1}^{2kT} \varepsilon^j_{B_u,B_{u+1}}
\]

is the sum of \( k+1 \) independent random variables with Erlang distributions with parameters \( 2kT \) and \( \lambda_{B_u,B_{u+1}} \) for \( u = 0, \ldots, k \).

Note that in case, \( \lambda = \lambda_{B_u,B_{u+1}} \)’s for all \( u \), then

\[
\text{CommTime}^{j}_{B_0,B_{k+1}}(T) = 2kT \sum_{u=0}^{k} C(T \mid B_u, B_{u+1}) + \varepsilon,
\]

where \( \varepsilon \) is a random variables with Erlang distribution with parameters \( 2(k+1)kT \) and \( \lambda \).

Note that, communication between robots \( B_0 \) and \( B_{k+1} \) while the robot \( B_0 \) is performing the task \( T \), will be made through the path with smallest communication time.

6 Capabilities of fog, cloud, and robots

Each task in \( T = \{T_1, \ldots, T_m\} \) can be performed by executing some smaller computational tasks (algorithms) in \( A = \{A_1, \ldots, A_k\} \), where each task \( T_i \) can be performed by executing algorithms

\[
A \mid T_i = \{A^i_1, \ldots, A^i_{k_i}\}.
\]
and \( A = \cup_{i=1}^{n_1} A \mid T_i \). \( A \) is the set of all algorithms that are necessary to perform all the tasks. Each computational task can be executed on the robots, fog or the cloud nodes. The algorithm’s dependencies can be drawn as a graph of algorithms, see [1].

Following the notation in [1], virtual algorithms are \( 1_{T_i} \) and \( 0_{T_i} \) in the semi-lattice transformation of the graph of algorithms \( G(A \mid T_i) \), \( SL(G(A \mid T_i)) \). Let

\[
B = [0, 1]^{kn_1(f+c)},
\]

where \( n \) is the number of robots, \( f \) is the number of fog nodes and \( c \) is the number of cloud nodes and \([0, 1] \subset \mathbb{R} \) is the closed interval between zero and one on the real line.

Note that, \( n + f + c \) is the set of all nodes in the architecture, and at a fixed time \( t \), let

\[
\pi^t : A \rightarrow B
\]

be a mapping of probabilities such that for every \( j = 1, \ldots, k, \pi \) maps \( A_j \in A \) to a \( k \times (n + f + c) \) matrix such that all components except on the \( j \)-th row of the matrix are zero, \( \sum_i (\pi^t(A_j))_{i,j} = 1^1 \), and if \((\pi^t(A_j))_{i,j} > 0\), then the algorithm \( A_j \) can be executed by the node \( i \). The mapping \( \pi \) shows all the probabilities of allocating all the algorithms to all nodes.

Assume that at the time \( t \) some changes appeared in the architecture. The mapping \( \pi^t \) is regardless of where all the algorithms are allocated, but when the time \( t \) changes, the mapping \( \pi^t \) may change, for example the case that some disturbances occur and so the algorithm allocation must dynamically change according to the highest probabilities. However, it has to be determined how to control the highest probability such that the optimal performance is preserved (minimum overall execution time and minimum memory usage by robots).

Let \( \pi^t \rightarrow \pi^{t+\delta} \), where \( \delta > 0 \) is a very small real number close to zero, then we will find the weight map \( \omega^{t\rightarrow t+\delta} \) where

\[
\pi^{t+\delta} = \pi^t + \omega^{t\rightarrow t+\delta},
\]

\( \pi^{t+\delta} \) has the same property as \( \pi^t \) has, and rows of \( \omega^{t\rightarrow t+\delta} \) have zero sums.

Let \( \Omega \) be the set of all mapping \( \pi^t : A \rightarrow B \) with the preceding conditions at any arbitrary time. We also assume that if a node \( i \) is not capable of executing an algorithm \( A_j \), then for all \( \pi^t \in \Omega \), \((\pi^t(A_j))_{i,j} = 0 \).

