Coordinating a Heterogeneous Robot Swarm Using Robot Utility-based Task Assignment (RUTA)

Tamer Abukhalil, Madhav Patil, Sarosh Patel, and Tarek Sobh, School of Engineering, University of Bridgeport, CT 06604, USA e-mail: tabukhal@ my.bridgeport.edu

***Abstract*— The goal of this work is the development of a task-oriented software application that facilitates the rapid deployment of multiple robotic agents. The task solutions are created at run-time and executed by the agents in a centralized or decentralized fashion. Tasks are divided into smaller sub-tasks which are then assigned to the optimal number of robots using Robot Utility Based Task Assignment (RUTA) algorithm. The system deploys these robots using its application program interfaces (API's) and uploads programs that are integrated with a small routine code. The embedded routine allows robots to configure solutions when decentralized approach is adopted.**

***Keywords*— Multi-robot task allocation, Decentralized utility-based task assignment, Heterogeneous Robot Swarm.**

I. Introduction

One of the key advantages of the corporative multi-agents robotic systems is fault-tolerance in which a robot can take over the task of a failing one. It has been proven that a single robot with multiple capabilities cannot necessarily complete an intended job using the same time and cost as that of multiple robotic agents. Different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the desired tasks may be too complex for one single robot, whereas it can be effectively done by multiple robots [1-3]. Modular robotic systems have shown to be robust and flexible in the tasks of localization and surveillance [4], reconnaissance [5]. Such properties are likely to become increasingly important in real-world robotics applications.

The main objective is to develop algorithms that can provide connectivity between multiple agents, besides building central software to track these agents. Such system design is motivated by our interest in multi-robot control for the deployment of potentially large numbers of cooperating robots with applications to tasks such as persistent navigation, object manipulation, and transportation. Online algorithms operate under the assumption that future events (inputs) are uncertain. Hence, they will occasionally perform an expensive operation to more efficiently respond to future operation. Generic and parameterized algorithms provide behaviors that are parameterized.

In the following section we provide a short analysis of existing swarm deployment environments. In section III we present the framework of a multi-robot coordination and deployment environment that provides decentralized/centralized control to mobile heterogeneous robotic agents. We do not consider any particular hardware or infrastructure of each swarm agent, as our focus is building control mechanisms that allow the system to operate several heterogeneous agents. In section IV, we bring a discussion on the coordination algorithm (RUTA). In section V we evaluate the proposed software framework in a human rescue application. Finally, Section VI presents a summary of the work and draws some conclusions.

II. Related Work

A comprehensive investigation and evaluation of the present multi-robotic systems (MRS) has been thoroughly discussed in our previous work [6]. In that survey we organized and classified ten swarm robotics systems and their corresponding behavioral algorithms into a preliminary taxonomy. We concluded that several algorithms have been developed to run on swarms of robots. These algorithms varied in complexity. Some provided basic functionality, such as leader following, while others exhibited complex interactions between the team of robots such as bidding on tasks according to arbitrary rules. Many early approaches in the literature concentrated on behavior-based technique where several desired behaviors are prescribed for each agent, and the final control is derived from a weighting of the relative importance of each behavior. On the other hand, recent researchers have begun to take a system controls perspective and analyze the stability of multiple robot agents. Other important hardware aspects of the current modular swarm robotic systems such as self-reconfigurability, self-replication, self-assembly, flexibility, and scalability were thoroughly analyzed in our other work [7-10].

In robotic control environments, a graphical application software such as MobileEyes [11] and the C++ based software URBI (Universal Real-time Behavior Interface) [12] are available as an open source systems. URBI provides GUI packages that aim to make a compatible code to different robots, and simplify the process of writing programs and behaviors for these robots. URBI works by incorporating sensor data to initiate commands to the robot. URBI packages, however, provide no abstractions. Therefore, it does not allow separating the controlling

system from the rest of the system. For example, a control system might be intimately tied to a particular type of robot and laser scanner. Moreover the URBI's uniform programming language is limited to few kinds of microcontrollers available on the market. The Player/Stage proposed by Gerkey et. al. [13] also produces tools for simulating the behavior of robots without an actual access to the robots hardware and environment. Its two main products are the Player robot server, a networked interface to a collection of hardware device drivers, and Stage, a graphical, two-dimensional device simulator.

