Semantic knowledge driven utility calculation towards efficient multi-robot task allocation

Chayan Sarkar, Sounak Dey, Marichi Agarwal
Embedded Systems and Robotics
TCS Research and Innovation, Kolkata, India
{sarkar.chayan, sounak.d, marichi.agarwal}@tcs.com

***Abstract*— With the advent of Industry 4.0 era, employing a team of robots within a factory floor or a warehouse is pretty prevalent today as robots can perform a known task with higher accuracy and efficiency if its capability permits. Efficiency and throughput of such a setup depend on careful task assignment and scheduling, which further depend on utility calculation. Though there exists a number of techniques to perform efficient task allocation, they assume the utility values are available and static. They neither consider all the relevant parameters nor the dynamic changes that may occur during task execution. Moreover, methods of automating such dynamic utility calculation (both at the start and at runtime) based on knowledge and semantics are not present and this is a hindrance to building a fully automated robotic workforce. In this article, we explore an avenue of semantic-based dynamic utility calculation and showcase its application for a use-case.**

I. Introduction

Mobile robots are a convenient tool to perform a set of tasks efficiently with minimal human intervention. Deploying a team of robots is often beneficial as opposed to a single robot [1]. Among others, efficiency of such a collaborative exploration depends largely on efficient multi-robot task allocation (MRTA), which has gained significant research attention due to its diverse range of applications [2], [3], [4]. The objective is to allocate tasks to the agents either instantaneously or time-extended fashion such that the overall goal can be achieved with minimal completion cost and/or time. A precursor to the task assignment with such an objective is to perform utility calculation. It generally refers to accurately estimating the cost of a particular task, if performed by a particular robot. This cost estimation helps to schedule and assign the tasks among the fleet of robots such that the overall cost to perform all the tasks can be minimized.

Industry 4.0 [5] standard proposes autonomy of robotic agents that are capable of coordinating among themselves, do inferencing, and take a decision on their own with minimal human intervention. Thus integrating knowledge bases to explore the relations between the concepts involved in different domains like robots, tasks, capabilities, environment, objects etc. is one of the prime focus of the research communities [6] [7]. Utility calculation at runtime is one step to that direction.

Related work. Gerkey *et al.* provides a formal analysis and taxonomy of all types of task allocation scenarios that can occur in a multi-robot system [8]. They provide the basic requirement of why optimization on utility is required while doing task allocation. For each of the different task allocation category, they present an optimization process based on utility. But it lacks the details about the list of parameters and methodology to calculate utility for a given set of task. Nnaji *et al.* [9] outlined a detailed methodology for a robot's utility calculation that is based on the utility theory. To do so, all features of a robotic device that are related to such a task must be identified and evaluated relative to one another. However, the scope of this method is restricted to the selection of a robot while purchasing it for a particular purpose. It is neither suitable for MRTA setup nor for a dynamic task allocation scenario. A similar approach is also proposed by Parkan *et al.* [10] for robot selection. Again, the utility calculation is neither suitable for MRTA setup nor it is supported by any knowledge-base to facilitate automation. Settembre *et al.* [11] has detailed a search and rescue scenario and outlined the importance of utility calculation. However, their method lacks the required details to actually calculate the utility values.

Challenges. If all the robots are homogeneous and the cost of performing a task is equivalent to the time required to complete the task, the utility calculation becomes trivial. However, in practice, it is a far more complex process and needs to consider a number of factors such as:

- Parameters specific to robot, environment, target object, etc., and their inter-relations need to be considered by a robot to estimate the cost of a task. As the existing task allocation algorithms assume a single utility value for a robot-task pair, combining heterogeneous parameters makes it a difficult task.
- Scenarios like "search and rescue" where manual control or supervision may not be possible in real time, robots may have to take a decision by themselves at runtime. For such cases, utility calculation mechanism should not be a hindrance towards automation.
- Usually manufacturers follow their own nomenclature and units for robotic specifications. Only domain experts can identify semantically similar terms and do necessary conversions. Presently, there is no standard mechanism to store and describe these data in a uniform machine-readable format such that semantics of those data can be interpreted by the robots.

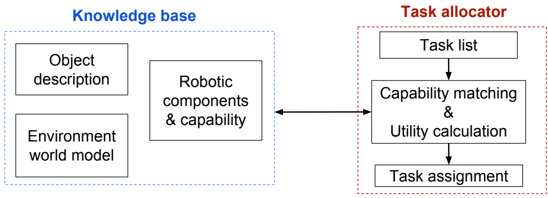

Fig. 1. Interaction between task allocator and knowledge base.