As we said, the dynamic of mapping is \( \pi^t \rightarrow \pi^{t+\delta} \), for \( t \rightarrow t + \delta \). Now \( t \rightarrow \infty \) means the current time or the limit of \( \pi^t \) for \( t \rightarrow \infty \) if there are some pattern for \( \omega^{t\rightarrow t+p\delta} \) for some \( p \) large enough, as a sequence of time segments.

Let \( \mathcal{N} \) be the set of all nodes in the architecture of a robotic network cloud system and \( |\mathcal{N}| = n + f + c \) and

\[
\text{idx} : \mathcal{N} \rightarrow \{1, \ldots, n + f + c\} \subset \mathbb{N}
\]

is the one-to-one and onto mapping that identifies the index of robot, fog, and cloud nodes in the set \( \mathcal{N} \).

Also,

\[
\alpha(j, \pi^t) = \left\{ i \mid (\pi^t(A_j))_{i,j} = \max_x (\pi^t(A_j)_{i,x}) \right\}.
\]

By considering \( \pi^t \) as follows:

\[
(\pi^t(A_j))_{i,j} = c_j \times a_1 \times a_2,
\]

\(^1\)This means that the algorithm \( A_j \) with probability 1 allocated to one of the nodes.
where \( c_j \) is a constant,

\[
a_1 = 1 - \left\{ \begin{array}{ll}
\frac{\text{ExecutionTime}_i(A_j)}{\sum \text{ExecutionTime}_i(A_j)} , & \text{ExecutionTime}_i(A_j) \neq 0 \\
0 , & \text{ExecutionTime}_i(A_j) = 0
\end{array} \right.
\]

where \( \text{ExecutionTime}_i(A_j) \) is the average execution time of the algorithm \( A_j \) if it is executed on the node \( i \), and by letting

\[
k(i) = \sum_{A_v \in \text{EF}} (\text{CT}(\text{idx}^{-1}(i), \text{idx}^{-1}(v))),
\]

where

\[
\text{EF} = \text{ExecutionFlows}(G) | A_j
\]

and

\[
\text{CT} = \text{CommTime}^1,
\]

the term \( a_2 \) can be obtained by Equation (3).

\[
a_2 = 1 - \left\{ \begin{array}{ll}
\frac{k(i)}{\sum \kappa(u)} , & k(i) \neq 0 \\
0 , & \text{otherwise}
\end{array} \right.
\]

The constant \( c_j \) is used for providing the equality \( \sum_i (\pi^t(A_j))_{i,j} = 1 \).

Assume that the probabilities of assigning \( A_j \) to nodes change recursively according to the algorithm’s order,

\[
(\pi^t(A_j))_{i,j} \rightarrow (\pi^t(A_j))_{i,j} + (\omega^{i-t+\delta}(A_j))_{i,j},
\]

where the term \( (\omega^{i-t+\delta}(A_j))_{i,j} \) can be estimated by (4),

\[
(\omega^{i-t+\delta}(A_j))_{i,j} \propto \frac{pr_i(A_j)}{\sum_{A_v \in \text{EF}} (\text{CT}(\text{idx}^{-1}(i), \text{idx}^{-1}(v)))},
\]

where \( pr_i(A_j) \) is the processing power of node \( i \), and the term in the denominators of (3) and (4) means the total communication time between the node \( i \), and the nodes that need the output of the algorithm \( A_j \) to execute their algorithms, \( \alpha(v, \pi^t) \) is the node with the highest probability in \( \pi^t \) at the column \( v \), \( \text{EF} \) means the set of execution flows where \( A_j \) is one of its algorithms, and

\[
A_v < \text{EF}
\]

means the algorithm \( A_v \) has appeared before the algorithm \( A_j \) in an execution flow.