Ayssam Elkady et. al. [14] have developed a framework that utilizes and configures modular robotic systems with different task descriptions. Their main focus was designing a middleware that is customized to work with different robotic platforms through a plug-and-play feature which allows auto detection and auto-reconfiguration of the attached standardized components installed on each robot according to the current system configurations. Therefore, the author's solution is mainly dealing with the abstraction layers residing between the operating system rather than software applications. A similar system hierarchy is used in Mobile-R [15] where the system is capable of interacting with multiple robots using Mobile-C library [16], an IEEE Foundation for physical agents standard compliant mobile agent systems. Mobile-R provides deployment of a network of robots with off-line and on-line dynamic task allocation. The control strategy structure and all sub-components are dynamically modified at run-time.

Approaches to multi-robot task allocation are divided into behavior-based and market-based approaches. ALLIANCE [17] is a behavior-based technique in which each robot performs a greedy task-selection algorithm for each task yielding a O(mn) per iteration where m and n are the number of tasks and robots respectively. At each iteration, each robot compares its own utility to that of the other robots and selects the task for which it is capable to perform. Because robots have to share their utilities in each iteration, communication overhead of O(n) is added to the overall execution time. ACO-based task allocation [18] is another behavior-based approach. In this technique, each robot has a corresponding task utility that decides if the robot is capable of executing a task by estimating the robot's utility for that task. Utilities are computed in a task-specific manner as a function of relevant sensor data. These utilities are periodically broadcasted to the other robots simultaneously to allow reassignment of tasks. Since each robot must broadcast its utility for each task, the system has a communication overhead of $O(mn)$ per iteration. Moving to auction-based approaches, In the M+ system [19], each robot considers all the currently available tasks at each iteration. For each task, each robot uses a planner to compute its utility and announces the resulting value to the other robots. With each robot broadcasting its utility for each task, we have communication overhead of $O(mn)$ per iteration. Similar to M+, the MURDOCH [20] task allocation mechanism also employs a variant of CNP. For each task auction, each available robot broadcasts its bid (i.e., utility), yielding communication overhead of $O(n)$ per iteration because of the asymmetric nature of MURDOCH's auctions. In ASyMTRe approach [21], the solution is based on perceptual schema representation of each robot's physical components. The solution requires the time to generate all the orderings of robots, which increases exponentially O(n!), and the actual reasoning time $O(mn^2)$ when utilities are being calculated. In ASyMTRe-D [22], the time is the average reasoning time O(mn) for the group to generate a solution.

III. System Architecture

We are developing an environment called UBSwarm- that runs an allocation algorithm to utilize robots that have different modular design and configuration of sensory modules, and actuators. The system is divided into two main subsystems, a robot deployment system and a robot control and translation system. The robot control system includes a robot control agent in which the user should provide all the parameters required for all sensors incorporated on robots. The user should also describe actuation methods used. The robot deployment system encapsulates a variety of high-level applications module which contains the tasks the platforms perform such as navigation, area scanning, and obstacle avoidance. A hardware abstraction layer is used to hide the heterogeneity of lower hardware devices and provide a component interface for the upper layers call. The basic hierarchy of the UBSwarm deployment environment is shown in Fig. 1.

The deployment system interacts with agents through various types of communication mediums. The deployment system takes the responsibility of running actions according to the definition parameters and the different integrations of the heterogeneous robots. Each application is implemented as a software module to perform a number of specific tasks used for sensing, decision-making, and autonomous action. Actions are platform independent robot algorithm; for example, it can be an obstacle avoidance algorithm or a data processing algorithm using Kalmans filter, etc. These actions can communicate together using message channels. A full description of the framework's components is provided in our previous work [5].