- Even if the robots are of the same type, value of certain parameters vary over time, e.g., battery state, current location, remaining weight carrying capacity (the robot may be already carrying some objects), etc. These need to be considered in runtime during dynamic task allocation.

Contributions. In this paper, we tackle most of these challenges. Specifically, our contributions are twofold. *First*, we provide a comprehensive tool/guideline to calculate utility for heterogeneous (as well as homogeneous) team of robots under a dynamic scenario. *Second*, we introduce semantic-enabled automated utility calculation and task allocation technique based on knowledge repositories aka ontologies.

II. Overview of utility calculation

The goal of utility calculation for MRTA is to provide the necessary and sufficient information to the task allocator such that it can allocate the tasks with minimum execution cost for the system. Fig. 1 shows a high-level system overview for utility calculation. The task allocator is assigned a set of given/discovered tasks with the relevant description. For each task, the task allocator consults the knowledge base to find the utility values for all the available robots. The three major components of the knowledge base that are required for utility calculation are shown in Fig. 1. Each of these components is represented using an ontology and an instance of these ontologies are populated with the relevant data for a particular environment. Given a task and a robot, the knowledge base provides all the relevant parameters and interdependencies for the robot-task pair. Then it combines these heterogeneous parameters to calculate a single utility value for the pair. In this process, it also filters the capable robots, i.e., the robots that are capable to perform a particular task. The utility values are stored in a matrix as a robot-task pair. Finally, this matrix is given as an input to a suitable task allocation algorithm.

III. Utility calculation modeling

Suppose a multi-robot system consists of a fleet of k robots and currently there are n tasks to be performed by them. The utility calculator generates a $k \times n$ matrix for each robot-task pair and passes it on to the task assignment algorithm. For robot i, the utility (cost) to perform task j can be calculated using the following equation,

$$u_{ij} = t_{ij}^{motion} + t_{ij}^{act} + t_{ij}^{rec}, \qquad (1)$$

where t_{ij}^{motion}, t_{ij}^{act}, and t_{ij}^{rec} are the total travel time for robot i, actuation time for the task j by robot i, and the recharge time for robot i, respectively. If the robot's battery is replaceable, t_{ij}^{rec} can be ignored. Now, t^{motion} depends on two parameters as shown in Eq. 2 – free path motion time (t_{ij}^{free}) and turning time t_{ij}^{turn}, where t_{ij}^{free} depends on the travel distance (d_{ij}^{total}) and speed of the robot (v_i) and t_{ij}^{turn} depends on average turning time (t_i^{turn}) and number of turns (n_{ij}^{turn}) on the robot's path. In case of an UAV, there is no turning time; rather it is the ascending/descending time to reach the desired flight level.

$$\begin{aligned} t_{ij}^{motion} &= t_{ij}^{free} + t_{ij}^{turn} \qquad (2) \\ t_{ij}^{free} &= \frac{d_{ij}^{total}}{v_i} \\ t_{ij}^{turn} &= t_i^{turn} * n_{ij}^{turn} \end{aligned}$$

t^{rec} is recharge time for the equivalent amount of energy spent by the robot (e_{ij}) to perform the task (Eq. 3).

$$t_{ij}^{rec} = \frac{e_{ij}}{e_i^{cap}} * t_i^{cap} \qquad (3)$$

Now, e_{ij} depends on different energy consuming activities – motion, sensing, actuation, load carrying, computation, and communication. These parameters are calculated based on the respective energy consumption rate (p_i) and the corresponding time for each of these activities.

$$e_{ij} = e_{ij}^{motion} + e_{ij}^{sense} + e_{ij}^{act} + e_{ij}^{load} + e_{ij}^{comp} + e_{ij}^{comm}$$

where,

$$\begin{aligned} e_{ij}^{motion} &= p_i^{motion} * t_{ij}^{motion}, \\ e_{ij}^{sense} &= p_i^{sense} * t_{ij}^{motion}, \\ e_{ij}^{act} &= p_i^{act} * t_{ij}^{act}, \\ e_{ij}^{load} &= p_i^{load_w} * t_{ij}^{load}, \\ e_{ij}^{comp} &= p_i^{comp} * (t_{ij}^{motion} + t_{ij}^{act}), \\ e_{ij}^{comm} &= p_i^{comm} * (t_{ij}^{motion} + t_{ij}^{act}). \end{aligned}$$