This means that the set \( \Omega \) has a relationship with the compatibility subspace, that is, for a node \( i \) and algorithm \( A_j \), the node \( i \) is compatible to execute the algorithm \( A_j \), if and only if,

\[
\left( \sum_{\pi^t \in \Omega} \pi^t(A_j) \right)_{i,j} > 0,
\]

i.e., for at least one mapping \( \pi^t \in \Omega \), \( (\pi^t(A_j))_{i,j} > 0 \) (the \( i \)-th row and the \( j \)-th column of the matrix \( \pi^t(A_j) \) is non-zero). If we change capabilities of any nodes to execute an algorithm, then the set \( \Omega \) will change accordingly.

\[\text{It is equivalent to } \frac{1}{\text{ExecutionTime}_i(A_j)} \text{ which is the average execution time for node } i \text{ to execute the algorithm } A_j.\]
Define the function
\[ \Pi : \mathcal{A} \times \mathcal{N} \rightarrow [0,1], \]
with
\[ \Pi(A_i,j) = \begin{cases} \max_{i,\ell \sim \mathcal{w},\pi' \in \Omega} \left( \pi'(A_j) \right)_{i,j}, & \text{If } \left( \sum_{\pi' \in \Omega} \pi'(A_j) \right)_{i,j} > 0, \\ 0, & \text{Otherwise.} \end{cases} \]
Then
\[ \mathcal{N} \setminus \Pi^{-1}(0) \mid_{\langle A_i,j \rangle} \]
shows the set of all nodes which are capable of executing the algorithm \( A_i \) with the highest probability.

Note that, virtual algorithms \( 0_T \) and \( 1_T \) for \( i = 1, \ldots, m \), are considered as elements of \( \mathcal{A} \). Now, the robot \( R \) is capable of performing a task \( T \), if and only if,
\[ \Pi(0_T, idx(R)) = \Pi(1_T, idx(R)) > 0. \]
Here, by \( \Pi(0_T, idx(R)) = 0 \), we mean that the robot \( R \) is not capable of starting the task \( T \) and by \( \Pi(1_T, idx(R)) = 0 \) we mean that the robot \( R \) is not capable of finishing the task \( T \).

With the preceding notation, the capability subspace denote by \( CPLT \) is the \([0,1]^{\mathcal{A} \times \mathcal{N}}\) subspace of \( S \). For a task \( T \) and a robot \( R \) define
\[ S_T(R) \mid_{CPLT} = S_T(R) \mid_{CPLT} = \Pi(\mathcal{A} \mid_T, \mathcal{N}) \times \Pi(1_T, idx(R)) \times \Pi(0_T, idx(R)). \]
This shows what are the highest probabilities of allocating algorithms of the task \( T \) to any nodes and also if the value of the function is the zero matrix, then this means that the robot \( R \) is not capable of performing the task \( T \). Now,
\[ SS(T, R) \mid_{CPLT} = \Pi(1_T, idx(R)) \times \Pi(0_T, idx(R)) \times \int \Pi(\mathcal{A} \mid_T, \mathcal{N}), \]
and since the sums of each row of \( \Pi(\mathcal{A} \mid_T, \mathcal{N}) \) are 1, it is an all 1 vector. So, we can drop the integration. Therefore,
\[ SS(T, R) \mid_{CPLT} = \Pi(1_T, idx(R)) \times \Pi(0_T, idx(R)). \]
Therefore, the task \( T \) will be allocated to a robot, if the preceding integration has the highest values for that robot. This is because the highest integration value means the robot has the highest probability for start and to finish the task at a certain time.

In addition, since the function \( \Pi \) dynamically changes by time, allocating new tasks depends on the dynamics of past task allocation and the dynamics of the system changes.

If we use the dynamic data transmission matrix, \( DT \) multiplied by
\[ AD^1, \ldots, AD^l, \]
where \( l \) is the length of the maximum execution flow and, \( AD \) is the adjacency matrix of the graph of algorithms, then we can obtain the maximum overall communication time for any algorithm allocation. Then, using similarly the execution time matrix and the \( 0,1 \) matrix induced by the matrix \( \Pi \) with the highest values component, that gives us the overall execution time for a given algorithm allocation.
Figure 2: Three robots cloud system. The numbers on the edges are the data transmission times, where $\varepsilon_i$'s are random variables from exponential distributions with parameter $\lambda_i$.