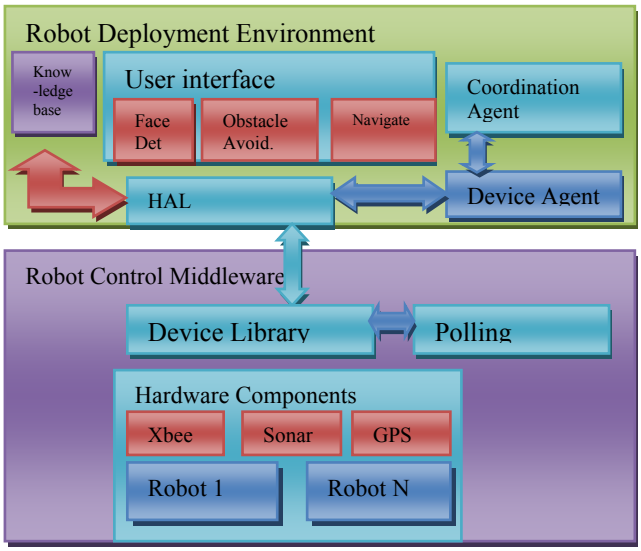


Figure. 1: Software Architecture

TABLE 1: FIVE ROBOTS AND THEIR CAPABALITIES

Robot	Available sensor (s) /capabilities	Wheels slip percentage
R1	VGA camera, URM Ping, V32 ultrasonic, 2-Dof arm, wheel Serial motors.	3%
R2	V32 Ultrasonic, 2-dof arm, two Serial motors	9%
R3	Sonar sensor, 1- Dof arm, two Serial motors	9%
R4	Serial motors, two sonar sensors, 1-Dof arm	5%
R5	VGA camera, Serial motors, two sonar sensors, 1-Dof arm	2%

TABLE 2: SENSING AND ACTUATION COMPONENTS

Sensing/actuation Component	Consumption rate
VGA Camera	20 mA
URM Ping	20 mA
V32 Ultrasonic	4 mA
2-Dof (2 servos)	2x(120 mA)
Serial motors for wheels	2x(160 mA)

IV. RUTA ALGORITHM

We assume $R = \{r_1, r_2, \ldots\ldots, r_n\}$ is a collection of n robots, where each robot Ri is represented by its available environmental sensors (ES), motor devices (MD), and communication devices (CD). Table 1 shows the configuration of robots used in experiment set 1. Table 2 shows the shows consumption rates for actuation and sensing components for robot R1.

Our approach to multi-robot task allocation problem (MRTA) is based on the following assumptions:

- T is task to be accomplished, which is a set of *m* subtasks that are basically composed of motor, sensor and communication devices that need to be activated in certain ways in order to accomplish this task. Its denoted as $T_i = \{v_{i1}, v_{i2}, v_{i3}, \ldots\ldots, v_{im}\}$ where v_{ij} is the subtask *j* performed by robot r_i and $1 \leq i \leq n$, $1 \leq j \leq m$
- A subset v_{ij} of T_i, can be allocated to robots concurrently if they do not have ordering constraints.
- To accomplish the task T_i on robot r_i, a collection of *n* plans (solutions), denoted $P_i = \{P_1, P_2, \ldots, P_n\}$, needs to be generated based on the task requirements and the robot capabilities.

