A Distributed Dynamic Framework to Allocate Collaborative Tasks Based on Capability Matching in Heterogeneous Multi-Robot Systems

Hoi-Yin Lee, Peng Zhou, *Member, IEEE*, Bin Zhang, Liuming Qiu, Bowen Fan, Anqing Duan, Jingtao Tang
Tin Lun Lam, *Senior Member, IEEE*, and David Navarro-Alarcon, *Senior Member, IEEE*

***Abstract*—Collaboration among a group of robots with heterogeneous capabilities is an important research problem that enables to combine different robot functionalities, and thus, conducts complex tasks that may be difficult to achieve by a single robot with limited resources. In this paper, we propose a new distributed task allocation framework based on the capability matching of heterogeneous robots. The framework is composed of an ontological dynamic knowledge graph model and a hardware control scheme to model the capability and optimize resource utilization for collaborative tasks. We introduce an intuitive hardware control scheme based on a dynamic knowledge graph that resolves possible conflicts between the hardware control of different types of robots. Action sequences are produced by a task and motion planning algorithm to collaboratively perform the assigned task. The performance of the proposed methodology is evaluated by both simulations and hardware experiments.**

***Index Terms*—Multi-Robot Systems; Task Allocation; Resource Allocation; Capability Modelling; Cognitive Systems.**

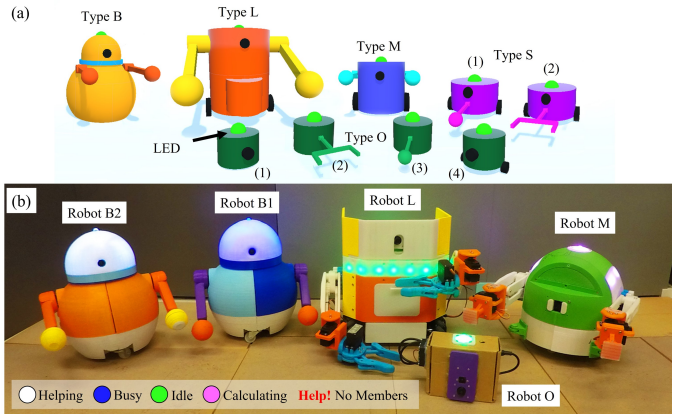

Fig. 1. Heterogeneous robots developed for (a) simulation (b) experiments in our multi-robots society. Robot B1 and B2 are non-assemblable and others are assemblable. All of them have different talents and can work collaboratively. The LED on their heads indicates the current state. Green: idle state, no job or task. Blue: busy state, job received. White: helping state, granting favor as a team-mate. In simulation: blue light with help: job received, no members and seeking favor.

I. Introduction

THE development of robot fleets that can autonomously assist humans in conducting productive tasks remains an open research problem that demands solutions in the pursuit of building the next generation of smart societies [1]. To reach this ambitious goal, effective collaboration is a key component that can enable groups of heterogeneous robots to improve productivity, as each individual robot can contribute with a specific skill to accomplish the task. Intuitively, a group of five robots (e.g., each with a different carrying capacity, sensing, and locomotion methods) may likely have better efficiency in loading goods than a single robot [2]. This principle of collaboration is naturally adopted by many human/animal groups, however, its effective implementation in multi-robot systems (MRS) is not trivial [3].

Various strategies have been developed to collaboratively perform tasks with MRS, e.g., some representative state-of-the-art methods include [4]–[9]. To optimize the performance of a group of heterogeneous robots, it is essential to develop methods that *characterize* and *exploit* the robots' distinct capabilities in their strategy. Various researchers have addressed this issue [10]–[13], e.g., the approach in [10] presents a capability model and a distributed task allocation algorithm based on auction theory. The suitability of a robot to work on a given task is determined by a metric that quantifies the strength level of the different robots' components. The method in [11] provides a multi-robot collaboration framework that adaptively allocates robots in a task based on their energy consumption. However, most existing approaches only focus on representing robot capabilities for task allocation (i.e., during the planning stage), and do not consider collaborative motions (e.g., forming teams and executing tasks) as part of their capability-based models. This critical issue has not been sufficiently studied in the literature.

Capability *enhancement* is another instrumental aspect of multi-robot collaboration. Robot teams which can reconfigure and augment their functions are expected to be more efficient than teams with a fixed skill set. In recent years, many researchers have developed various self-reconfigurable/self-assemblable systems [4], [6], [14]–[20], which have demonstrated high flexibility in their structural properties. The focus of most previous studies has been primarily on hardware modification, however, to algorithmically find an optimal capability enhancement strategy for MRS, we need to develop decision-making methods that incorporate interactions between heterogeneous units with their associated capabilities.

This work is supported in part by the Research Grants Council of Hong Kong under grant 15212721 and in part by the Jiangsu Industrial Technology Research Institute Collaborative Research Program Scheme under grant ZG9V.

H.-Y. Lee, P. Zhou, B. Zhang, L. Qiu, B. Fan, A. Duan and D. Navarro-Alarcon are with The Hong Kong Polytechnic University, Department of Mechanical Engineering, Kowloon, Hong Kong. (contact e-mail: dna@ieee.org)

J. Tang and T. L. Lam is with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, Guangdong, China.

TABLE I
COMPARISON WITH THE FEATURES AND FUNCTIONS IN STATE-OF-THE-ART METHODS FOR MRS.

Methods	Capability	Coll.	Hardware Utilization	Job Execution
[10]	✓	-	-	-
[11]	✓	-	-	✓
[5]	-	✓	-	✓
Ours	✓	✓	✓	✓

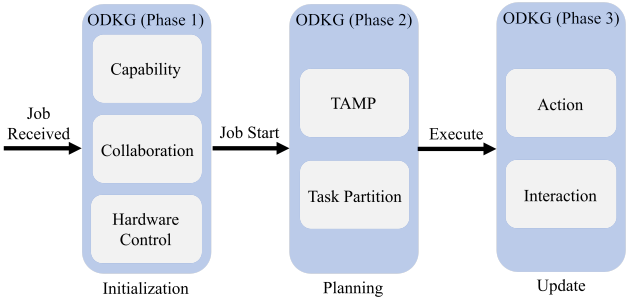

Fig. 2. The full structure and use of the proposed ODKG. phase 1: initialization of fundamental setup for collaboration: phase 2: task planning for multi-robots; phase 3: information update.

Although great progress has been recently achieved, more research needs to be conducted to develop these types of enhancement functionalities.

The development of *capability-oriented task allocation* strategies is another critical problem. Standard allocation methods have been used by many researchers [21]–[25], who have mostly focused on optimizing the task sequence and execution time, but not on allocating tasks based on the robots' capabilities (which is an under-explored problem). To automatically select different robot types and their roles in the collaborative task, an optimal allocation algorithm may need to consider the semantic relation between the robots, their functions, and the environment. This can be done by using a cognitive architecture graph [26]–[29], a model that can effectively capture this kind of robot-function-environment relationship. Despite its usefulness, cognitive graph methods have not been thoroughly used in task allocation problems involving heterogeneous MRS.

All these previous works have laid very good foundations for the analysis and control of collaborative MRS. However, there are two main (open) problems that hinder the development of task allocation methods for collaborative robots: (1) Difficulty to develop analytical capability models of different robot types in a consistent way; (2) Difficulty to allocate resources/capabilities to execute a collaborative motion task with a team composed of heterogeneous units. The former problem comes mainly due to the lack of a common representation method to characterize the capabilities of robotic systems. Efficient collaboration (e.g., to determine how a specific robot can contribute to the task) requires a shared skill representation framework. The latter problem is due to the fact that traditional approaches typically rely on centralized decision algorithms, which are computationally costly, and unfeasible to implement in heterogeneous systems.

There are many state-of-the-art approaches to facilitate the collaboration between multiple robots. We compare our methods with [5], [10], [11] (see Table I). The method in [10] is a strength-based approach that capability is characterized by its strength level. The decisions calculated by this method are only based on the binary existence of a given skill. Despite that the model in [10], [11] can represent the heterogeneity of the robot's capabilities, these methods cannot capture the merging/augmentation of different robots' capabilities after the assembly. These methods can optimally allocate tasks but do not optimize the utilization of hardware resources. As these methods cannot handle capability augmentation by MRS with heterogeneous capabilities, the available working modes are limited to the traditional approaches (individual or simple group work without optimizing resource utilization). A detailed job execution policy and cognitive knowledge sharing for collaborative works are also not fully studied. The tasks in [5] are divided into sub-tasks where robots make decisions based on Bayesian Delegation and model-based reinforcement learning. Although this method has a more comprehensive framework for collaborative task execution, the model does not consider the robots' capabilities.

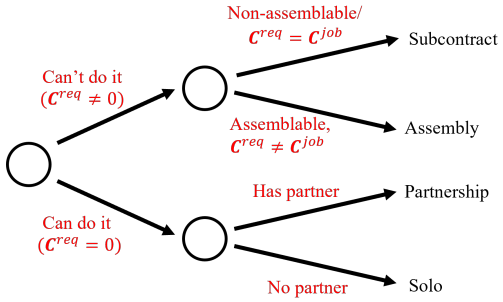

Fig. 3. Decision tree for different working modes. If a robot can do the job alone, then the working mode is either solo or partnership. If a robot cannot do the job alone and is non-assemblable, it can only apply subcontract mode. If it is assemblable, the mode is assembly or subcontract.

Our aim in this paper is to address these issues by developing a new distributed task allocation framework based on a semantic graph model. The proposed solution combines representation, enhancement and allocation methods of heterogeneous capabilities in robot teams. The original features of our new methodology are summarized as follows:

- a novel capability-based matrix masking method to represent heterogeneous robot configurations (thus, reducing hardware redundancy) and allocate their roles in a collaborative task;
- an effective method based on ontological dynamic knowledge graphs to model and locally optimize the motion and interaction between heterogeneous robots;
- a graph adjacency approach that uses the least cost path to optimally utilize and coordinate resources in a group of robots.

The rest of this paper is organized as follows: Sec. III, IV, and V describe the methodology; Sec. VII presents simulation and experimental results; Sec. VIII gives final discussions and conclusions.

II. PRELIMINARIES

A. Notation

Matrices and column vectors are denoted by bold letters, such as $\mathbf{M}$ and $\mathbf{m}$. We use $[\mathbf{M}]_{ij}$ to denote the entry at the i-th row and j-th column of a matrix $\mathbf{M}$, and $[\mathbf{v}]_i$ to denote the i-th element of a vector $\mathbf{v}$. Throughout the paper, the superscript $*^k$ is used to represent different instances of a model, e.g., $*^{self}$, $*^{job}$, $*^{req}$ represent the robot, job, and required versions of the structure $*$.

B. Definition of Terms

Roles: There are four roles in the proposed framework: 1) *Agent*: refers to the robot that receives the assigned job and which is responsible for completing it. 2) *Helper*: describes the robot that assists the *Agent* in completing the job. 3) *Team*: refers to various *Agents* and *Helpers* jointly working on the same job. A *Team* must contain one *Agent* and at least one *Helper*. 4) *Member*: Refers to any robots in a *Team*.

Capability: Refers to the configuration of a robot that is either pre-set in the factory or set after a hardware enhancement. In this paper, we use the matrix $\mathbf{C}$ with various superscripts to represent the capability of a system/problem, e.g., $\mathbf{C}^k$ is used for the general case, $\mathbf{C}^{self}$ for a robot's own, and $\mathbf{C}^{job}$ for the capability demanded by a specific job.

In our method, we assume the distinctive hardware components are categorized into different capability domains, e.g., defined based on their nature. For that, we use the following three general capability domains: perception, manipulation, and locomotion. For example, standard 2D vision sensors and infrared cameras can both be grouped into a perceptual domain while grippers belong to the manipulation domain.