IV. Semantic knowledge representation

In order to complete a broad set of tasks on their own, a team of robots requires the domain knowledge available either *locally* or *in cloud* in some machine-readable and machine interpretable format. Relevant concepts from domains like environment, task, robot capabilities, robot components, target objects, dynamic values of robot parameters, etc., and their inter-relation, dependencies, constraints are needed to be captured in a structured knowledge model (i.e., ontology) via a robot-interpretable language such as OWL/RDF [12], [13], [14]. However, we limit our discussion in this paper on the use of knowledge base in utility calculation and task allocation only.

Robotic tasks like *picking, placing, moving, lifting, surveying*, etc., from different domains, depends on certain parameters like *weight of the target object, the location of the target, the size of the area under surveillance*, etc. However, not all tasks involve all the parameters that are required for utility calculation. For example, *surveying* an area does not need *weight of the object* while *picking* task

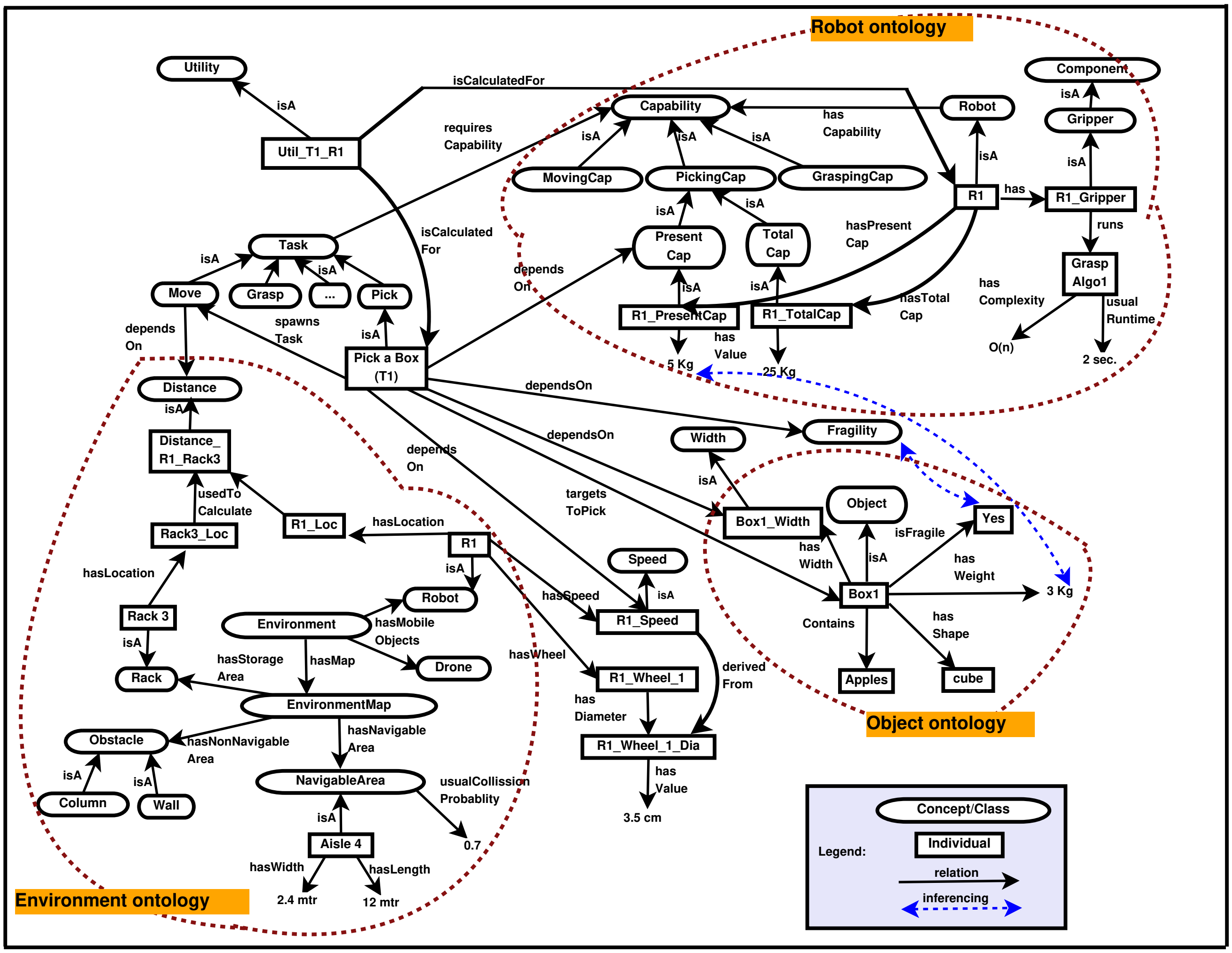

Fig. 2. Use of semantic knowledge base for calculating utility for a robot R1 against a task T1.

does not involve *size of an area*. For the first level of automation, the robots need to know the required parameters for a given task. Thus, for a global set of tasks say, $\{T_1, T_2, T_3, T_4,...,T_n\}$ and a global set of parameters say, $\{P_1, P_2, P_3, P_4,...,P_m\}$, there has to be a well defined one-to-many mapping between elements of task set and parameter set. The knowledge base needs to capture such *task vs. parameter* dependencies so that robots can query to know which parameters to consider against a given task. This is the first role of domain knowledge base for automated robotic utility calculation.

These parameters often depend on features of robots, environment, and other domain-specific parameters. To elaborate, let us take an example of a warehouse specific task like *"Pick a box of apple and place it at storage rack no. 3 in aisle no. 4"*. Now, for utility calculation as per Eq. 1, one need information like: *travel time to reach the box, time required to pick up the box, travel time to the storage rack, energy spent during these travelling/picking/storing etc.