We define a cost function for each robot, specifying the cost of the robot performing a given task, and then estimate the cost of a plan performing the given task. We consider two types of cost:

- A robot-specific cost determines the robot's particular cost (e.g., in terms of energy consumption or computational requirements) of using particular capabilities on the robot r_i to accomplish a task v_{ij} (such as a camera or a sonar sensor). We denote robot $r_i's$ cost by cost (r_i, v_{ij}).
- The cost of a plan P_i performing a task T_i is the sum of individual cost of n robots performing sub-tasks m that are in the plan P_i, which is denoted in (1):

$$Cost(Pi, Ti) = \sum_{i=0}^{n} \sum_{j=1}^{m} \text{cost}\left(r_i, v_{ij}\right) \quad (1)$$

The problem we address here is the optimal assignment problem (OAP). The solution is called Robot Utility-based Task Assignment (RUTA) and it can be summarized as the following: given (T, R), find a solution P_i to each task T_i such that $Cost(Pi, Ti)$ is minimized.

We assume that sub-tasks v_{ij} allocated to robot r_i must be ordered into a schedule $\sigma_i = (v_{i1}, t_1, t'_1), \ldots\ldots, (v_{ij}, t_j, t'_j)$ for $1 \leq j \leq m$ where v_{ij} is the subtask performed from time t_j *to* t'_j. Each sub-task assigned to a robot is denoted by a triple; $\alpha_j = < type, t_{ej}, rate_j >$ representing the v_{ij} task type whether its sensing or actuation type, time assigned to the task until it is accomplished (so $t_{sj} = t'_j - t_j$), and the consumption rate (in mA) for this particular subtask respectively. Depending on the robot $r_i's$ location, the time spent on each task must equal $r_i's$ assigned share of the workload. We also assume that the distance in meters between robot r_i to subtask v_{ij} is d_{ij}. Taking these values into account, each robot can be represented as $\beta_i = < id, w_i, Prem_i >$, representing the robot's id, percentage of wheel slip, power remaining, and distance to the sub-task respectively. The cost of a robot r_i performing a subtask v_{ij} is calculated by dividing the robot r_i battery remaining power by the product of multiplying the sensor and/or actuator consumption rate with the percentage of time in which it is operating. This is determined by equations (2-5).

$$\varphi_{manip\,ij} = 0.7 \times \left[\left(\frac{t_{sj}}{t'_m}\right)\left[\frac{Prem_i}{rate_{act\,j}}\right]\right] \quad (2)$$

$$\varphi_{nav\,ij} = 0.7 \times \left[\left[\frac{prem_i}{rate_{servo\,j}}\right] \times \frac{1}{w_i}\right] \quad (3)$$

$$\varphi_{sens\,ij} = 0.9 \times \left[\left(\frac{t_{sj}}{t'_m}\right)\left[\frac{Prem_i}{rate_{sens\,j}}\right]\right] \quad (4)$$

$$\varphi_{given\,ij} = \varphi_{manip\,ij} + \varphi_{nav\,ij} + \varphi_{sens\,ij} \quad (5)$$

Where t'_m is the total time predetermined for the robot r_i to complete all of its subtasks in seconds, w_i is the pre-assumed percentage of robot r_i wheel slip, and $\varphi_{manip\,ij}$, $\varphi_{nav\,ij}$ and $\varphi_{sens\,ij}$ are the qualities to perform manipulating, navigation, and sensing subtasks respectively. Depending on the subtask type, the value of any of these quality functions is null if they are not taking place in the subtask. $\varphi_{gevin\,ij}$ is the total quality of subtask v_{ij} being performed by robot r_i. When obstacle avoidance task is being performed, the quality function $\varphi_{given\,ij}$ has higher values than the other qualities because it includes navigation as well as sensing

subtasks. The priorities of subtasks must be considered and are calculated according to the schedule of tasks σ_i that is set to robot r_i. The priority of robot r_i performing a subtask v_{ij} is defined in (6), its value is varying from 0 to 1.

$$pri_{ij} = \frac{1}{2} \times min\left[(u_1 \times (t - t_j), 1\right] \quad (6)$$