Types of Collaboration: In our method, we consider four working modes (see Fig. 3) to accomplish a mission, namely, *Solo*, *Partnership*, *Assembly* and *Subcontract*. The *Solo* and *Partnership* modes are for situations where the capability of a single robot is sufficient for carrying out the job. When there is only one robot around, the robot will act as an *Agent* and work in a *Solo* mode. When there are other robots available, they can act as *Helpers* and form a *Team* with the *Agent* to collaborate and share the workload under the *Partnership* mode. The difference between *Solo* and *Partnership* modes is the number of robots involved. If there is only a single robot, then, it belongs to *Solo* mode. Otherwise, it is a *Partnership* mode and the workload is split equally.

The capability of a single robot may not be able to match the demands of the job. Therefore, the *Assembly* mode allows robots to create larger structures with enhanced capabilities that fulfil the requirements. In this mode, the robot receiving the job needs to be assemblable. For a non-assemblable robot, it can instead adopt the *Subcontract* mode and transfer the job to other robots. Neighbours who agree to help will then act in *Solo* mode. For example, if a robot has more than 30% of battery power and has no current job/task assignment, it is defined as an *idle robot*. If a helper is running out of battery or the agent loses communication with its helper for a period of time, then, the agent losses a helper and may require to find another robot to continue the task.

III. ONTOLOGICAL DYNAMIC KNOWLEDGE GRAPH

A. Graph Theory

Graphs are structures that model relations between different objects. We denote a graph with the following pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for $\mathcal{V} = \{v_1, \ldots, v_n\}$ as a set of vertices, $\mathcal{E} = \{v_i v_j | v_i, v_j \in \mathcal{V}; i, j = 1, \ldots, n\}$ as a set of edges, v_i as the i-th vertex, and n as the number of vertices. The neighborhood of v_i is defined as $\mathcal{N}_i = \{v_j \in \mathcal{V} | v_i v_j \in \mathcal{E}\}$ [30]. For an undirected graph $\mathcal{G}$, its degree matrix is defined as $\mathbf{D}(\mathcal{G}) = \mathrm{diag}[d(v_1), \ldots, d(v_n)]$, where $d(v_i)$ is the degree of the vertex v_i and equal to the cardinality of the neighborhood $\mathcal{N}_i$. The graph's adjacency matrix $\mathbf{A}(\mathcal{G}) \in \mathbb{R}^{n \times n}$ satisfies:

$$[\mathbf{A}]_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

With the degree matrix and the adjacency matrix, we compute the Laplacian matrix of a graph $\mathcal{G}$ as $\mathbf{L}(\mathcal{G}) = \mathbf{D} - \mathbf{A}$, which is symmetric and positive semi-definite.

B. Application of ODKG in MRS

MRS usually work in highly dynamic/uncertain conditions, where changes can be external (e.g., obstacles and temperature in the environment) or internal (e.g., intermittent communication among agents). To perform tasks in these situations, MRS may need to build and track all semantic relations between the variable elements. For that, we adopt a dynamic knowledge graph to construct this information [2], [31]. The purpose of this dynamic knowledge graph is to provide a framework that models the heterogeneity of capabilities and allocates resources efficiently in the various working modes.

An *ontology* is a collection of branches describing the nature of objects, their attributes, and the relation among them [32], [33]. Their structure can be modeled as a hierarchical graph. In our method, we propose to use an ontological dynamic knowledge graph (ODKG) to represent the semantic information of the MRS during collaborative object manipulation tasks, see Fig. 2. When the robot receives a job, it enters Phase 1, which requires matching the capability, locating suitable members, and generating a hardware control schema (i.e. Phase 1 in Fig. 2). Then, it starts planning the action sequence to complete the job as illustrated in Phase 2 of Fig. 2. The robot executes the plan and updates the ODKG with the latest perceived information as shown in Phase 3 in Fig. 2. This approach is based on the following assumptions: 1) all robots are autonomous and capable of coordination via minimal local communication; 2) all robots can act upon the latest environment changes to collaboratively reach the target.

The proposed ODKG is a two-layer hierarchical structure that ontologically stores the information of internal physical components and the external environment, and models the interaction between them [29], [33], see Fig. 4. All the elements in the ODKG can be updated, created, and deleted based on the present observation/interaction [33]. The ODKG is a digital twin of reality but expressed in a graphical-semantic model [28]. We model the problem with two graph layers representing the Robot Layer $\mathcal{G}^R$ and Job Layer $\mathcal{G}^{job}$, respectively. We use the case depicted in Fig. 4 to exemplify the structure of the proposed knowledge graph.

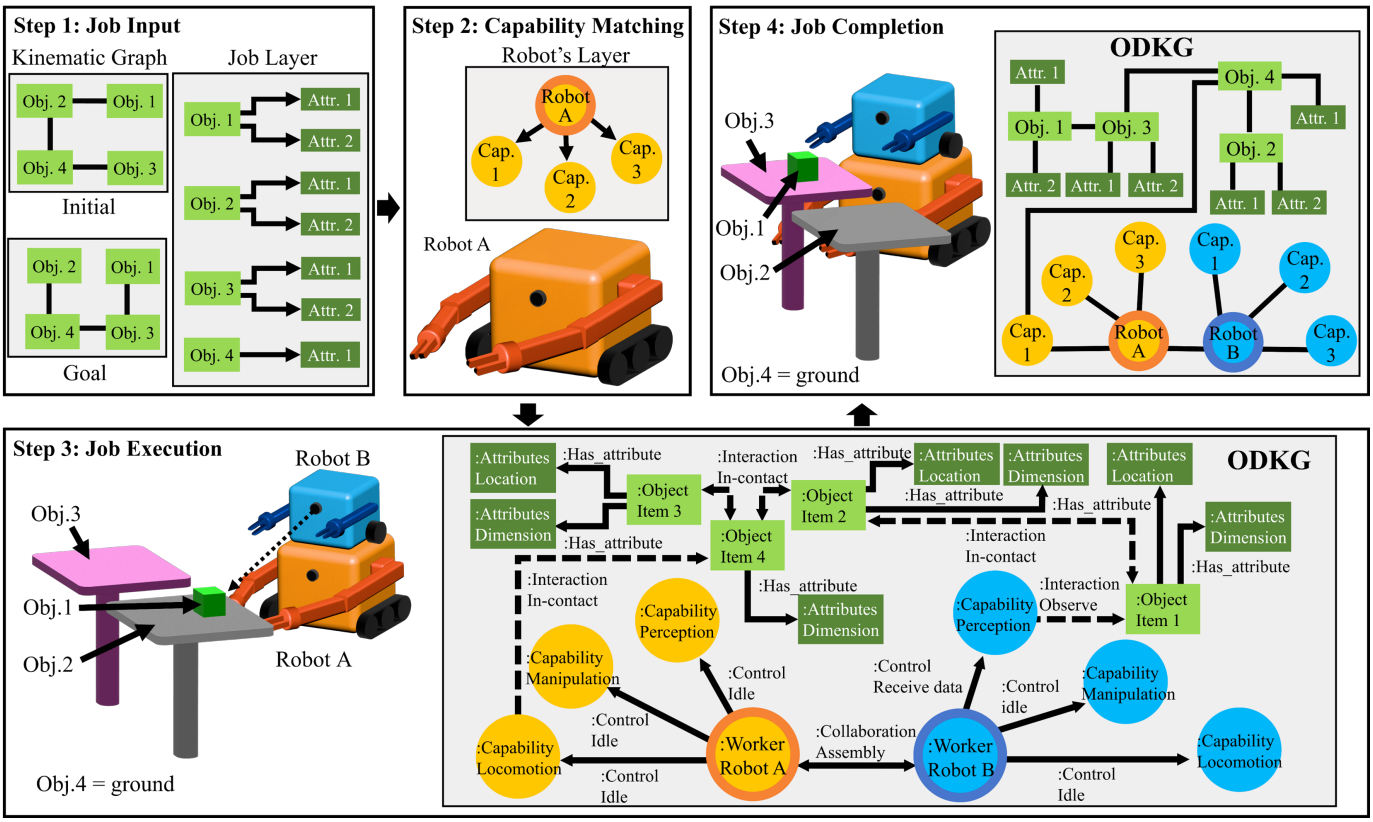

Fig. 4. Example of the implementation of ODKG in the MRS with a pick-and-place task. At the beginning (Step 1: Job Input), kinematic graphs and job layer graph are sent to the agent when assigning a job. Then, the agent performs capability verification in step 2 and collaborates with the optimal neighbor in Step 3. An ODKG is constructed for task execution and is composed of a Robot Layer and a Job Layer. A full ODKG is illustrated in step 3. During the task execution in steps 3-4, the ODKG evolves with the latest interaction information between robots and the environments. When the job is completed (e.g. an object is moved to the targeted table as illustrated in step 4), a connected ODKG is formed and the kinematic relationships between objects are identical to the goal stated in step 1. A simplified version of a connected ODKG is shown in 'Step 4: Job Completion'.

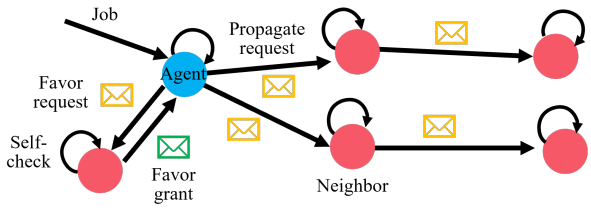

Fig. 5. Multi-worker systems with a job assignment. Each node represents a worker. The blue one denotes the agent. The yellow envelope indicates the favor request message. The green envelope indicates the favor granting reply message. Once a job is passed to a worker, a self-capability check is conducted before spreading the request to neighbors. If the neighbor cannot do it and has not time-out yet, it propagates the request to its neighbors.

Robot Layer. This layer represents the state of the MRS, i.e., the collaboration status of the robots and their components. It consists of the robots and their attributes, which are modelled as nodes in the graph $\mathcal{G}^R$. The attributes refer to the hardware components equipped in the robots. The collection of attributes defines the robot's specific capability. The relations between robots and between the robots and their attributes are represented as edges in the graph $\mathcal{G}^R$. We define the undirected edge between robots as a 'collaboration relation' and the directed edge between a robot and its attribute as a 'control relation'. An example of the Robot Layer is shown in Fig. 4.

Job Layer. The objects/items involved in a job as well as its attributes are included in this layer. Key information about the object (e.g., its dimension, weight, location) is encoded into the attributes. In the context of Task and Motion Planning (TAMP), this layer represents an extended kinematic graph that contains physical relationships between objects in the world and includes the necessary attributes of objects. The relation between objects can be changed by the robots (simply by interacting with them) and hence is defined as an 'interaction'. An attribute is linked to an object, and such relation is named 'has_attribute'. In Fig. 4, a simplified example of an individual job layer is shown in step 1 and a detailed job layer in an ODKG is shown in step 3.

Cross-Layer Relation. Connections between two items of different layers are defined as an 'interaction'. Physical interaction with an object is an 'in-contact' type of interaction while non-physical interaction is defined by its nature, such as 'observe' for perception.

Once a job is allocated to an agent, a kinematic graph of the current state $\mathcal{G}^{job}_{initial}$ and the final goal state $\mathcal{G}^{job}_{goal}$ with the required capability is passed to the agent, as illustrated in Fig. 5. Then, we combine $\mathcal{G}^R$ and $\mathcal{G}^{job}_{initial}$ to form the present state of the ODKG, see Fig. 4. Based on the job description, the agent then generates the job execution plan by TAMP [34] with task partition [35]. Members of the same team will undertake different tasks with the optimally selected components. The

state transition planning to achieve the goal is formulated with TAMP in Sec. VI-B.