Table 1: The average execution time of the algorithm nodes in the cloud system’s nodes. The average execution time of the virtual algorithms 0 and 1 in the cloud system’s nodes are considered 0.

|        | Data | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ | $A_7$ | $A_8$ |
|--------|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Robot $R_1$ | 0    | 2     | 6     | 4     | 2     | 6     | 4     | 2     | 6     |
| Robot $R_2$ | 0    | 2     | 6     | 4     | 2     | 6     | 4     | 2     | 6     |
| Robot $R_3$ | 0    | 2     | 6     | 4     | 2     | 6     | 4     | 2     | 6     |
| Fog $F$     | 0    | 1     | 3     | 2     | 1     | 3     | 2     | 1     | 3     |
| Cloud $C$   | 0    | 0.5   | 1.5   | 1     | 0.5   | 1.5   | 1     | 0.5   | 1.5   |

The matrix $DT$ can be transformed into a matrix induced by the highest values components of $\Pi$ that allows us to find the maximum overall communication times for a given algorithm allocation. The main challenge is to control the highest probabilities.

**Remark 1.** Note that, the mapping $\pi^t$ is needed to have the condition that each of its rows has at least one non-zero component, otherwise, there is a node in the architecture that is superfluous.

7 Example.

We made a simulation for a cloud robotic system consisting of three robots, one fog server, and a cloud, see Figure 2. In Figure 2, we assume that $\varepsilon_i$’s are random variables from exponential distributions with parameter $\lambda_i$, with $\lambda_1 = 2$, $\lambda_2 = \lambda_3 = 4$, and $\lambda_4 = 8$. We consider the newly arrived task $T$ with the corresponding graph of the algorithms which can be seen in Figure 3. The average execution time of algorithms in each node is shown in Table 1. Assume that the algorithms are allocated to the cloud system’s nodes according to Table 2. Table 2, implies that the number of requests sent to the fog node by each robot is 2 and the number of requests sent by each robot to the cloud node is 7, where we obtain:

$$S_T(R_1) |_{\text{Communication}} = [0, 0.00266],$$
Figure 3: Graph of algorithms for the newly arrived task.

Table 2: The algorithms allocation.

| Edge | Fog       | Cloud |
|------|-----------|-------|
|      | $R_1$ $R_2$ $R_3$ | $A_3$ and $A_7$ Data, $A_1$, $A_2$, $A_4$, $A_5$, $A_6$, and $A_8$ |
This means that the task will be scheduled to the robot $R_3$ considering only the Communication subspace.

Now, if we assume that the robot $R_3$ is not compatible with the task, then

$$SS(T, R_3) |_{\text{CMPT}} = 0,$$

and

$$SS(T, R_1) |_{\text{CMPT}} = SS(T, R_2) |_{\text{CMPT}} = 1.$$  

Hence, 

$$SS(T, R_1) |_{\text{CMPT}, \text{Communication}} = 0.00266,$$

$$SS(T, R_2) |_{\text{CMPT}, \text{Communication}} = 0.00407,$$

and

$$SS(T, R_3) |_{\text{CMPT}, \text{Communication}} = 0.$$  

This means that the task will be scheduled to the robot $R_2$ considering both of the Communication and CMPT subspaces.

Also,

$$\Pi(1_T, \text{idx}(R)) = c_1 \times \prod_{\text{allnodes}}^{a_1} (1, 1, 1, 1, 1)$$

$$\Pi(0_T, \text{idx}(R)) = c_{11} \times \prod_{\text{allnodes}}^{a_1} (0.65, 0.81, 0.71, 0.91, 0.92)$$

where $c_1 = 0.2$ and $c_{11} = 0.385$. Therefore,

$$\Pi(\_ , \text{idx}(R)) = \left\{ \begin{array}{l}
R_1 \quad R_2 \quad R_3 \quad F \quad C \\
0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \\
0.163 \quad 0.203 \quad 0.178 \quad 0.228 \quad 0.23 \\
\end{array} \right\}, 1_T$$

$$\Pi(\_ , \text{idx}(R)) = \left\{ \begin{array}{l}
R_1 \quad R_2 \quad R_3 \quad F \quad C \\
0.033 \quad 0.041 \quad 0.036 \quad 0.046 \quad 0.046 \\
\end{array} \right\}, 0_T$$

which implies

$$SS(T, \_ ) |_{\text{CPLT}} = (0.033, 0.041, 0.036, 0.046, 0.046),$$

This means that the task will be uploaded to the cloud or to the fog, if possible, otherwise, the task will be scheduled to the robot $R_2$.