Where t is the current time elapsed since the beginning of the task, t_j is the time when the task is announced as declared in the schedule σ_i. The parameter u_1 adjusts how the priority should increase with the value of $(t - t_j)$.The assignment of a subtask v_{ij} to the specific robot (that is capable of accomplishing it) is determined by the Utility function of a robot r_i performing a task v_{ij} as in (7):

$$utility_{ij} = \max(0,\ u_2 \times (d_{ij}{}^{-1/2} \times \varphi_{given\ ij} \times pri_{ij}) \quad (7)$$

Where $utility_{ij}$ is the nonnegative utility of robot r_i for sub-task v_{ij} , $1 <= i < =n$, $1 <= j <= m$, u_2 is the weighted coefficient to adjust the effect of the variables inside the equation. d_{ij} is the distance in meters between robot r_i to subtask v_{ij}. We assume that each robot r_i is capable of executing at most one task at any given time. We also assume that multiple agents can also share a single sub-task in which they divide the workload. Initially the task is introduced to the coordination agent which in turn performs the following algorithms:

Centralized Algorithm 1: Input: (T, R, M, N)

1. *Schedule sub-tasks v_j , such that ordering constraints are satisfied.*
2. *if (N=1) then Stop*
3. *Else*
4. *Sort the robots according to decreasing computational and sensory capabilities*
5. *Initially the $utility_{ij}$ for all robots and subtasks is equal to 0*
6. *Calculate utilities of each of the N robots*
7. *Based on the task requirement T, pick "at least" two robots with highest utility values.*
8. *For each sub-task v_j*
9. *For each robot r_i of the two selected robots*
10. *Assign subtask $v_j$to r_i based on the task requirements*
11. *→Add (r_i, v_j) to plan Pi*
12. *→Update parameters in v_j*

Decentralized Algorithm 2: Input (R, N)

1. *Utility is calculated on each robot*
2. *Two robots with highest utility values will begin their pre-programmed plans*
3. *While task is not complete {*
4. *Each robot's utility value is shared with the other robots. When a robot is introduced to the system or If a sensor fails on one robot r_i by which it prevents it from completing task v_j, it sends a request (bid) to the other robots in the team.*
5. *Robot waits for reply (t_{out}) to hear respond from the most fit one (based on the winner highest utility value).*
6. *Task v_j is taken over by the winning robot. }*

IV. *HUMAN RESCUE EXPERIMENT*

Our swarm system is composed of rather semi-intelligent heterogeneous robots. The robotic platforms as shown in fig. 2 are built using Arduino UNO, Arduino Due, and Digilent PIC boards.

A. Task allocation

A human rescue algorithm has been developed for the swarm so that robots can autonomously cooperate and coordinate their actions so that a human dummy can be pulled away in a minimal time. Cooperation between robots is achieved by exchanging messages when an additional robot is needed to pull the object. The desired pulling distance for 1200g human dummy was 2.5 meters. The sequenced photos in fig. 3 show an example of five robots pulling the dummy. We executed centralized as well as decentralized modes to perform three trials for each experiment set indicated by the number of robots, and obtained data on the completion time and the number of successful experiments. In total, we have performed 30 trials.

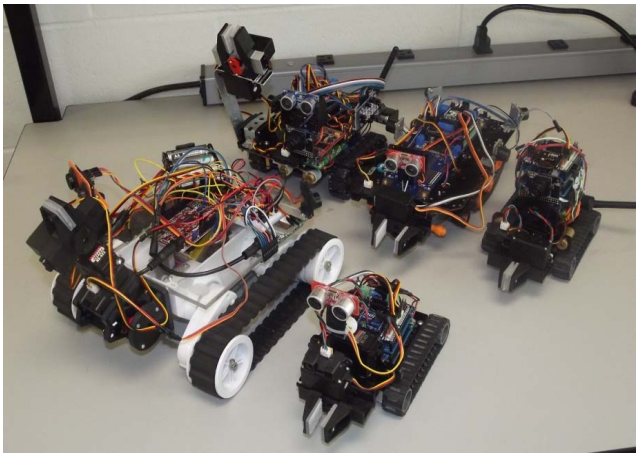
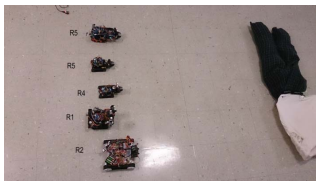