This graphical storage method facilitates the information update process between multiple workers [36]. As different layers have their own uniqueness, they can be viewed and utilized independently for the purpose of simplicity. For example, $\mathcal{G}^R$ can be used in the hardware control of robots, and $\mathcal{G}^{job}$ can be used to check the job progress. In this paper, we assume all manipulable job objects are virtually connected in $\mathcal{G}^{job}$ when the job is completed.

The dynamic connectivity of the graph can reflect the current job progress based on the previous interaction history. When the job is in progress (i.e. a manipulable job object is being transported/manipulated), the job layer is a disconnected graph. Job completeness can be determined based on the connectivity in the Job Layer. The skeleton of the graph in each layer can be expressed with the Laplacian matrix $\mathbf{L}(\mathcal{G}^{job})$ [30], [37]. $\mathbf{L}(\mathcal{G}^{job})$ is always positive semi-definite. Its smallest eigenvalue is always zero. Thus, connectivity is determined by evaluating the second smallest eigenvalue. If a connected graph is formed (i.e. the second smallest eigenvalue is larger than zero) and the kinematic relation between all the job objects is the same as the goal $\mathcal{G}^{job}_{goal}$, then, the job is completed.

We here introduce two methods to represent capabilities and match capabilities. Based on their nature, we refer to these methods as (1) Passive approach and (2) Active Distributed approach. Both methods are dependent on the robot layer in ODKG. The first one adopts a waiting (hence passive) strategy of the agents towards collaboration, while the second one uses a constant search (hence active) for the best collaboration strategy. Task planning and job execution processes of both methods are identical and only depend on the ODKG.

IV. Passive Distributed Capability Matching

A. Strength-Based Capability Modelling — How strong am I?

To construct an ODKG for heterogeneous robots, we first model the configuration of each robot, which is equipped with (possibly) different hardware components. The capacity of each component is quantified by a *strength* metric whose magnitude encodes the capability to perform a given task. Large values signify a higher (or more specialized) capability.

In our proposed model, robots are either: (1) Assemblable, i.e., capable to form structures with other assemblable robots, or (2) non-assemblable, which can only work in parallel with other systems. Both non-assemblable and assemblable robots are capable to perform certain (typically simple) tasks, however, the strength level required to perform a challenging job could exceed that of a single robot. In this situation, a robot can collaborate with others by undergoing a capability enhancement or by simply passing the job to others [2]. To deal with this issue, we propose a capability-based dynamic task allocation algorithm that enables the shared execution of challenging tasks by a team of heterogeneous robots.

Note that each capability is denoted as an attribute node that is connected to the robot object in $\mathcal{G}^R$. We model the robot's capabilities as a matrix $\mathbf{C}^{self} \in \mathbb{R}^{N\times 5}$, where $N = D \times n$, for D as the number of the capability domains, and n as the number of strength levels. The proposed capability matrix is constructed with the following three terms:

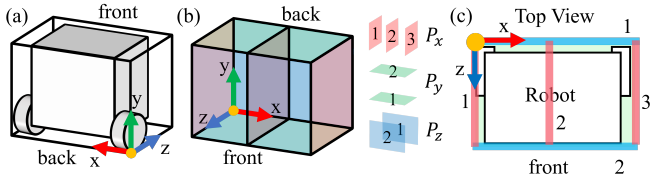

Fig. 6. Coordination system of the component location. (a) Origin is denoted with the orange circle; (b) planes definition for each axis; starting from Plane 1; (c) top view of the robot with the coordination system with Plane 2 on the y-axis excluded.

$$\mathbf{C}^{self} = \begin{bmatrix}\mathbf{s}^{self} & \mathbf{q}^{self} & \mathbf{P}^{self}\end{bmatrix} \tag{2}$$

where the strength vector $\mathbf{s}^{self} \in \mathbb{R}^{N\times 1}$ stores the intensity levels for the D capabilities in the structure:

$$\mathbf{s}^{self} = \begin{bmatrix}\underbrace{s_1,\ldots,s_n}_{\text{capability 1}} & \cdots & \underbrace{s_{N-n+1},\ldots,s_N}_{\text{capability } D}\end{bmatrix}^\intercal \tag{3}$$

for s_i as the ith strength level. Within the same capability domain, the strength levels are listed in ascending order, i.e., $s_1 < \cdots < s_n$ for capability 1 and so on. The structure of this strength vector will act as an indicator signal for protocol agreement between different robots.

The quantity vector $\mathbf{q}^{self} = [q_1,\ldots,q_N]^\intercal \in \mathbb{R}^{N\times 1}$ contains the number of robot components that belong to a specific capability domain, which are here denoted by d. In our model, components with a high strength level are capable of executing those tasks demanding a lower level within the same capability domain. Therefore, the quantity vector $\mathbf{q}^{self}$ is computed based on the following rule: $q_i = \text{count}(\text{level}(comp.) \geq s_i)$, where q_i equals the number of components with a strength level that is no smaller than s_i. e.g., if a robot has two components in the first capability domain whose strength levels are s_1 and s_2, thus, the quantity is $q_1 = 2$ and $q_2 = 1$.

The position matrix $\mathbf{P}^{self} \in \mathbb{R}^{N\times 3}$ stores the locations of the robot components and has the following structure:

$$\mathbf{P}^{self} = \begin{bmatrix}\mathbf{p}^{self}_x & \mathbf{p}^{self}_y & \mathbf{p}^{self}_z\end{bmatrix} \tag{4}$$

where the N-dimensional column vectors $\mathbf{p}^{self}_x$, $\mathbf{p}^{self}_y$ and $\mathbf{p}^{self}_x$ contain the x-axis, y-axis and z-axis coordinates of the robot components. We can write $\mathbf{P}^{self}$ in the following form:

$$\mathbf{P}^{self} = \begin{bmatrix}\mathbf{r}^{self}_1 & \mathbf{r}^{self}_2 & \cdots & \mathbf{r}^{self}_N\end{bmatrix}^\intercal \tag{5}$$

where the row vectors $\mathbf{r}^{self}_i = [x_i, y_i, z_i]^\intercal$ denote the 3D position of each component. In our model, we define the coordinate frame's origin such that the components' positions are always represented with positive values. We assume that these locations fall within a set of pre-specified integer values, as conceptually represented as planes in Fig. 6. The lowest plane in the proposed model is Plane 1, therefore, the location is always greater than or equal to 1 for any component, and 0 for non-existing components. If there is more than one component with the same strength level, only the most distant from Plane 1 is considered in the position structure.

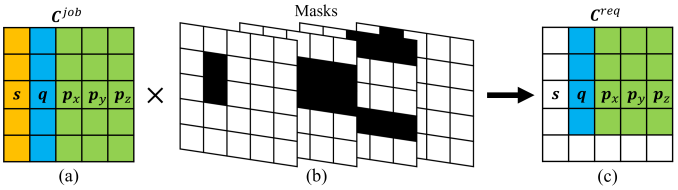

Fig. 7. Capability verification visualization. (a) Capability of a job $\mathbf{C}^{job}$; (b) position and Quantity Masks; (c) results $\mathbf{C}^{req}$ after applying masks on (a); only colored areas carry meaningful information; white refers to redundant data, i.e. 0.

B. Capability Verification — Can I do the Job?

The 'Job' is assigned randomly to the agent. As shown in Fig. 3, once a job is passed to an agent, capability verification is carried out to determine the operation mode. A job is represented with a similar structure to (2), instantiated as $\mathbf{C}^{job} = [\mathbf{s}^{job}, \mathbf{q}^{job}, \mathbf{P}^{job}]$. We can verify the suitability of the agent to perform the task by comparing the quantity of the components $\mathbf{q}^{job}$ required by the job with $\mathbf{q}^{self}$. This is done with a simple Rectified Linear Unit (ReLU) [38] as:

$$[\Delta\mathbf{q}]_i = \max(([\mathbf{q}]_i^{job} - [\mathbf{q}]_i^{self}), 0) \tag{6}$$

The elements in the vector $\Delta\mathbf{q}$ describe the missing components that the agent needs to find to perform the job.

The location verification between $\mathbf{P}^{job}$ and $\mathbf{P}^{self}$ of components in each capability domain if similarly performed with ReLu. The required components location $\mathbf{P}^{req}$ is computed as:

$$[\mathbf{r}_{k,j}^{req}]_i = \max(([\mathbf{r}_{k,j}^{job}]_i - [\mathbf{r}_{k,min}^{self}]_i),\ 0) \tag{7}$$

$$\mathbf{P}^{req} = [\mathbf{r}_1^{req}\ \ldots\ \mathbf{r}_N^{req}]^\intercal = [\mathbf{p}_x^{req}\ \mathbf{p}_y^{req}\ \mathbf{p}_z^{req}] \tag{8}$$

where $\mathbf{r}_{k,j}^{job}$ is the jth row vector of the kth capability domain. $\mathbf{r}_{k,min}^{self}$ is the vector of $\mathbf{P}^{self}$ that is the closest to $\mathbf{r}_{k,j}^{job}$, hence, best meets the specifications of the job. The required components location for a *Helper* is specified in $\mathbf{P}^{req}$.

To simplify calculations, our model adopts a binary masking approach [39], where masks are constructed to cross-map the robots' characteristics with the demanded job capability $\mathbf{C}^{job}$, see Fig. 7 for a conceptual representation. To determine the quantity of the location of each required component, we need to link the demand in $\mathbf{P}^{req}$ with $\mathbf{q}^{job}$ by constructing a binary mask of $\mathbf{P}^{req}$. To this end, we convert $\mathbf{P}^{req}$ into a vector with the same dimension as $\mathbf{q}^{job}$, and denote it as $\mathbf{m}^p$:

$$\mathbf{m}^p = \mathbf{P}^{req}\ \mathbf{e}_3 \in \mathbb{R}^{N\times 1} \tag{9}$$

where $\mathbf{e}_3 = [1,1,1]^\intercal$. To determine the number of final missing components (i.e. $\mathbf{q}^{req}$), we have to integrate the missing quantity $\Delta\mathbf{q}$ with the demanded quantity in terms of location $\mathbf{m}^p$. As the demand in position has a higher priority than that of the quantity needs, to avoid creating conflicting quantity results in $\mathbf{q}^{req}$, we cross-map the difference vector $\Delta\mathbf{q}$ with the mask $\mathbf{m}^p$. We use ${}^{\mathcal{B}}\{\mathbf{m}^p\}$ to normalize all terms in $\mathbf{m}^p$ greater than 0 to be 1, where ${}^{\mathcal{B}}\{*\}$ denotes the element-wise binarization operator. We combine ${}^{\mathcal{B}}\{\Delta\mathbf{q}\}$ with $\neg{}^{\mathcal{B}}\{\mathbf{m}^p\}$ by performing the logical 'and' operation. To this end, let us introduce the following term:

$$[\mathbf{b}]_i = [\Delta\mathbf{q}]_i([{}^{\mathcal{B}}\{\Delta\mathbf{q}\}]_i \wedge (\neg[{}^{\mathcal{B}}\{\mathbf{m}^p\}]_i)) \tag{10}$$

where $\neg$ and $\wedge$ denote the *not* and *and* logical operator, respectively. $\mathbf{b}$ is a quantity column vector in which the components represented in each $[\mathbf{b}]_i$ are only required from a quantity perspective, not a location perspective. To combine the quantity demands from $\mathbf{P}^{req}$ and $\mathbf{b}$, we apply the mask ${}^{\mathcal{B}}\{\mathbf{m}^p\}$ on $\mathbf{q}^{job}$ and combine the results with $\mathbf{b}$ to solve $\mathbf{q}^{req}$ as follows:

$$[\mathbf{q}^{req}]_i = [\mathbf{q}^{job}]_i[{}^{\mathcal{B}}\{\mathbf{m}^p\}]_i + [\mathbf{b}]_i \tag{11}$$

where $\mathbf{q}^{req}$ is a vector that integrates the missing components' quantity in terms of both location and quantity. Following (2), the final required capability $\mathbf{C}^{req}$ is constructed as:

$$\mathbf{C}^{req} = \begin{bmatrix}\mathbf{s}^{job} & \mathbf{q}^{req} & \mathbf{P}^{req}\end{bmatrix} \in \mathbb{R}^{N\times 5} \tag{12}$$

Once $\mathbf{C}^{req}$ has been determined by the agent, the working/collaboration format is determined following the rules described in Fig. 3. We now present a representative case study to illustrate the proposed methodology.