* But these data are not directly calculable as they are dependent on multiple sets of sub-parameters and their inter-relations. For example, the travel time depends upon the distance between the box and the current location of the robot, running cost of the path-planning algorithm, number of turns to be taken during navigation, the speed of the mobile robot, etc. Similarly, picking up the box depends upon weight and size of the box, fragility level of the box, computation cost for box identification, computation cost for pickup actuation, the energy required for gripping/picking, etc. Finally, to bring the picked-up box to its destination, various parameters like, height of rack no. 3, distance between aisle no. 4 and the box's current location, present load carrying capacity and battery level of the robot, battery recharge time/battery swap time, etc. are also need to be considered.

These basic parameters are not always specified in a uniform fashion by different manufacturers. For example, *speed of a mobile robot* is mentioned in *miles per hour* (MPH) for some robots while *meter per second* (m/s) is used for others. Moreover, some manufacturers mention *number of wheel rotations per second* instead of specifying speed. Thus one needs *wheel diameter* to calculate the speed in such cases. For a heterogeneous set of robots, these issues need to be solved semantically for uniform comparison and utility calculation.

```
calling...   retrieveRobotDetails (T1, R1):

     Details for Task-1 & Robot-1
+------------------------------+----++--------------+-------+
|        Specifications        |Data||   Contraints  | Values|
+------------------------------+----++--------------+-------+
|          Distance (m)        |100 ||Residual Energy| 10000 |
|          No. of turns        | 4  ||Required Energy|8293.69|
|     Average velocity (m/s)   |0.65|+--------------+-------+
|Turning time for 1.5708 rad (s)| 5  |
| Sense, compute & motion energy|21.6|
|      (without load) (J/s)    |    |
|Sense, compute & motion energy |23.6|
|    (with load of 1Kg) (J/s)  |    |
|     Actuation energy (J/s)   |1.2 |
|   Communication energy (J/s) |2.3 |
+------------------------------+----+
```

Fig. 3. Output of the API *retrieveRobotDetails(T1, R1)*.

If above task (of picking a box of apple) is assigned to a heterogeneous team of robots, then, in order to calculate the time required to reach to a height of say, 5 meters (height of rack 3), one needs to consider *average reaching speed* of the arm in case of a ground moving robot or *ascending speed* in case of an aerial robot. Without knowing that these two parameters are semantically similar, there is no way that a system can autonomously calculate utility.

The domain ontologies further help in utility calculation by supplying certain restriction or limitations that are there or needs to be satisfied. To elaborate: for above example, if the turning radius of a robot/drone is found to be one meter while the aisle width (found from warehouse map) is 0.8 m then the robot is not capable of doing the task at all and hence no utility calculation is done for it against this task.