Figure. 2: The Heterogeneous swarm robots



Five robots are configured

Four robots are approaching

The fifth robot, R2 is called

Five robots crossing the finish line

Figure. 3: A dummy being pulled for 2.5 meters using five robots

TABLE 3: SUCCESSFUL PULLING DISTANCE ACCORDING IN CENTRALIZED APPROACH

Team Size	**Weight of dummy**	**Average Pulling distance (meters)**	**Average Time (seconds)**
1	300g	1.6	134.3
2	800g	2.3	144.2
3	1200g	4	149.6
4	1200g	5.2	100.2
5	1200g	5.2	110.3

The team that is constructed of four robots was able to accomplish the task in an average of 100.2 seconds with a standard deviation of 10.3 seconds. For this experiment, the decentralized parameters and negotiation timeout values are set as follows: wait for reply is 0.85s and wait for confirmation is 4s. Table 3 shows performance data collected from the centralized experiments. As an example, in both approaches the total cost of task (T_{rescue}) performed by the robots $r_i's$ in the capability-based ordering (R2, R3, R1, R4, R5) is determined by the robots utility functions associated with each of the following tasks:

$$T_{rescue} = \sum_{i=1}^{5} U_{rescue(i)}$$

$$\sum_{i=1}^{5} U_{rescue(i)}$$

$$= \sum_{i=1}^{5} (utility_{i(nav)} + utility_{i(detect)} + utility_{i(grip)} + utility_{i(pull)})$$

$$utility_{ij} = \max(0,\ u_2 \times (d_{ij}{}^{-1/2} \times \varphi_{given\ ij} \times pri_{ij})$$

Where j = 1,2,3,4 i = 1,2,3,4,5, $U_{rescue(i)}$ is the overall utility of robot r_i, and $utility_{i(nav)}$, $utility_{i(detect)}$, $utility_{i(grip)}$, $utility_{i(pull)}$ are utilities of the navigation, object detection, gripping, and pulling subtasks. The first subtask to be performed is navigation, the utility for first robot using the centralized algorithm is calculated as follows: Robot 1, j = navigation

$$utility_{1(nav)} = \max(0, \quad u_2 \times (d_{1j}{}^{-1/2} \times \varphi_{given\ 1j} \times pri_{1j})$$

$$\varphi_{given\ 1j} = \varphi_{nav\ 1j} = 0.7 \left[\left[\frac{prem_1}{rate_{servo\ (1)}}\right] \times \frac{1}{w_1}\right]$$

$$\varphi_{given\ 1j} = 0.7 \left[\left[\frac{2200}{130}\right] \times \frac{1}{3}\right]$$

$$\varphi_{given\ 1j} = 0.7\ [5.58]$$

$$\varphi_{given\ 1j} = 3.90$$

Initially priorities of all sub-tasks are equal to 1, and $u_2 = 1$ hence,

$$utility_{1(nav)} = \max(0, \quad 1 \times (1^{-1/2} \times 3.90 \times 1)$$

$$utility_{1(nav)} = 3.90$$

B. Fault tolerance

In Fig. 4, the plan utility is plotted for four different team sizes. At time 25 seconds, an error is introduced to one of the robots in each team. We can observe that the team accumulated utility drops down at that point then as both approaches re-allocate the tasks the overall utility increases. The figure shows the accumulative team utility over time. The sub-task that is assigned to the faulty robots is taken over by the rest of the team as a result of the reasoning algorithms executed by the two control schemes. The centralized results always have a higher utility than that of the Decentralized RUTA, because the centralized approach operates with complete information received from the robot team. Moreover, the decentralized approach's core functionality is based on the use of time-based parameters (i.e. wait-reply t_{out} and wait-for-confirmation) that not only requires more communication overhead amongst the robots but also increases the time slot given to the particular subtask and thus increases the robot's cost. Because all of the previous architectures execute greedy algorithm for task allocation, the solution quality of greedy optimization algorithms can be difficult to define. Evaluating each architecture depends strongly on the nature of the experiment.