Example: Consider that the capability requirements of a job $\mathbf{C}^{job}$ and the agent's capability configuration $\mathbf{C}^{self}$ are:

$$\mathbf{C}^{job} = \begin{bmatrix}1 & 1 & 1 & 4 & 1\\ 2 & 0 & 0 & 0 & 0\end{bmatrix},\ \mathbf{C}^{self} = \begin{bmatrix}1 & 1 & 2 & 3 & 2\\ 2 & 0 & 0 & 0 & 0\end{bmatrix}$$

Based on the structure (2), we use (6) to obtain $\Delta\mathbf{q}$ and (7)–(8) to compute $\mathbf{P}^{req}$ as follows:

$$\Delta\mathbf{q}^\intercal = \begin{bmatrix}\max(1-1,0) & \max(0-0,0)\end{bmatrix} = \begin{bmatrix}0 & 0\end{bmatrix} \tag{13}$$

$$\mathbf{P}^{req} = \begin{bmatrix}\max(1-2,0), \max(4-3,0), \max(1-2,0)\\ \max(0-0,0), \max(0-0,0), \max(0-0,0)\end{bmatrix} = \begin{bmatrix}0,1,0\\ 0,0,0\end{bmatrix} \tag{14}$$

We calculate $\mathbf{m}^p$ with (14) by using (9) as follows: $\mathbf{m}^p = \mathbf{P}^{req}\mathbf{e}_3 = [1,0]^\intercal$, which is used to obtain the binary mask ${}^{\mathcal{B}}\{\mathbf{m}^p\} = [1,0]^\intercal$. The term $\mathbf{b}$ is computed by using (10) as follows $\mathbf{b} = [0(0\wedge 0), 0(0\wedge 1)]^\intercal = [0,0]^\intercal$. The required quantity vector can be calculated via (11) with the binary operations $\mathbf{q}^{req} = [1(1)+0, 0(0)+0]^\intercal = [1,0]^\intercal$. By combining $\mathbf{s}^{job} = [1\ 2]^\intercal$, $\mathbf{q}^{req}$ and $\mathbf{P}^{req}$ from (14), we can compute the final required capability matrix as shown below:

$$\mathbf{C}^{req} = \begin{bmatrix}\mathbf{s}^{job} & \mathbf{q}^{req} & \mathbf{P}^{req}\end{bmatrix} = \begin{bmatrix}1 & 1 & 0 & 1 & 0\\ 2 & 0 & 0 & 0 & 0\end{bmatrix} \tag{15}$$

where $\mathbf{C}^{req}$ shows that the agent is looking for a helper with one component ($[\mathbf{q}^{req}]_1 = 1$) containing the following features: a strength level of 1 ($[\mathbf{s}^{job}]_1 = 1$), and a height of 1 unit ($[\mathbf{p}_y^{req}]_1 = 1$).

C. Favor Granting — Let me Give you a Hand

Once $\mathbf{C}^{req}$ is ready, the agent starts searching for members. The communication among the MRS is represented using standard computer network models [40], [41]. If a solo/partnership/subcontract working format is possible (see Fig. 3), the agent spreads $\mathbf{C}^{job}$ to its idle neighbors. If assembly mode is possible, the agent spreads $\mathbf{C}^{req}$ with $\mathbf{C}^{job}$.

If an idle robot receives the request, it evaluates the agreement protocol based on the structure of its strength vector $\mathbf{s}^{self}$, and if it is asynchronous, the robot carries out

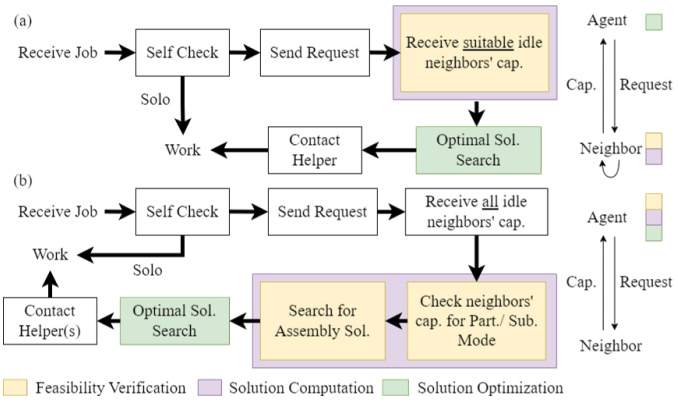

Fig. 8. (a) Passive Distributed Method: agent only focuses on solution optimization with pre-processed solutions. (b) Active Distributed Method: agent actively takes part in all decision-making processes with all unprocessed neighbors' data.

synchronization mapping with the incoming message. Afterwards, the robot computes the capability difference $\Delta\mathbf{C} = [\mathbf{s}^{job}, \Delta\mathbf{q}, \Delta\mathbf{P}]$ by applying (6)–(8) where $\Delta\mathbf{P} = \mathbf{P}^{req}$ in (8).

If $\Delta\mathbf{C} \neq [\mathbf{s}^{job}, \mathbf{0}_{N\times 4}]$, the neighbor lacks certain requested capabilities and forwards the incoming message to its neighbors. If $\Delta\mathbf{C} = [\mathbf{s}^{job}, \mathbf{0}_{N\times 4}]$, the robot replies with a favor granting message and sends back its own capability, which we denote as $\mathbf{C}^{fav}$, and the robot becomes a responsive neighbor.

D. Member Selection — Who will be the Best Helper?

The optimal member can be determined by comparing the capabilities of each responsive neighbor $\mathbf{C}^{fav}$ [42]. To compare matrices efficiently, we do not simply take the large matrix $\mathbf{C}^{fav}$ for calculation. Instead, we condense it to a column vector $\mathbf{k}$:

$$[\mathbf{k}^k]_j = [\mathbf{s}^k]_j[\mathbf{q}^k]_j[\mathbf{p}_x^k + \mathbf{p}_y^k + \mathbf{p}_z^k]_j, \quad \text{for } k = job, req, fav \tag{16}$$

where we multiple the elements in $\mathbf{s}^k$ and $\mathbf{q}^k$ to amplify their priorities in $\mathbf{k}^k$. We use addition in $\mathbf{P}$ to lower the impact of each $\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z$ to the overall condensation results. The similarity between the received favors and the requirement can be reflected by the Euclidean distance. We represent the similarity with the Euclidean distance score θ^k [43]:

$$\theta^k = \frac{1}{1 + ||\mathbf{k}^{fav} - \mathbf{k}^k||} \times 1000, \quad \text{for } k = job, req \tag{17}$$

where a shorter distance implies higher similarity.

The solution is obtained by finding the neighbor with the highest θ^k. The neighbor with the most similar capability to the demanded one is denoted as ω_o, and determined by:

$$\omega_o = \arg\max_{\omega}(\theta_1^k \dots \theta_{2a}^k), \quad \text{for } k = job, req \tag{18}$$

where ω is an active neighbor and a is the number of active neighbor. A favor acceptance acknowledgment is sent back to ω_o indicating its role. A timeout approach is used to indicate favor rejection [41].

V. Active Distributed Capability Matching

The method described in Section IV uses a strength-based strategy to describe the *specialty* of a component (i.e. the differences between components in each capability domain, describing how strong is a given component) with a passive distributed approach, where neighbors take part in determining the operability of a task, and the agent is only responsible for finding the optimal solution among all available pre-calculated solutions that are passively received from neighbors. However, the passive method (which we shall refer to as Method 1) may not provide global optimal results that consider the combination of *all* idle neighbors. To obtain global optimal results, we propose an active distributed method.

A. Type-Based Capability Modelling

Components can also be categorized into different types. There might be a situation where the types of components are considered instead of the strength level, such that each type of component can only carry out one type of job (e.g. some thermal cameras provide both thermal and RGB images while others only provide thermal information). To this end, we introduce an active distributed type-based capability approach, which we shall refer to as Method 2. A conceptual comparison between these two methodologies is illustrated in Fig. 8. Method 1 can provide local optimal results with a shorter elapsed time in capability matching. Method 2 can provide global optimal results but the elapsed time is longer. Hence, the trade-off is time with optimality.

We define the type-based specialty and quantity in new formats and denote them as $\mathbf{T}^k = [\mathbf{t}_1^k, \cdots, \mathbf{t}_D^k]^\intercal$ and $\hat{\mathbf{q}}^k = [\hat{q}_1^k, \cdots, \hat{q}_D^k]^\intercal$ where D is the number of capability domains. Each row refers to one capability domain d, for $d \leq D$. For example, if there are type 1 and type 2 components in the capability domain d, then $\mathbf{t}_d^k = [1, 2]^\intercal$ and the corresponding quantity is $\hat{q}_d^k = [2]$. As the dimensions of different $\mathbf{t}_i^k$ may not match with each other, we append 0s to the end of $\mathbf{t}_i^k$ to make sure $\mathbf{T}^k$ is a matrix. The position $\hat{\mathbf{P}}_i^k$ of each component $\mathbf{t}_\mathbf{i}^\mathbf{k}$ is individually denoted as $\hat{\mathbf{P}}_i^k = [\hat{\mathbf{p}}_{i,x}^k\ \hat{\mathbf{p}}_{i,y}^k\ \hat{\mathbf{p}}_{i,z}^k]$. The position of all components is represented as $[[\hat{\mathbf{P}}_1^k]^\intercal\ [\hat{\mathbf{P}}_2^k]^\intercal \dots]^\intercal$.

B. Capability Verification

The agent runs capability verification when it receives a job. Elements in $\mathbf{T}^{job}$ are compared with $\mathbf{T}^{self}$ row-by-row. If an element from $\mathbf{T}^{job}$ can also be found in $\mathbf{T}^{self}$, it is considered as specialty matched. If there is an element in $\mathbf{T}^{self}$ that fails to meet the requirement, the agent cannot work alone. Similar to Sec. IV-B, ReLu is used for quantity comparison. The quantity difference (i.e. $\Delta\hat{\mathbf{q}}$) is computed as in 6. If all the values in the result $\Delta\hat{\mathbf{q}}$ are equal to zero, solo working is feasible in terms of the quantity, otherwise, a co-worker is required. The agent sends a capability request to its neighbors but does not disclose any details of the assignment. Idle neighbors reply with their configurations $\mathbf{T}^\omega$, $\hat{\mathbf{q}}^\omega$, and the corresponding $\hat{\mathbf{P}}^\omega$ without any pre-calculation where we denote idle and replied neighbors as active neighbors (i.e. ω).