V. Implementation and Results

We have created the ontology in OWL/RDF using Protege. The ontology is modular i.e., the concepts and relationships around the robotic entity are captured in *robot.owl* while *environment.owl* captures the same for the environment where the task is being executed. Some relevant concepts, relations, and individuals are shown in Fig. 2. To interact with the ontology and related database, an *API interface* has been created based on RDFLib (https://github.com/RDFLib/rdflib). Some example APIs are: *findRequiredHWcapability(task), findObjectDetails(object), getRackPos(rack#), retrieveRobotDetails(task, robot)* etc.

Fig. 3 shows sample output of the API *retrieveRobotDetails(task T1, robot R1)*, which provides parameter values of the Robot-1 (R1) with respect to the task T1. One can get similar details of other robots (say, Robot-2 and Robot-3) for the same task and can calculate utility for each of such robots using those values (detailed in Table I). The relevant parameter values are derived from [15]. Note energy required for Robot-3 is incalculable as it's actuation energy is undefined. This is because it does not have the picker arm and hence do not have the capability to pick the box. Similarly, Robot-2 needs 11.3 KJ to complete the task, which is higher than its residue energy (5 KJ). Hence, Robot-1 is the only recommended one for the task.

VI. Conclusions

In this paper, we have identified practical gaps in utility-based task allocation mechanism for the multi-robot systems and proposed a semantic knowledge-based framework for doing the same. This leads to a fully autonomous MRTA system. We demonstrate its use via a *"pick & place"* task, specific to a warehouse. In future, we plan to extend this framework to make a fully autonomous, semantic enabled, self-decision making system capable of completing any task.

TABLE I
Sample utility calculation for three robots w.r.t. a task.

Specifications	***Robot-1***	***Robot-2***	***Robot-3***
Distance (m)	100	150	70
No. of turns	4	2	7
Average velocity (m/s)	0.65	0.68	0.65
Turning time for 1.5708 rad (s)	5	5	4
Sense, compute & motion energy (without load) (J/s)	21.6	21.9	22.3
Sense, compute & motion energy (with load of 1Kg) (J/s)	23.6	24.7	25.1
Actuation energy (J/s)	1.2	1.05	∞
Communication energy (J/s)	2.3	2.25	2.1
Energy Required/Residue (KJ)	8.3/10	11.3/5	∞/9
Recommendation	***YES***	***NO***	***NO***

References

[1] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration."

[2] T. S. Dahl, M. Matarić, and G. S. Sukhatme, "Multi-robot task allocation through vacancy chain scheduling," *Robotics and Autonomous Systems*, vol. 57, no. 6-7, pp. 674–687, 2009.

[3] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2005, pp. 1145–1150.

[4] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[5] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE, 2016, pp. 3928–3937.

[6] M. Tenorth and M. Beetz, "Knowrobknowledge processing for autonomous personal robots," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 4261–4266.

[7] M. Stenmark and J. Malec, "Knowledge-based instruction of manipulation tasks for industrial robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 56–67, 2015.

[8] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[9] B. O. Nnaji and M. Yannacopoulou, "A utility theory based robot selection and evaluation for electronics assembly," *Computers & industrial engineering*, vol. 14, no. 4, pp. 477–493, 1988.

[10] C. Parkan and M.-L. Wu, "Decision-making and performance measurement models with applications to robot selection," *Computers & Industrial Engineering*, vol. 36, no. 3, pp. 503–523, 1999.

[11] G. P. Settembre, P. Scerri, A. Farinelli, K. Sycara, and D. Nardi, "A decentralized approach to cooperative situation assessment in multi-robot systems," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 31–38.

[12] G. s. Stephan, H. s. Pascal, and A. s. Andreas, *Knowledge representation and ontologies*. Springer, 2007.

[13] S. Bechhofer, "Owl: Web ontology language," in *Encyclopedia of database systems*. Springer, 2009, pp. 2008–2009.

[14] S. Dey, D. Jaiswal, R. Dasgupta, and A. Mukherjee, "Organization and management of semantic sensor information using ssn ontology: An energy meter use case," in *2015 9th International Conference on Sensing Technology (ICST)*, Dec 2015, pp. 468–473.

[15] A. Rico, I. Ruchkin, B. Schmerl, and D. Garlan, "Hardware power modeling for turtlebot," 07 2016.