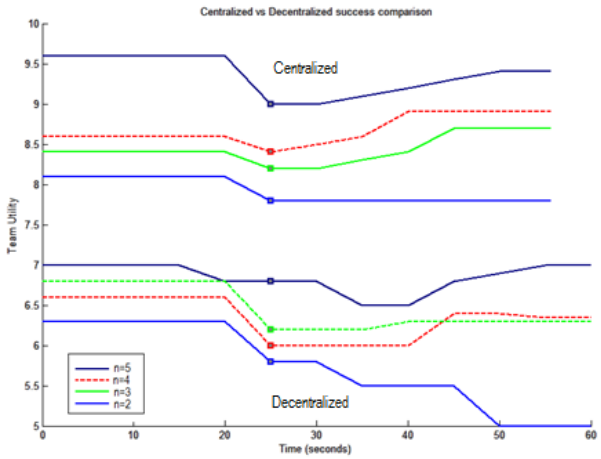


Figure. 4: Centralized vs. Decentralized Team Utility

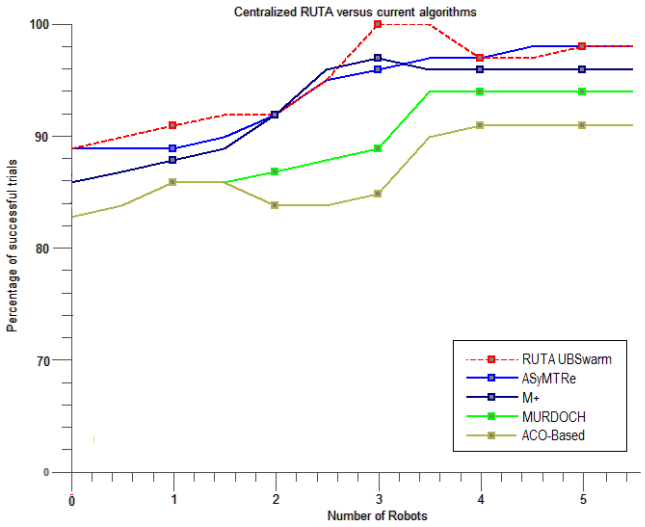


Figure 5: Comparison of RUTA with current methods

The input to the experiment is the set of robots, tasks, and the environment that they are operating in. However, by taking each of the previous system's utility equations and applying those in our centralized approach would give a proper comparison between our system and the current systems. As shown in fig. 5, the comparison is made by calculating the utilities of the systems for 15 human rescue trials performed using 1200gms dummy on teams that are composed of 2, 3, 4, and 5 robots respectively.

V. CONCLUSION

A task-oriented software application that facilitates the rapid deployment of multiple robotic agents is presented in this work. The task solutions are created at run-time and executed by the agents in a centralized or decentralized fashion. A core component of the system's framework is responsible for generating solutions when the robots are first deployed and during run-time to re-plan solutions to accommodate changes in the robot group. Tasks are fractioned into smaller sub-tasks which are then assigned to the optimal number of robots using a novel Robot Utility Based Task Assignment (RUTA) algorithm. We have also shown a reasoning algorithm that generates multi-robot utilities through a negotiation process in a decentralized/centralized manner. The negotiation process enables each robot to find the best solution by reassigning subtasks through the process of finding the utility of executing the specific sub-task. Compared with the centralized RUTA, the decentralized RUTA provides more fault-tolerance and flexible method for forming solutions. However, it trades off its solution quality.