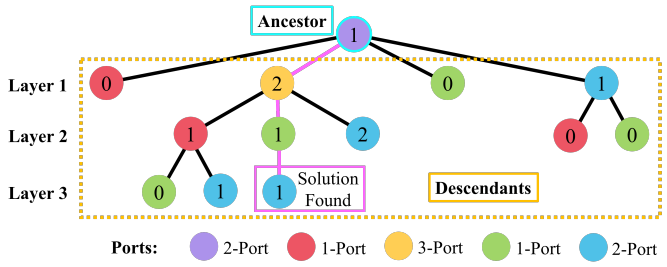

Fig. 9. A modified assemble search tree. The purple node indicates the agent which is also the ancestor Υ. Others are descendants υ. Ports refer to the number of assembled ports in a robot. The number in each node shows the currently available node remains. The search ends when a solution is found in layer 3.

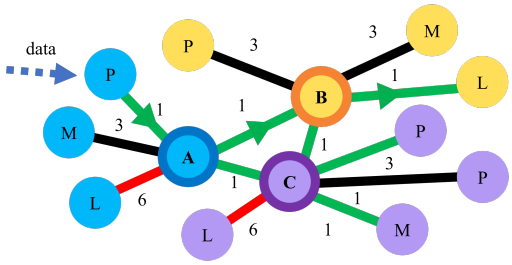

Fig. 10. Example of hardware control when new sensory data arrives to a group of assembled robot team that contains A, B, and C workers. A locomotion capability for action is in request. 'P', 'M', and 'L' stands for perception, manipulation, and locomotion components. The edges in red, black, and green indicate the highest, medium, and lowest cost connections. The arrow indicates the flow of the action command resulting from the least cost path calculation.

C. Collaboration Determination

In the following, 'solution' can be referred as possible collaboration results. It can be a single robot for partnership/subcontract mode or a list of robots for assembly. To calculate the optimal solution, the agent makes all the decisions actively based on all the received neighbors' capabilities. The position of a component is compared with ReLu when quantity and type are matched. If the result obtained from ReLu is not equal to 0, the component does not fit the requirement. If it is equal to 0, the component satisfies the requirement. All the neighbor's capabilities undergo a partnership and subcontract collaboration checking with the same capability verification stated in Sec. V-B. If there is any neighbor who fits the job criteria, it implies partnership/subcontract collaboration is available and it is one of the solutions. For capability enhancement, having more neighbors means a larger amount of combinations have to be analyzed, which is computationally costly and time-consuming. Inspired by the assembly approach used in [4], we modify the breadth-first search algorithm in tree topology to determine the optimal capability enhancement solution as depicted in Fig. 9. We refer this tree as the modified assemble search tree. The agent constructs this tree after it receives responses from its neighbors.

In this tree topology, we use a node to represent a robot. The agent is the ancestor Υ while the idle neighbors are descendants υ. Nodes involved in a path from Υ to any υ are the member of a team. Our goal is to obtain a path (i.e. a solution) for the optimal assemble combination that has no repeated or redundant robots in it.

Each robot has a limited number of assembly ports, here denoted as $\mathcal{P}$ and which is equal to the maximum number of degrees in its node. A υ in the tree is a node with at least one degree/assembly port, i.e. $\mathcal{P} > 0$. The number of available ports after assembly with the current node is the team's available ports $\mathcal{A}_{\mathcal{P}}$, which is calculated as:

$$\mathcal{A}_{\mathcal{P}} = \sum_{i=1}^{f} \mathcal{P}_i - 2(f-1) - g \tag{19}$$

where f is the number of nodes involved in the path from Υ to the current node. When two robots assemble, an assembly port from each robot will be used, hence, two ports are deducted from the remaining available ports (i.e., the second term in (19)). g represents the number of ground-contacted robots. For example, in a "tower-like" assembly (see Fig. 4) with robots attached on top of each other, a single robot is responsible for the team's mobility, thus, $g = 1$.

A descendant υ is created if and only if there is an assembly port left from its parent's layer and a new combination can be built. A solution check is carried out after a new layer is constructed. At the lowest level of υ, a path's capability is compared with the capability requirement if the path's $\mathcal{A}_{\mathcal{P}} \geq 0$. If there is a path satisfying the requirement, it is one of the solutions and the search is completed. A new layer will only be developed if there is no solution found in the current layer. An example of this modified search tree is shown in Fig. 9.

If more than one solution is available, we apply a capability similarity comparison to locate the optimal solution. Similar to (17), we define the Euclidean distance score $\hat{\theta}^i$ as follows:

$$\hat{\theta}^i = \frac{1}{1 + ||\hat{\mathbf{q}}^i - \hat{\mathbf{q}}^{job}||} \times 1000 \tag{20}$$

where we only focus on the number of components. The specialty is omitted here as a higher type index component does not imply it is better than that with a lower type index. i is an available solution that denotes the capability of an idle neighbor ω in partnership/subcontract mode or the overall capability of a robot team in assembly mode. The optimal member list can be reached with the largest $\hat{\theta}^i$ by (18).

VI. Capability-Based Task Allocation

A. Control of a Team of Robots

To address the hardware coordination of a team of robots, we first construct a weighted graph $\mathbf{W}(\mathcal{G}^R)$ from our ODKG. $\mathbf{W}(\mathcal{G}^R)$ is an adjacency matrix but with the cost included for edges. We use cost to indicate the availability and priority of accessing a component. We set the range of cost to be $[1, 2m]$, where m is the total number of members in a team. If there is any component not available to be used freely in the collaboration assembly (e.g., locomotion of Robot B in Fig. 4), the highest cost (i.e. $2m$) is used in the edge between the robot node and the component node. If there is any component not requested by the agent/job, these components are treated as redundant with a cost equal to m in the linkage. Otherwise, the component is regarded as free and with the minimum cost (i.e. 1). This situation is visualized in Fig. 10.

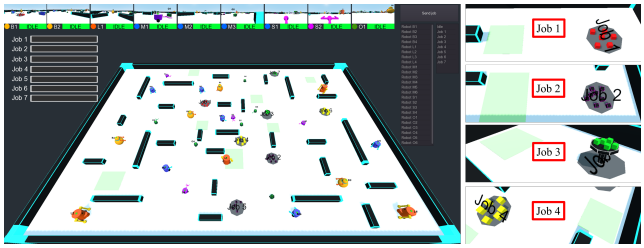

Fig. 11. Simulated robot factory with an enlarged view of each job.

When a robot detects new sensory information, it activates some functions/actions of the hardware components, such as moving or grasping. In this section, we focus on the hardware selection of each task, but not on task planning. Each capability has its own action set $\mathcal{A} = \{a_1, a_2, \dots\}$. To achieve accurate control, we find a minimum cost path with the Dijkstra algorithm [30], [44] from the current capability attribute node that receives the new sensory data to the capability attribute node that equips the required action a_i. Consider the situation shown in Fig. 10 as an example. When the perception component of Worker A receives a new useful observation, Worker A analyses the data and decides to take an action a_1, e.g., to move forward. This finds the minimum cost path from its 'P' node to a capability node that contains a_1 in its action set. From the node 'P' of Worker A to a locomotion capability node 'L' of Worker B, only 3 units of cost are involved, which is the lowest cost in the graph. Hence, the team controls the locomotion component of Worker B to execute the action a_1.

The proposed control mechanism can be used with all types of working modes. Its purpose is to ensure the action is taken by the correct component in a robot team without hardware conflicts. Fig. 10 conceptually depicts the approach.

B. ODKG: Planning and Update

To determine the response for each sensory data and environment change, we plan the action with the ODKG. Coordinating task allocation with capability consideration is complicated. Therefore, we utilize Task and Motion Planning (TAMP) [34] with task partition [35] to generate a feasible action policy $\pi = \{a_1, a_2, \dots\}$ for low-level motion planning. We use the Force-Based Algorithm for Motion Planning [45] when implementing the distributed path planning on multi-agents for each action a_i. In the task planning stage, once a job is allocated to an agent, the current $\mathcal{G}^{job}_{initial}$ and the final goal environment state $\mathcal{G}^{job}_{goal}$ with the required capability is passed to the agent. We take the $\mathcal{G}^{job}_{initial}$ and $\mathcal{G}^{job}_{goal}$ as the input for TAMP framework.

The pose of all objects in the kinematic graph is with respect to a constant object in the world frame or to a random object. When there is a new interaction caused by an action a_i (e.g., 'in-contact' or 'observe'), a new relationship between objects is created, and state transitions occur. A state, here denoted as $\mathtt{s}_i$, refers to the dynamic status of a group of variables. In our case, a state refers to the status of the environment and the robot-environment interaction. Each a_i drives the state $\mathtt{s}_i$ towards the next state $\mathtt{s}_{i+1}$ represented as $a = \langle \mathtt{s}_i, \mathtt{s}_{i+1} \rangle \in \Gamma$.

TABLE II
CONFIGURATION OF ROBOTS (METHOD 1)

Types	Amount	Perc.	Man.	Loc.	Size	Coll.	Ports
B	2	1, lv 1	2, lv 1	lv 2	4	lv 1	—
L	1	1, lv 1	2, lv 1	lv 2	5	lv 4	3
M	3	1, lv 1	2, lv 1	lv 2	3	lv 3	2
S1	1	1, lv 1	1, lv 1	lv 1	2	lv 3	1
S2	1	1, lv 1	1, lv 2	lv 1	2	lv 3	1
O1	1	1, lv 3	—	—	2	lv 2	1
O2	1	—	1, lv 2	—	2	lv 2	1
O3	1	—	1, lv 1	—	2	lv 2	1
O4	1	1, lv 2	—	lv 1	2	lv 2	1

TABLE III
CONFIGURATION OF ROBOTS (METHOD 2)

Types	Perc.	Man.	Loc.	Size	Coll.	Ports
B	1, type 1	2, type 1	type 1	4	type 1	—
L	1, type 1	2, type 1	type 1	5	type 1-4	3
M	1, type 1	2, type 1	type 1	3	type 1-3	2
S1	1, type 1	1, type 1	type 1	2	type 1-3	1
S2	1, type 1	1, type 2	type 1	2	type 1-3	1
O1	1, type 3	—	—	2	type 3	1
O2	—	1, type 2	—	2	type 2	1
O3	—	1, type 1	—	2	type 2	1
O4	1, type 2	—	type 1	2	type 2	1

After an action, the transition between the current state to the next state is denoted as $\Gamma \subseteq \mathtt{S} \times \mathtt{S}$, where Γ is the transitions, $\mathtt{S} = \{\mathtt{s}_0, \mathtt{s}_1, \dots\}$ is a set of states, $\mathtt{s}_0, \mathtt{s}_{goal} \subseteq \mathtt{S}$ are the initial and the goal state. We match the actions in π to the corresponding robot with $\mathbf{W}(\mathcal{G}^R)$ by task partition [35].

VII. RESULTS

A. Simulation Setup

To validate the proposed methodology, we develop a small virtual robot world with a game engine (Godot). The system enables to simulate multiple robots that collaboratively perform different types of jobs (see Fig. 11). As this method is focused on decentralized multi-robot collaboration, all agents move randomly in the idle state and execute their own scripts independently. No centralized algorithm is used for coordination. Data is stored by the main system for our further analysis. We perform all simulations on a PC equipped with an RTX-3060 GPU.

A simple box pick-and-place task is chosen as the job. The mission is simple but allows testing the proposed method as each job has different capability demands. For instance, some jobs require a certain strength/type of specialty while some do not. A job can generally be carried out by various working modes. Different kinds of assembly combinations may also be available for the same job.