Now altogether,

$$SS(T, R_1) |_{\text{CMPT} \cup \text{Communication} \cup \text{CPLT}} = 8.78e-5,$$

$$SS(T, R_2) |_{\text{CMPT} \cup \text{Communication} \cup \text{CPLT}} = 1.67e-4,$$

and

$$SS(T, R_3) |_{\text{CMPT} \cup \text{Communication} \cup \text{CPLT}} = 0.$$  

In the example, we assumed that all the algorithms of the task $T$ are allocated to the cloud and the
fog, mainly to the cloud. Now the main factor for scheduling the task $T$ is the overall communication time of nodes that need to be minimized. The overall communication time will be minimized if the task will be uploaded to the node with the smallest communication distance to the nodes where algorithms of the task $T$ are allocated to, wherein the example the natural solution is to upload the task to the cloud. However, if the tasks can only be assigned to a (robot) node, then the natural solution is to schedule the task to the robot(s) with the smallest communication distance to the fog and the cloud where all the resources are allocated. Hence, the natural choices of the robots to schedule the task $T$ is either $R_2$ or $R_3$ in view of the architecture, Figure 2. But now since we assume that the robot $R_3$ is not compatible with the task $T$, the robot $R_2$ is the most suitable choice. Applying the proposed method confirms this natural choice.

8 Conclusion

In our proposed method to use hyperspace with robots and tasks as subspaces, we provided solutions for task allocation under a single (important) factor at a time. For each factor, we provide a proper optimal solution for task allocation. By considering more than a single factor ($F_1, \ldots, F_k$), an optimal solution for task allocation can be obtained by finding the maximum value of the

$$SS(T, R) | F_1 \cup \ldots \cup F_k .$$

By using the proposed hyperspace, instead of formulating an optimization problem and then finding a solution for it, we provide directly the solution by comparing hyper volumes generated by the intersection of the relative task space and each of the relative robot space. If the volume is largest, then the task will be allocated to that robot.

We can extend this theoretical framework and apply it to time windows of tasks, minimizing makespan, task modification, change the number of nodes, and so on. In addition, this theoretical framework can be classified as both a centralized and distributed model, that is, either each node finds the size of its hyperspace with respect to its current status, and the given task or a centralized node does that.

We proposed a new framework based on theoretical modeling for task allocation. We studied dynamic task allocation optimizing several factors but independently and for task arrival as a single task at a time that can be extended easily to several tasks at a time. For future work, we will extend the method to more factors and provide real-world tests.

References

[1] Saeid Alirezazadeh and Luís A. Alexandre. Optimal algorithm allocation for single robot cloud systems, 2020.

[2] Muhammad Usman Arif. An evolutionary algorithm based framework for task allocation in multi-robot teams. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI-17, pages 5032–5033. AAAI Press, 2017.

[3] J. A. Bondy and U. S. R. Murty. Graph theory, volume 244 of Graduate Texts in Mathematics. Springer, New York, 2008.
[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC-12, pages 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[5] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. Assignment Problems. Society for Industrial and Applied Mathematics, 2012.

[6] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, and K. Nakamura. Qos-aware robotic streaming workflow allocation in cloud robotics systems. IEEE Transactions on Services Computing, pages 1–14, 2018.

[7] Z. Cheng, P. Li, J. Wang, and S. Guo. Just-in-time code offloading for wearable computing. IEEE Transactions on Emerging Topics in Computing, 3(1):74–83, 2015.

[8] M. C. Gombolay, R. J. Wilcox, and J. A. Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. IEEE Transactions on Robotics, 34(1):220–239, 2018.

[9] Matthew Gombolay, Ronald Wilcox, and Julie Shah. Fast scheduling of multi-robot teams with temporospatial constraints, 2013.

[10] Jun He, Mohamed Badreldin, Ahmed Hussein, and Alaa Khamis. A comparative study between optimization and market-based approaches to multi-robot task allocation. Advances in Artificial Intelligence, 2013:256524, 2013.

[11] G. Hu, W. P. Tay, and Y. Wen. Cloud robotics: architecture, challenges and applications. IEEE Network, 26(3):21–28, May 2012.

[12] Oliver C. Ibe. 1 - basic concepts in probability. In Oliver C. Ibe, editor, Markov Processes for Stochastic Modeling, pages 1 – 27. Elsevier, Oxford, 2 edition, 2013.

[13] I International Federation of Robotics IFR worldwide. International Federation of Robotics IFR, industrial robots, 2020.

[14] Anil K Jain and Stan Z Li. Handbook of face recognition, volume 1. Springer, 2011.

[15] Frederick Jelinek. Statistical methods for speech recognition. MIT press, 1997.

[16] K. Kamei, S. Nishio, N. Hagita, and M. Sato. Cloud networked robotics. IEEE Network, 26(3):28–34, May 2012.

[17] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering, 12(2):398–409, April 2015.

[18] S. Li, Z. Zheng, W. Chen, Z. Zheng, and J. Wang. Latency-aware task assignment and scheduling in collaborative cloud robotic systems. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pages 65–72, July 2018.

[19] Gerard McKee. What is Networked Robotics?, pages 35–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
[20] Gerard T. McKee and Paul S. Schenker. Networked robotics. In Gerard T. McKee and Paul S. Schenker, editors, Sensor Fusion and Decentralized Control in Robotic Systems III, volume 4196, pages 197 – 209. International Society for Optics and Photonics, SPIE, 2000.

[21] Nathan Michael, Shaojie Shen, Kartik Mohta, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. Collaborative Mapping of an Earthquake Damaged Building via Ground and Aerial Robots, pages 33–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[22] Vishnu Nath and Stephen E. Levinson. Autonomous Military Robotics. Springer Publishing Company, Incorporated, 2014.

[23] Hisashi Osumi. Application of robot technologies to the disaster sites. Report of JSME Research Committee on the Great East Japan Earthquake Disaster, pages 58–74, 2014.

[24] L. E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation, 14(2):220–240, 1998.

[25] Erwin Prassler and Kazuhiro Kosuge. Domestic Robotics, pages 1253–1281. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[26] Larry T. Ross, Stephen W. Fardo, and Michael F. Walach. Industrial Robotics Fundamentals: Theory and Applications. Goodheart-Willcox Co., USA, 3rd edition, 2017.

[27] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5):637–646, Oct 2016.

[28] Bruno Siciliano and Oussama Khatib. Springer Handbook of Robotics. Springer Publishing Company, Incorporated, 2nd edition, 2016.

[29] P.J. Springer. Military Robots and Drones: A Reference Handbook. Contemporary world issues. ABC-CLIO, 2013.

[30] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, and N. Hagita. Building knowledge-enabled cloud robotics applications using the ubiquitous network robot platform. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5716–5721, Nov 2013.

[31] N. Tsiogkas and D. M. Lane. An evolutionary algorithm for online, resource-constrained, multi-vehicle sensing mission planning. IEEE Robotics and Automation Letters, 3(2):1199–1206, 2018.

[32] L. Wang, M. Liu, and M. Q. . Meng. Hierarchical auction-based mechanism for real-time resource retrieval in cloud mobile robotic system. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 2164–2169, May 2014.

[33] L. Wang, M. Liu, and M. Q. . Meng. Real-time multisensor data retrieval for cloud robotic systems. IEEE Transactions on Automation Science and Engineering, 12(2):507–518, 2015.

[34] L. Wang, M. Liu, and M. Q. . Meng. A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems. IEEE Transactions on Cybernetics, 47(2):473–484, Feb 2017.

[35] Gang Xu and Zhengyou Zhang. Epipolar geometry in stereo, motion and object recognition: a unified approach, volume 6. Springer Science & Business Media, 2013.
[36] Y. Xu, H. Qian, and X. Wu. *Household Service Robotics*. Elsevier Science, 2014.