REFERENCES

[1] Y. Xinan, L. Alei, and G. Haibing, "An algorithm for self-organized aggregation of swarm robotics using timer," in *Swarm Intelligence (SIS), 2011 IEEE Symposium on*, 2011, pp. 1-7.

[2] L. Bayindir and E. Sahin, "A review of studies in swarm robotics," *Turkish Journal of Electrical Engineering,* vol. 15, pp. 115-147, 2007.

[3] N. Kalde, O. Simonin, and F. Charpillet, "Comparison of Classical and Interactive Multi-Robot Exploration Strategies in Populated Environments," *Acta Polytechnica,* vol. 55, pp. 154-161, 2015.

[4] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Swarm robotic odor localization," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2001, pp. 1073-1078.

[5] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots,* vol. 11, pp. 319-324, 2001.

[6] T. Abukhalil, M. Patil, and T. Sobh, "Survey on Decentralized Modular Swarm Robots and Control Interfaces," *International Journal of Engineering (IJE),* vol. 7, p. 44, 2013.

[7] M. Patil, T. Abukhalil, and T. Sobh, "Hardware Architecture Review of Swarm Robotics System: Self-Reconfigurability, Self-Reassembly, and Self-Replication," *ISRN Robotics,* vol. 2013, 2013.

[8] S. Patel and T. Sobh, "Manipulator Performance Measures - A Comprehensive Literature Survey," *Journal of Intelligent & Robotic Systems,* pp. 1-24, 2014/02/15 2014.

[9] A. Y. Elkady, M. Mohammed, and T. Sobh, "A New Algorithm for Measuring and Optimizing the Manipulability Index," *J Intell Robot Syst,* vol. 59, pp. 75-86, 2010/07/01 2010.

[10] A. ElSayed, E. Kongar, S. M. Gupta, and T. Sobh, "A Robotic-Driven Disassembly Sequence Generator for End-Of-Life Electronic Products," *Journal of Intelligent & Robotic Systems,* vol. 68, pp. 43-52, 2012.

[11] M. INC, "Documentation & Technical Support for MobileRobots Research Platforms," ed: Adept MobileRobots INC 2006.

[12] J.-C. Baillie, "The URBI Tutorial," ed: Gostai, 2006.

[13] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, 2003, pp. 317-323.

[14] A. Elkady, J. Joy, and T. Sobh, "A plug and play middleware for sensory modules, actuation platforms and task descriptions in robotic manipulation platforms," in *Submitted to Proc. 2011 ASME International Design Engineering Technical Conf. and Computers and Information in Engineering Conf.(IDETC/CIE'11)*, 2011.

[15] S. S. Nestinger and H. H. Cheng, "Mobile-R: A reconfigurable cooperative control platform for rapid deployment of multi-robot systems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 52-57.

[16] B. Chen, H. H. Cheng, and J. Palen, "Mobile-C: a mobile agent platform for mobile C/C++ agents," *Software: Practice and Experience,* vol. 36, pp. 1711-1733, 2006.

[17] L. E. Parker, "ALLIANCE: An architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on,* vol. 14, pp. 220-240, 1998.

[18] L. Jiang and R. Zhan, "An autonomous task allocation for multi-robot system," *Journal of Computational Information Systems,* vol. 7, pp. 3747-3753, 2011.

[19] S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 1999, pp. 1234-1239.

[20] B. P. Gerkey and M. J. Matarić, "Murdoch: Publish/subscribe task allocation for heterogeneous agents," in *Proceedings of the fourth international conference on Autonomous agents*, 2000, pp. 203-204.

[21] F. Tang and L. E. Parker, "ASyMTRe: Automated synthesis of multi-robot task solutions through software reconfiguration," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1501-1508.

[22] F. Tang and L. E. Parker, "A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 3351-3358