We design nine robot types with diverse capabilities (see Fig. 1). The amount and configuration of each robot type are indicated in Table II and III, where "Amount", "Perc.", "Man.", "Loc", "Coll.", and "Ports" refer to the number of robots in the field, perception devices, manipulation tools, locomotion units, collaborability, and assembly ports, respectively. In this numerical study, we focus on the 'tower-like" assembly, where height is the most significant element in terms of dimension. "Size" refers to the height of a robot, and "lv" denotes level. Assemblability can be reflected in the number of available

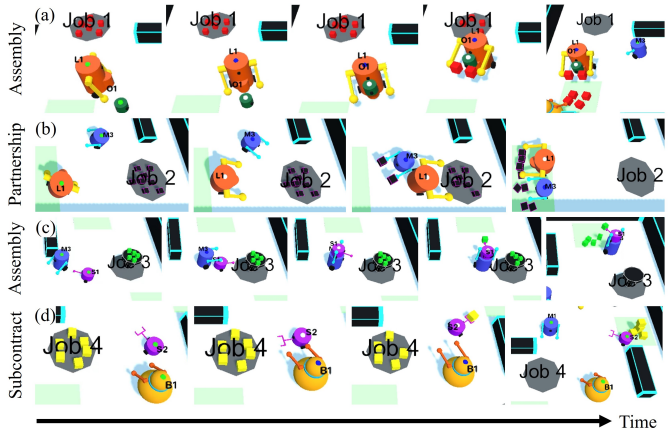

Fig. 12. Demonstration of different types of collaboration working process: (a) Assembly mode in Job 1: the orange robot goes and picks up the green robot for assembly. (b) Partnership mode in Job 2: the orange robot helps the blue robot in partnership mode. (c) Assembly mode in Job 3(a): the blue robot assemblies with the purple robot to extend its height for reaching the green box. (d) Subcontract mode in Job 4: the yellow robot passes the job to the purple robot as only the purple robot can meet the job requirement.

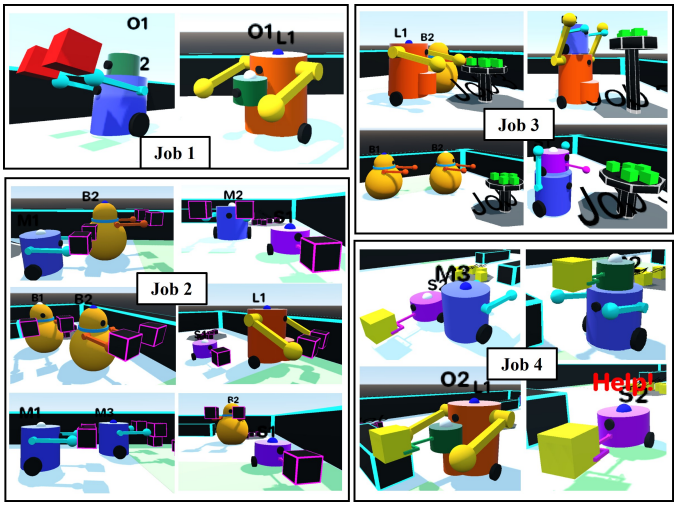

Fig. 13. Example results of different combinations: Job 1: assembly; Job 2: partnership; Job 3: partnership and assembly; Job 4: solo and subcontract

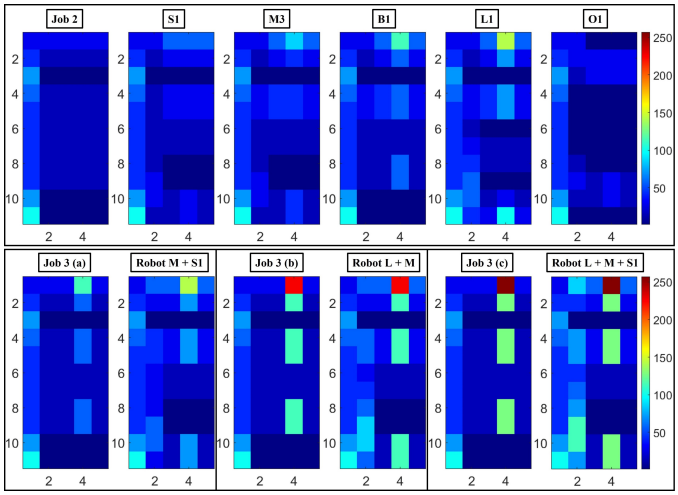

Fig. 14. Strength-based capability $\mathbf{C}$ visualization for Job 2 and 3 with each robot and team are shown with the range of $[0, 255]$. The larger number in the capability is, the higher the color index is. Job 3(a): 4-unit height; Job 3(b): 8-unit height; Job 3(c): 9-unit height.

assembly ports that are shown in the last column. "1, lv 1" and "1, type 1" refers to one component with strength level 1 and one component with type 1 specialty respectively.

Four different job types are created. Job 1 focuses on the specialty of perception. It requires a level 3 vision device to detect heat and is designed to test the assembly collaboration where the assembly mode is the only available option. Job 2 focuses on the member selection of multi-robot collaboration, and it only requires level 1/type 1 component for each capability domain. Capability enhancement is unnecessary in this job type. Job 3 focuses on the robot's capability enhancement in height, where three different height constraints are examined: (a) 4-unit height, (b) 8-unit height, and (c) 9-unit height. Job 3 is designed to examine all kinds of collaboration formats mentioned in the previous section. There are two types of robots (Type B and Type L), rich in height that they can conduct Job 3(a) in solo/partnership mode. Thus, all working modes are potential solutions. Job 4 focuses on the specialty in the manipulation domain. Type S1 robot is the only robot that is capable of accomplishing Job 4 in solo mode, and the Type O2 robot can accomplish Job 4 but in assembly mode.

B. Capability Enhancement

1) Passive Distributed Methodology: Results of optimal member determination are presented in Fig. 12. In Fig. 12 and Fig. 13, robots are able to perform the optimal capability enhancement to satisfy the job requirement. Take Job 3(a) as an example, where there are only two robot types. B and L are capable to conduct the job in solo mode (see Fig. 14). Type M robot has to assemble with Type S1 robot to satisfy the requirement as shown in 12 (c). Through assembly collaboration, those incapable robots (i.e. Type M and Type S1 robots) can now complete Job 3 with help from others.

2) Active Distributed Methodology: To test the performance of the capability enhancement approach, we use Job 3(c) as an example where at least three robots' involvement is needed. Fig. 14 shows that only when Robot L, M, and S1 assemble, their team capability of color level can reach the color level of the Job3(c) requirement. The assembly process is demonstrated in Fig. 15 (1)–(5). Four different robots are placed close to each other such that all of them are in the same neighborhood. When the blue Robot M3 receives a job, it calculates the feasibility of solo working. As Robot M3 cannot do it alone, it sends a help request to all idle neighbors which include orange Robot L1, purple Robot S1, and green Robot O1. To satisfy the job requirement, the agent computes the optimal solution which is to assemble with Robot L1 and S1. The assembly search tree results and the capability enhancement process is demonstrated in Fig. 15.

C. Collaboration Performance and Task Efficiency Analysis

Different member combinations are tested with our method, see Fig. 13. Robots can locate a suitable co-worker with an optimized capability to match results and form a team. Different working modes and collaboration strategies may

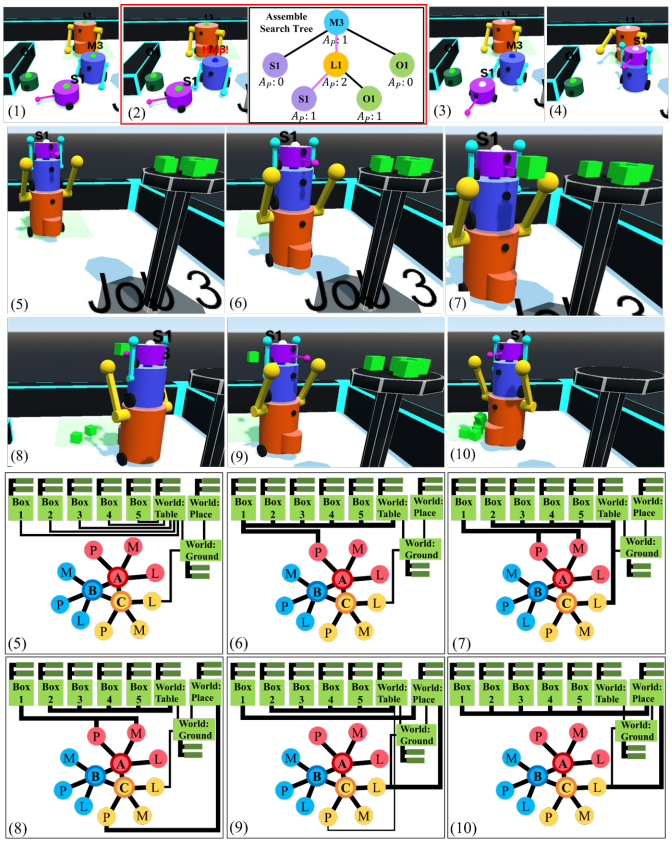

Fig. 15. Multi-robot assembles. (1) Before receiving any job. (2) Job 3 is assigned to the blue one. Its LED turns blue with a red "Help" word, indicating it is seeking help. A_P in the Assemble Search Tree refers to available port remains after the current assemble and the solution is indicated with pink lines. (3) The optimal solution is found. The LED on the purple and the orange robots turn white. (4) Assemble with reference to the solution. (5)-(10) Working and updating ODKG correspondingly where workers A, B, and C are the purple, blue, and orange robots.

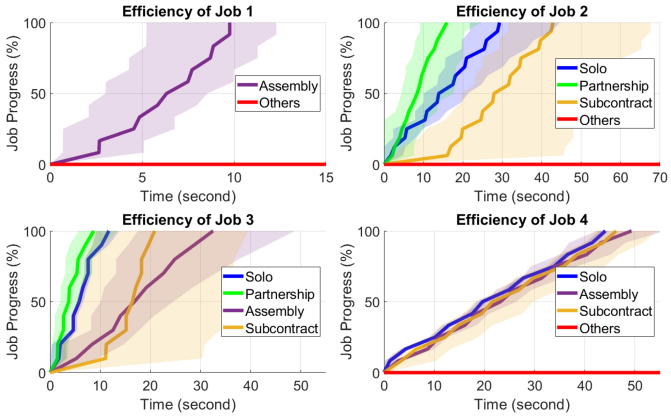

Fig. 16. Task efficiency of Job 1-4 under diverse working modes.

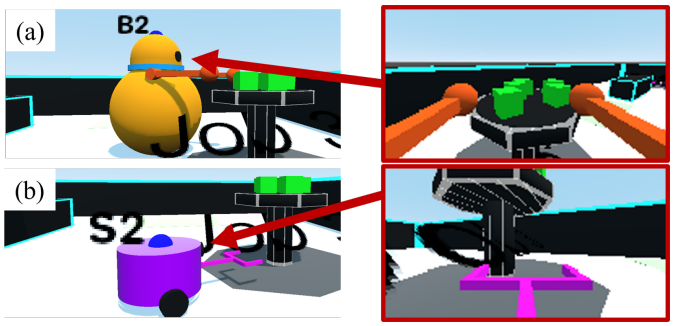

Fig. 17. Comparison our method with the state-of-the-art method. (a) The agent with our method can reach the box on the table; (b) the agent with the state-of-the-art method cannot reach the box on the table.

lead to different task efficiency results. To investigate the differences, robots are randomly placed on the field and each combination is tested 10 times.

The task efficiency outcomes of various working modes are shown in Fig. 16. The shaded area shows the upper and lower boundary of the results. In Job 1, as there is only one type of feasible solution, if the agent is not an assemblable robot, the mission cannot be completed. In Job 2 and 3, it can be easily observed that partnership collaboration can reach the highest efficiency as more manpower can boost productivity. Solo mode is the second most efficient way to conduct the job as there is no extra computation involved in collaboration decision-making. Subcontract is the slowest workable solution in Job 2. Although subcontract is the sub-optimal way to work, it is still valid as the job is completed. The assembly mode in Job 3 is the most time-consuming as assembly takes time, such as approaching the agent and performing assembly. In Job 3 and 4, there is a large overlapping proportion between the assembly and the subcontract mode. The disparity between these two working modes is not significant. In Job 4, robots can either subcontract with a Type S2 robot or assemble with a Type F2 robot. Thus, the performance of solo, assembly and subcontract are similar.

It can be summarized that partnership is the best strategy in general for completing a project with the least amount of time spent. As there are no differences between the outcomes with the designed workable solutions, the model is valid in terms of the working mode.

D. Experimental Setup

To validate the feasibility of our proposed model, a small robot setup with 4 types of robots is developed (see Fig. 1 and 18). Robot B1 and B2 are Type B robots; Robot L is Type L; Robot M is Type M; Robot O and O1 are Type O1 robots; Robot O4 is a Type O4 robot. The color of the LEDs on the robot's body indicates their current status (see Fig. 1). In the experiment, robots first analyze their observed data and then pass the useful information to others. In this section, we focus on Job 1 and Job 3 with the assembly mode. The requirement of Job 1 and 3 are the same as the simulation. To make the experiment more relevant to the real-world application, Job 1 requires moving goods away from a 'simulated explosive' object. Job 3 requires picking goods from a table with a height of 8 units and placing them into a box. The robots use a Raspberry Pi 4b for processing and communication, and an Arduino for low-level control. The Job is assigned from an Ubuntu computer to the robots. All programs and decisions are executed individually.

Fig. 18. Experiment: Job 1-3 with different methods. (a)(1) The agent receives a job and seeks favor. (a)(2) – (5) start working with various working modes: solo, partnership, or assembly. (b) A new job 5 is included that which requires 8-unit of height and a level 3/type 3 perception capability to locate a non-hot box. (b)(1)–(4) Robot L assembly and work with Robot M and Robot O1. (c) Method 1 and 2 give the same member selection results when Job 2 is assigned to Robot M1. (d) The average task efficiency of different working modes.

E. Collaboration Analysis with Object Pick-and-Place Tasks

In both methods, the agents form the optimal strategy based on their capabilities. In Job 1, we hide a heater inside one of the blue boxes to simulate the existence of explosive material. All blue boxes are identical and cannot tell the difference from the outlook. The robot needs to acquire a heat detector to differentiate the safe box from the dangerous one. We assign the job to Robot M. In Fig. 18, Robot M transfers states from $\mathtt{s}_{initial}$ to $\mathtt{s}_{assemble}$ and to $\mathtt{s}_{goal}$. Robot M successfully conducts the assembly with Robot O. With the extended thermal vision from Robot O, the team is capable to select the correct box $\{a_{detect}, a_{grasp}\}$ and move it to the desired area $\{a_{move}, a_{release}\}$, away from the explosive box. The experiment proves the robot's ability to execute the job collaboratively and to utilize the optimal components via the proposed methodology.

In Job 3, the height constraint is unachievable for any single robot in the field. To test the optimal member localization and the assembly working mode between Robot L and M, we specifically assign the job to Robot L. In the experiment, Robot L assembles with the appropriate worker which is Robot M. Through the implementation of the ODKG, the robot team can achieve optimal component control. The team uses the gripper from Robot M instead of that of Robot L, to grasp $\{a_{grasp}\}$ and unload $\{a_{release}\}$ the goods. With the capability enhancement, the team can pick up the goods from a table and put the goods into the target box for packing. Based on the results, it can be concluded that the proposed model is feasible and valid. It matches our observation in the simulation results. It also proves the potential for adopting the proposed model to solve the real-world multi-robot collaboration problem.

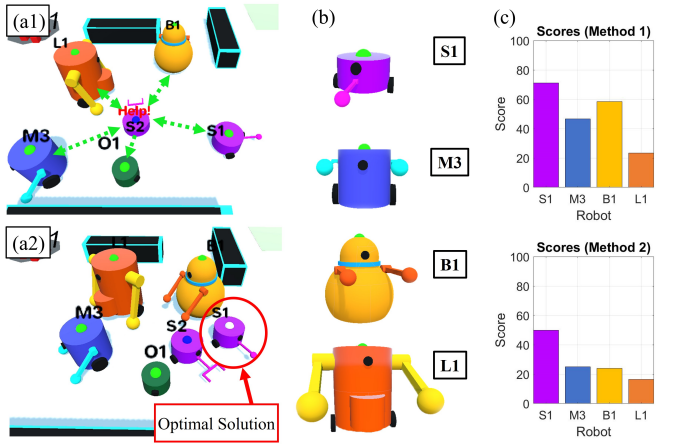

Fig. 19. (a1) Robot S2 sends a favor request; (a2) robot S1 is selected in both methodologies. (b) Potential Members. (c) Similarity score θ for each potential member with Method 1 and 2.

F. Performance Comparison

We numerically compare (i.e., with the simulation environment) the optimal performance of our proposed methods by placing multiple potential members around the agent. As shown in Fig. 19, all 6 types of robots are placed next to each other. We assign Job 2 to Robot S2. All the neighbors can conduct the job in solo mode, except Type O robots.

1) Passive Distributed Methodology: Job 2 requires a strength level 1 manipulation component. As Robot S2 is capable to carry out the job by itself, it looks for a partner to conduct the job in parallel and to boost productivity. The agent sends a favor request containing $\mathbf{C}^{job}$ to all its idle neighbors. The agent receives the favor-granted messages (i.e. $\mathbf{C}^{fav}$) from all robots, except Robot O1. As there are more than one neighbors who fit the criteria, there are multiple available solutions. From the scores in Fig. 19, we can observe that Robot S1 has the highest score. Results can be verified with the visualized capability in Fig. 14, where Robot S1 has the most similar capability to that of Job 2. Thus, the job is shared with Robot S1 in a Partnership form of collaboration.

2) Active Distributed Methodology: Job 2 requires a type 1 manipulation component. However, Robot S2 has a type 2 manipulation component, which means its capabilities do not match the requirement. As the solo mode is not feasible due to the lack of a type 1 manipulation component, Robot S2 needs a helper. A help request is broadcasted to the idle neighbors. The agent receives the capabilities of all idle neighbors. As the solo mode is not feasible and there are insufficient assembly ports on Robot S2, based on Fig. 3, the remaining solution is to adopt the Subcontract mode. There are 4 potential Subcontract collaboration solutions. The optimal solution is located via the scoring calculation where Robot S1 is the optimal choice with the highest score $\hat{\theta} = 50$ (see Fig. 19). Visually, Robot S1's color is the most similar to that of Job 2 shown in Fig. 14. Hence, the job is transferred to Robot S1 with the Subcontract mode.

We also compare the performance differences via experiments. In Fig. 18(c), we assign the same Job 1 to Robot M1 to evaluate the results differences and accuracy in selecting a member under passive and active approaches. Both approaches show the same results, with Robot O1 selected as the optimal neighbor.

Both methodologies are valid and have their own features. The passive approach puts individual consideration at the top priority while the active approach puts more emphasis on resource allocation. The differences in practical scenarios are that the passive strategy is more suitable when resources are not limited. The active strategy is more applicable in situations where resources are scarce. More details are discussed in the Sec. VIII-A.

G. ODKG Validation

We validate the ODKG model by assigning various types of jobs to our heterogeneous MRS. When the robot receives a job assignment, the agent builds the ODKG model and uses it for decision-making. In Fig. 12 and 13, the teams are capable of completing the assignment in solo or working collaboratively. Consider the three-robot assembly case with the use of Method 2 in the following (see Fig. 15). After the assembly, three robot ontology layers are connected. In the beginning, all objects 'box' are linked to the object 'table'. The locomotion capability node 'L' of the orange robot which is denoted as worker 'C' in ODKG and 'table' are connected with the object 'ground' as shown in Fig. 15 (5). When the purple robot (denoted as 'A' in ODKG) sees the 'box', there exists an interaction between the 'box' and the perception node 'P' of the worker 'A', demonstrated in Fig. 15 (6). As the kinematic graph shows that the observed 'box' is on the 'table', the robot has to go to the 'table' to reach the 'box'. Thus, the robot team moves towards the 'table' with the locomotion capability of 'C' and grasps the 'box' with the manipulation capability of 'A' in Fig. 15 (7). The team locates and moves to the target 'place' for unloading the 'box' shown in Fig. 15 (8)–(9). This process is repeated until all boxes are linked to the 'place' as illustrated in Fig. 15 (10). In this numerical example, the agent formulates the collaboration strategy and completes the job successfully.

Both passive and active methods provide similar results in the experiments. In the experiments shown in Fig. 18 (a) with the Job 1 Method 1 and 2, the agent Robot M collaborates with its member Robot O in assembly mode. An ODKG is constructed and evolves with time. From steps 1 to 3 shown in Fig. 18 (a), the unique perception information gathered by the Robot O (i.e. the location of the non-explosive goods) is shared to Robot M. The agent then triggers the 'move' and 'grasp' actions of its 'locomotion' and 'manipulation' capabilities with the proposed hardware coordination system.

We tested the partnership and solo modes with different approaches in the Job 2 experiments (see Fig. 18 (a)). The agent Robot M2 receives the job and works with Robot M1 in partnership mode. If Robot M1 lacks sufficient power, then Robot M2 works alone in solo mode. We use the robot layer in ODKG to coordinate the hardware allocation problem. In partnership mode, the least cost path from a feedback node (e.g. 'perception' node) to any action-related node (e.g. 'locomotion', 'manipulation' capability node) is 2. It indicates that using the node from the same robot which receives information is the best hardware allocation result.

In the experiments with Job 3 Method 1 and 2, the agent Robot L receives the job and collaborates with Robot M. An ODKG is built and used to coordinate the hardware resources in the team. As both robots are mobile, Robot M sends a 'move' action command to the ODKG to decide which robot's component to be used when it locates the goods in Step 3 shown in Fig. 18 (a). However, after reaching and grasping the goods, Robot M loses sight. The team relies on the perception information from the agent and uses the agent's 'locomotion' capability to move towards the unloading area. When the team reaches the unloading area, the team uses the 'manipulation' capability from Robot M to release the goods.

To further evaluate the effectiveness of the ODKG application, we introduce Robot O4 which is a Type O4 robot. We create a new job: Job 5, which requires an 8-unit of height, type 1 and 3 perception capabilities to determine the 'non-simulated explosive' box, and type 1 manipulation and locomotion components. In Fig. 18 (b), the agent Robot L finds the correct robots to be its helper: Robot L assembles with Robot M1 and Robot O1. During the task execution, they use the type 3 'perception' device of Robot O1 to detect hot objects. By combining the perception data of Robot O1 with Robot M1, the team successfully locates the non-hot box. With the hardware optimization in ODKG, the team uses the 'manipulation' component of Robot M1 to interact with the non-hot box and control the 'locomotion' of Robot L to navigate the environment. We illustrate the performance of the proposed methods in the accompanying video https://vimeo.com/726691079

VIII. Discussion and Conclusions

A. Discussion

We evaluate the proposed ODKG with different working modes and collaboration strategies. Through simulations and experiments, we demonstrate the efficiency of ODKG in heterogeneous MRS. The average task efficiency of adopting various working modes is shown in 16 and 18 (d). Both results proved that the partnership mode gives the optimal outcome with the shortest elapsed time for job accomplishment.

Both approaches are feasible and effective in capability-based task allocation. The passive approach allows neighbors to share some computational cost from the agent by focusing on individual offerings, but not the whole group. The active approach however values the resource utilization of the whole community before the individual. Passive and active methods are related to the collaboration attitude while strengths and types are for the capability modelling. These methods can be used interchangeably. For example, the passive approach can be used with type-based modelling and vice versa. In Method 1, the average elapsed time of ODKG (Phase 1) from job receives to member selection is around 0.17s in both small (8 robots) and large-scale (24 robots) simulation environments. Method 2 takes around 0.18s and 2.53s respectively as the agent requires more time to compare all neighbors' capabilities with the requirement, thus, the time increases with the number of neighbors.

There are two main limitations of the proposed methodology: (1) the processing time increases with the number of capabilities, strength levels, capability types, and neighbors, and (2) the capabilities of all robots have to be quantified beforehand. Various kinds of jobs can be assigned to the robots to evaluate their strength level and to classify them into different categories.

B. Conclusion

This paper presents a distributed dynamic framework to allocate collaborative tasks based on capability matching in heterogeneous MRS. Our methodology is designed to handle various kinds of working modes/collaboration between heterogeneous robots and to broaden the operability scope of the job nature of a robot. Two approaches are presented and their performances are tested through numerical simulation and real-world experiments. Various multi-robot collaborative tasks guided by the proposed method are conducted with optimized strategies. Future work aims to extend our proposed method to include various self-reconfigurable robots. Different real-world applications and collaboration modes will be explored to validate our method's performance.

References

[1] E. Montijano and C. Sagüés, *Distributed consensus with visual perception in multi-robot systems*. Springer, 2015.

[2] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: A survey," *Trans. on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 514–529, 2018.

[3] D. Tarapore, D. Floreano, and L. Keller, "Task-dependent influence of genetic architecture and mating frequency on division of labour in social insect societies," *Behavioral Ecology and Sociobiology*, vol. 64, no. 4, pp. 675–684, 2010.

[4] C. Liu, Q. Lin, *et al.*, "Smores-ep, a modular robot with parallel self-assembly," *arXiv preprint arXiv:2104.00800*, 2021.

[5] R. E. Wang, S. A. Wu, *et al.*, "Too many cooks: Coordinating multi-agent collaboration through inverse planning," *Int. Conf. on Autonomous Agents and Multiagent Systems*, 2020.

[6] J.-F. Boudet, J. Lintuvuori, *et al.*, "From collections of independent, mindless robots to flexible, mobile, and directional superstructures," *Science Robotics*, vol. 6, no. 56, p. eabd0272, 2021.

[7] J. Li, J. Wu, *et al.*, "Blockchain-based trust edge knowledge inference of multi-robot systems for collaborative tasks," *Communications Magazine*, vol. 59, no. 7, pp. 94–100, 2021.

[8] L. Yan, T. Stouraitis, and S. Vijayakumar, "Decentralized ability-aware adaptive control for multi-robot collaborative manipulation," *Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2311–2318, 2021.

[9] Y. Zhou, J. Xiao, *et al.*, "Multi-robot collaborative perception with graph neural networks," *Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2289–2296, 2022.

[10] Y. Huang, Y. Zhang, and H. Xiao, "Multi-robot system task allocation mechanism for smart factory," in *8th Joint Int. Information Technology and Artificial Intelligence Conf.* IEEE, 2019, pp. 587–591.

[11] Y. Emam, S. Mayya, *et al.*, "Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities," in *Int. Conf. on Robotics and Automation*. IEEE, 2020, pp. 7719–7725.

[12] S. Mayya, D. S. D'antonio, *et al.*, "Resilient task allocation in heterogeneous multi-robot systems," *Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1327–1334, 2021.

[13] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," *Trans. on Robotics*, vol. 33, no. 2, pp. 346–358, 2017.

[14] M. A. Karimi, V. Alizadehyazdi, *et al.*, "A self-reconfigurable variable-stiffness soft robot based on boundary-constrained modular units," *Trans. on Robotics*, 2021.

[15] A. L. Christensen, R. O'Grady, and M. Dorigo, "Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots," *Swarm Intelligence*, vol. 2, no. 2, pp. 143–165, 2008.

[16] R. O'Grady, A. L. Christensen, and M. Dorigo, "Autonomous reconfiguration in a self-assembling multi-robot system," in *Int. Conf. on Ant Colony Optimization and Swarm Intelligence*, 2008, pp. 259–266.

[17] S. Kernbach, O. Scholz, *et al.*, "Multi-robot organisms: State of the art," *arXiv preprint arXiv:1108.5543*, 2011.
[18] Y. Tu, G. Liang, and T. L. Lam, "Freesn: A freeform strut-node structured modular self-reconfigurable robot-design and implementation," in *2022 Int. Conf. on Robotics and Automation. IEEE*, 2022.
[19] D. Zhao and T. L. Lam, "Snailbot: a continuously dockable modular self-reconfigurable robot using rocker-bogie suspension," in *2022 Int. Conf. on Robotics and Automation*. IEEE, 2022, pp. 4261–4267.
[20] G. Liang, H. Luo, *et al.*, "Freebot: A freeform modular self-reconfigurable robot with arbitrary connection point-design and implementation," in *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2020, pp. 6506–6513.
[21] K. Macarthur, R. Stranders, *et al.*, "A distributed anytime algorithm for dynamic task allocation in multi-agent systems," in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 25, no. 1, 2011, pp. 701–706.
[22] G. Anders, C. Hinrichs, *et al.*, "On the influence of inter-agent variation on multi-agent algorithms solving a dynamic task allocation problem under uncertainty," in *Int. Conf. on Self-Adaptive and Self-Organizing Systems*. IEEE, 2012, pp. 29–38.
[23] Q. Yang, Z. Luo, *et al.*, "Self-reactive planning of multi-robots with dynamic task assignments," in *2019 Int. Symp. on Multi-Robot and Multi-Agent Systems*. IEEE, 2019, pp. 89–91.
[24] K. Lerman, C. Jones, *et al.*, "Analysis of dynamic task allocation in multi-robot systems," *The Int. Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.
[25] M. Chen and D. Zhu, "A workload balanced algorithm for task assignment and path planning of inhomogeneous autonomous underwater vehicle system," *IEEE Trans. on Cognitive and Developmental Systems*, vol. 11, no. 4, pp. 483–493, 2018.
[26] C. Moulin-Frier, T. Fischer, *et al.*, "Dac-h3: a proactive robot cognitive architecture to acquire and express knowledge about the world and the self," *IEEE Trans. on Cognitive and Developmental Systems*, vol. 10, no. 4, pp. 1005–1022, 2017.
[27] J. Ginés, F. Martín, *et al.*, "Social navigation in a cognitive architecture using dynamic proxemic zones," *Sensors*, vol. 19, no. 23, p. 5189, 2019.
[28] J. Akroyd, S. Mosbach, *et al.*, "Universal digital twin-a dynamic knowledge graph," *Data-Centric Engineering*, vol. 2, 2021.
[29] X. Zhou, A. Eibeck, *et al.*, "An agent composition framework for the j-park simulator-a knowledge graph for the process industry," *Computers & Chemical Engineering*, vol. 130, p. 106577, 2019.
[30] M. Mesbahi and M. Egerstedt, "Graph theoretic methods in multiagent networks," in *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.
[31] J. Bai, L. Cao, *et al.*, "From platform to knowledge graph: evolution of laboratory automation," *JACS Au*, vol. 2, no. 2, pp. 292–309, 2022.
[32] B. Smith, "Ontology," in *The furniture of the world*. Brill, 2012, pp. 47–68.
[33] A. Olivares-Alarcos, D. Beßler, *et al.*, "A review and comparison of ontology-based approaches to robot autonomy," *The Knowledge Engineering Review*, vol. 34, 2019.
[34] C. R. Garrett, R. Chitnis, *et al.*, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.
[35] F. L. Ratnieks and C. Anderson, "Task partitioning in insect societies," *Insectes sociaux*, vol. 46, no. 2, pp. 95–108, 1999.
[36] E. A. Sisbot, R. Ros, and R. Alami, "Situation assessment for human-robot interactive object manipulation," in *RO-MAN*, 2011, pp. 15–20.
[37] P. Song, W. Zheng, *et al.*, "Speech emotion recognition based on robust discriminative sparse regression," *IEEE Trans. on Cognitive and Developmental Systems*, vol. 13, no. 2, pp. 343–353, 2020.
[38] B. Hanin, "Universal function approximation by deep neural nets with bounded width and relu activations," *Mathematics*, vol. 7, p. 992, 2019.
[39] M. Mancini, E. Ricci, *et al.*, "Adding new tasks to a single network with weight transformations using binary masks," in *European Conf. on Computer Vision*. Springer, 2018, pp. 180–189.
[40] S. Santra and P. P. Acharjya, "A study and analysis on computer network topology for data communication," *Int. Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 1, pp. 522–525, 2013.
[41] N. Seddigh and M. Devetsikiotis, "Studies of tcp's retransmission timeout mechanism," in *Int. Conf. on Communications. Conf. Record*, vol. 6. IEEE, 2001, pp. 1834–1840.
[42] M. Lagomarsino, M. Lorenzini, *et al.*, "Pick the right co-worker: Online assessment of cognitive ergonomics in human-robot collaborative assembly," *IEEE Trans Cognitive Developmental Systems*, 2022.
[43] T. Segaran, *Programming collective intelligence: building smart web 2.0 applications*. " O'Reilly Media, Inc.", 2007.
[44] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *Int. Conf. on Mechanic Automation and Control Engineering*. IEEE, 2011, pp. 1067–1069.
[45] S. H. Semnani, A. H. de Ruiter, and H. H. Liu, "Force-based algorithm for motion planning of large agent," *IEEE Trans Cybernetics*, 2020.

Hoi-Yin Lee received the B.Eng. degree in Mechanical Engineering from The Hong Kong Polytechnic University of Hong Kong (PolyU), Kowloon, Hong Kong, in 2021. She is currently pursuing her Ph.D. degree in Mechanical Engineering at PolyU. Her research interests include multi-robot systems, perceptual robots, image processing, and automation.

Peng Zhou received his Ph.D. degree in Mechanical Engineering from The Hong Kong Polytechnic University, KLN, Hong Kong. He currently is a researcher at PolyU. His research interests include deformable object manipulation, robot learning, and interactive perception.

Bin Zhang received the Master degree in control science and engineering from the China Academy of Space Technology, Beijing, China, in 2020. Since 2021, he has been pursuing a Ph.D. degree in mechanical engineering at The Hong Kong Polytechnic University, Hong Kong. His current research interests include multi-agent systems and control theory.

Liuming Qiu is currently pursuing a B.Eng. degree in mechanical engineering at The Hong Kong Polytechnic University, KLN, Hong Kong. His research interests include the design and development of robot mechanisms, signal processing, and motion control.

Anqing Duan received the Ph.D. degree in robotics from the Italian Institute of Technology and the University of Genoa in 2021. He is currently Research Associate at The Hong Kong Polytechnic University. His research interest includes robot learning and control theory.

Jingtao Tang is currently pursuing a Ph.D. degree in computer and information engineering at the School of Science and Engineering, Chinese University of Hong Kong (Shenzhen), Guangdong, China. His current research interests include multi-robot systems and deep reinforcement learning.

Tin Lun Lam (Senior Member, IEEE) received the Ph.D. degree in mechanical and automation engineering from The Chinese University of Hong Kong (CUHK), N.T., Hong Kong, in 2010. He currently is an Assistant Professor at the School of Science and Engineering, Chinese University of Hong Kong (Shenzhen), Guangdong, China. His research interests include multi-robot systems, soft robotics, and human–robot collaboration.

David Navarro-Alarcon (Senior Member, IEEE) received the Ph.D. degree in mechanical and automation engineering from The Chinese University of Hong Kong in 2014. He is currently an Assistant Professor at the Department of Mechanical Engineering, the Hong Kong Polytechnic University. His current research interests include perceptual robotics and control theory. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS.