This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication.

Task Allocation for Heterogeneous Robots Using a Self-Organizing Contextual Map

Matt Ross
School of Psychology
University of Ottawa
Ottawa, Canada
mross094@uottawa.ca

Pierre Payeur
Faculty of Engineering
University of Ottawa
Ottawa, Canada
ppayeur@uottawa.ca

Sylvain Chartier
School of Psychology
University of Ottawa
Ottawa, Canada
sylvain.chartier@uottawa.ca

***Abstract*—The Self-Organizing Contextual Map (SOCM) is applied to address the task allocation problem in the context of multi-robot heterogeneous systems. Inter-variability and overlap of common capabilities (sensors, actuators, and descriptors) exist between robotic agents. As a starting point, a range of search-and-rescue tasks are used to form a set of binary symbol codes that identify selected tasks. Binary vectors are then predefined to represent factitious features (task requirements) to solve these search-and-rescue tasks, forming the attribute codes. The combination of these vectors is used to form associations between the features of a task (requirements) and a robotic agent's capabilities. Using a recursive stepwise approximation and calculating the Euclidean distance, feature and contextual maps are formed to differentiate between tasks. Then similar predefined binary vectors are used that represent the robotic agent's capabilities. Using this information, nested domains are created on top of the contextual map that form allocation regions for all learned tasks. Three generalization tests are then performed to assess the SOCM's robustness and task allocation abilities. Overall, the results suggest that the applied SOCM encapsulates a true heterogeneous system for addressing a simplified version of the task allocation problem.**

Keywords—Task allocation, multi-robot systems, heterogeneous systems, self-organizing contextual map (SOCM), search and rescue.

I. Introduction

Multi-agent systems have emerged as a significant topic of interest within robotics due to their potential benefits for a large range of real-world applications, such as search and rescue tasks [1]. These larger tasks are typically divided into smaller subtasks that can be worked on by individual robotic agents or cooperative teams. However determining which robotic agents are suitable/capable of solving a given task can be challenging and is known as the task allocation problem [1], [2].

Task allocation is dependent on the characteristics of the robotic agents involved; they can either be homogeneous or heterogeneous [1], [3], [4]. Homogeneous systems comprise of identical agents with the same capabilities, sensors, and actuators. Within these systems, task allocation is typically not based on agent capability, but rather on additional factors like remaining battery or location [3]. This type of system is common when addressing the task allocation problem and has been applied in numerous ways. Some of these include market-based strategies, where task allocation is dependent on the incurring cost [5], and swarm approaches to name a few [1], [6]. However the task allocation problem becomes increasingly difficult in the context of heterogeneous systems, where significant inter-variability amongst agent capabilities and hardware can exist [1], [4], [7]. Furthermore, another dimension of complexity is added when there is any overlap in capabilities between agents in the system [1]. However, these systems offer versatility, flexibility, and practicality from a design perspective where having a single agent with all the necessary hardware is simply too arduous [1], [4].

Inspired from the formation of local assemblies in the brain [8], [9], the unsupervised Self-Organizing Map (SOM) has become a powerful tool due to its nonlinear topology preservation from a high-dimensional input space to a low-dimensional grid [10]–[12]. This endows the SOM with both dimension reduction properties [8], [12] and the clustering of input information that shares common features [9]–[12]. The SOM and its variations has been found in a myriad of applications including the control of robotic manipulators to representations of semantic relationships [10] (see [9] for a review).

In recent years the SOM has become an alternative approach to multi-task assignment due to the underlying competition during learning [13]. This competition allows winners, as well as their neighbors, to move closer to a given target providing a key component for the task allocation problem. The SOM has been applied to multi-robot systems for solving the task allocation problem for mobile wheeled robots [14] and autonomous underwater vehicles (AUVs) [13], [15], [16]. Although studies tackle other issues such as path planning [13] and dynamic environments [14], they often assume that the robotic agents are homogeneous. An inhomogeneous approach was used for AUVs and was defined as different battery types for agents [16]. However while this is still heterogeneous, we make the hypothesis that more variability between agents is needed to fully encapsulate a true heterogeneous system.

We therefore propose using a SOM that has been extended to higher levels of processing, the Self-Organizing Contextual Map (SOCM) [17], to solve the task allocation problem for a fully heterogeneous robotic system. The robotic agents in this system are defined as heterogeneous as they vary with respect to their capabilities (on-board sensors, actuators, and descriptors), but are permitted to overlap with some common features. The SOCM was initially proposed for the organization of a group of various animals with predefined features to form a meaningful animal map based on common features [17]. Here we applied the SOCM to task allocation for a heterogeneous unmanned robotic system to perform various search-and-rescue tasks with factitious features. In the current implementation the robustness and generalization qualities of the SOCM are studied to show some of the advantages of this method for task allocation.

With respect to the task allocation taxonomy, the current study is conducted considering a Non-Dependent-Single Robot-Single-Task Instantaneous Allocation (ND-SR-ST-IA) [2], [18]. Briefly stated this means that only one agent is assigned a single task at any given time with no consideration of future assignments [1], and task allocation is

dependent only on the capabilities of the agent and the given task requirements [2].

The reminder of this paper is divided as follows: Section II.A-B outlines the architecture and input patterns used by the SOCM. Section II.C describes the learning procedure for differentiation between tasks and the formation of the contextual feature map. Section II.D describes the post-learning phase where the contextual feature map is created for task allocation. Section III outlines various tests and their respective simulation results. Finally in Sections IV and V, we discuss and conclude the overall findings of our work.

II. METHODS

A. Architecture

The SOCM used here is comprised of two layers, a high dimensional input layer (23 dimensions per task) and a smaller 2-dimensional output layer, comprised of a 8x8 square lattice of units (sixty-four units). To prevent underfitting the data, the size of this output layer was estimated [19] according to (1).

$$\boldsymbol{M = 5\sqrt{Obs}} \tag{1}$$

where M is the number of units and Obs is the number of features observed for all tasks (130 total, see TABLE I.).

This estimation suggests a total of approximately 57 units in the output layer. However, to keep a consistent square lattice, this number was rounded up to 64 units (8x8). Each input vector is initially connected to every unit in the output layer via randomized synaptic weights between 0-1 (Fig. 1, where k∈[1-8]).

B. Input Patterns

The input patterns for the SOCM contain two components: the symbol and attribute codes. The symbol code is a vector of zeros the length of the list of all tasks, with a value of 1 identifying the selected task. This provides the network with the capability of extracting the corresponding hardware requirements from the symbol (label) only. The second component of the pattern comprises of the attribute code, which is directly encoded using a binary vector for each task that describes the presence or absence of the available hardware on a robot agent that is essential to completing that task (TABLE I.). These two codes are then concatenated (forming a 23 dimensional vector) to allow association between the features of search-and-rescue task (its requirements) and the capabilities of a robot (available hardware) according to (2):

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_s} \\ \boldsymbol{x_a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_s} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{x_a} \end{bmatrix} \tag{2}$$

where the x_s is the symbol code vector and x_a is the attribute vector. All input vectors are then normalized to a unit length of one prior to learning.

C. Learning: How the SOCM differentiates tasks

Learning for the SOCM takes place offline and is initialized with random weights for synaptic connections between the input layer and all units of the output layer (64 units). This means that initially the network has no knowledge on how to differentiate between tasks based on the available hardware mounted on a given robotic agent, and the hardware needed to complete a given task (see graph a) of Fig. 2).

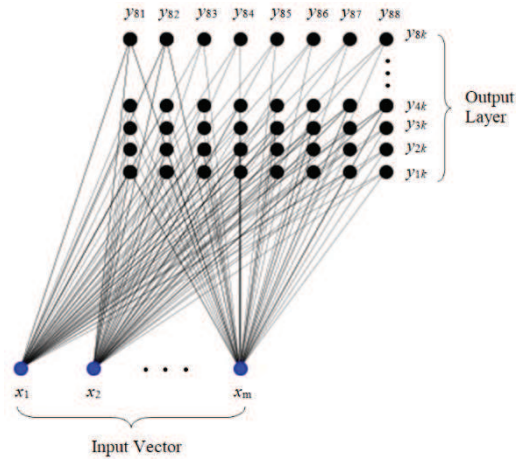


Figure 1. SOCM architecture. In the above architecture, each component of only one input vector projects to all units in the output layer. With m=23, and k ∈[1, 8].

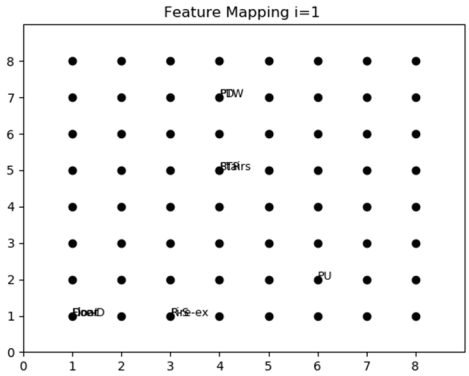
TABLE I. SEARCH AND RESCUE TASKS AND HARDWARE REQUIRED

Hardware Required (Features)		*Opening a door (Door)*	*Person unconscious (PU)*	*Person drowning (PD)*	*Fire detection (Fire-D)*	*Climbing stairs (Stairs)*	*Room search (R-S)*	*Person trapped –rubble (PTR)*	*Person trapped-water (PTW)*	*Flotation device (Float)*	*Fire extinguisher (Fire-ex)*
Sensors	Infrared	1	0	0	0	1	0	1	0	0	1
	Ultrasonic	1	0	1	0	1	0	0	1	1	0
	Camera	1	1	1	1	1	1	1	1	0	1
	CO_2 detector	0	0	0	1	0	0	0	0	0	1
Actuators	Pan-tilt	1	0	0	0	1	1	1	1	0	1
	Arm	1	0	1	0	0	0	0	0	0	1
	Deploy O_2	0	0	1	0	0	0	1	1	1	0
Descriptors	Mobile	1	1	0	0	1	1	1	1	1	1
	Fast	0	1	1	0	0	1	1	1	1	1
	Battery >3h	0	1	1	1	0	0	1	1	1	0
	Land	1	1	0	1	1	1	1	0	0	1
	Aquatic	0	0	1	0	0	0	0	1	1	0
	Small	0	0	0	0	0	0	1	1	0	0

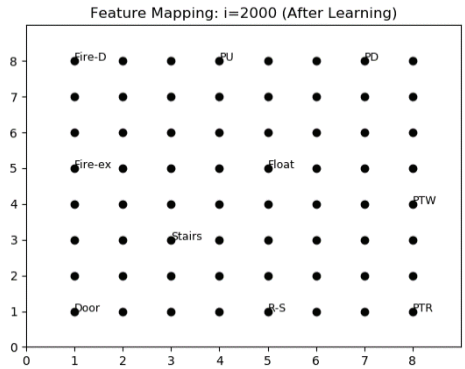
To differentiate between tasks the network uses a recursive stepwise approximation [9] by randomly selecting one of the input patterns and finds the "best matching unit" (BMU) from the output layer by computing the Euclidean distance (3) between the input vector and the weight vector of each unit in the output layer. The unit with the shortest distance is the winner.

$$\boldsymbol{i(x) = argmin_j \| x(n) - w_{j(n)} \|} \tag{3}$$

where $x(n)$ is the selected input at iteration n, $w_{j(n)}$ is the weight vector of a unit in the output layer, and $i(x)$ is the shortest Euclidean distance for that input (BMU).



a)



b)

Figure 2. Evolution of the feature map during learning: a) the output layer with overlaying feature map after 1 learning iteration, and b) the output layer with overlaying feature map after 2000 learning iterations. See TABLE I for abbreviations.

The BMU and its neighbors (other units that were excited) move closer in space to that input, making them more similar to that input. The size of the neighborhood around the BMU is calculated using a Gaussian function (Eq. 4, 5) with a decreasing standard deviation over time, according to (6), in order to allow the formation of distinct clusters.

$$h_{j,i(x)} = exp\left(-\frac{d_{j,i}^2}{2\sigma(n)^2}\right) \tag{4}$$

$$d_{j,i}^2 = \|r_j - r_i\|^2 \tag{5}$$

where $d^2_{j,i}$ is the difference between the position r_j of excited unit j in relation to the position of the BMU, r_i, for a two-dimensional lattice, with $\sigma(n)$ representing the radius (standard deviation) of the topological neighborhood.

$$\sigma(n) = \sigma_0 exp\left(-\frac{n}{\tau_1}\right) \tag{6}$$

where σ_0 is the initial radius of the topological neighborhood (set to a value of 5), n the current iteration, and τ_1 is a time constant calculated by (7).

$$\tau_1 = \frac{N}{log\sigma_0} \tag{7}$$

and here N is the total number of iterations (2000 in present study).

Weights are then updated for all excited units via Hebbian learning with the inclusion of a forgetting term and neighborhood function influence (8).

$$w_j(n+1) = w_j(n) + \eta(n)\, h_{j,i(x)}(n)(x(n) - w_j(n)) \tag{8}$$

where $w_j(n)$ are the weights for each unit j, $\eta(n)$ the learning rate, $h_{j,i(x)}(n)$ the influence of the neighborhood function, and $x(n)$ is the selected input, all at the current iteration n.

After updating, both the radius of the topological neighborhood (σ) and the learning rate (η) are permitted to decrease according to (6) and (9) respectively.

$$\eta(n) = \eta_0 exp\left(-\frac{n}{\tau_2}\right) \tag{9}$$

where η_0 is the initial learning rate set at 0.1 and τ_2 another time constant set to N (2000 here).

This whole process is repeated until $n = N$ iterations, that is until convergence and formation of the feature map (see graph b) of Fig. 2). Convergence was estimated by minimizing the quantization error (see Fig. 3) as calculated by (10).

$$QE(n) = \frac{1}{N}\sum_{i=1}^{N} \|x(n) - i(x)\| \tag{10}$$

where the quantization error (QE) is the difference between the input $x(n)$ and the distances, calculated in Eq. (3), from units in the output layer at iteration (n).

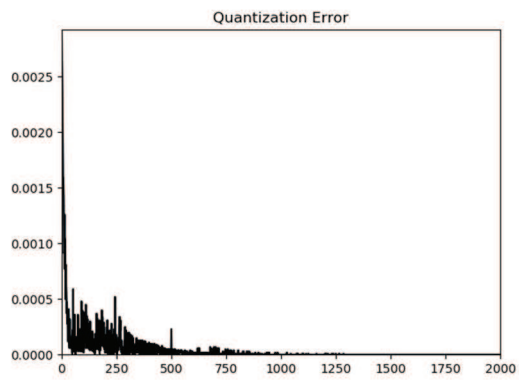


Figure 3. Quantization Error across iterations. Calculation of the distance between input data points and output units.

D. Post Learning: How the SOCM allocates tasks

After 2000 learning iterations, a contextual map is derived from the Euclidean distance (3) for the weights instead of inputs. This results in a topology for weight connections where the BMU for each weight vector is represented by one of the input tasks (Fig. 4). This allows the clear distinction of task-clusters based on the BMUs for each task and their respective topological neighborhoods. Once complete, information about the specialization of available robotic agents is used to create vectors in the same manner and size (23 dimensions) as the task-input (see TABLE II), with a zero vector for the symbol code. These "robot" vectors are then used to create nested domains on top on the contextual map (Fig. 5). This automatically assigns the previously-learned tasks to the best suited available robotic agent based on its specializations. All processes, from initialization to final map formation, take less than a minute (approximately 21.204 seconds) on an average desktop computer for offline training.

TABLE II. AVAILABLE HETEROGENEOUS ROBOTIC AGENTS

Available Hardware		*Robot # 1*	*Robot # 2*	*Robot # 3*
Sensors	Infrared	1	0	0
	Ultrasonic	0	1	1
	Camera	1	1	0
	CO2 detector	0	0	0
Actuators	Pan-tilt	1	0	0
	Arm	0	1	0
	Deploy O2	0	0	1
Descriptors	Mobile	0	1	0
	Fast	0	0	1
	Battery >3h	1	0	0
	Land	1	1	0
	Aquatic	0	0	1
	Small	0	0	0

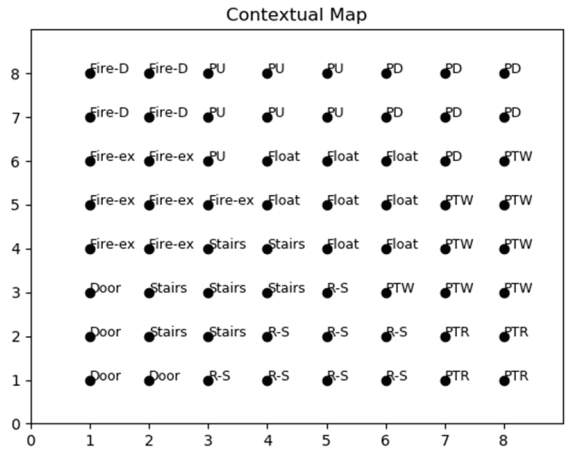


Figure 4. Contextual map of the 2-D topology for weight connections. Each task abbreviation represents the Best Matching Unit (BMU) for each weight vector. See TABLE 1 for abbreviations.

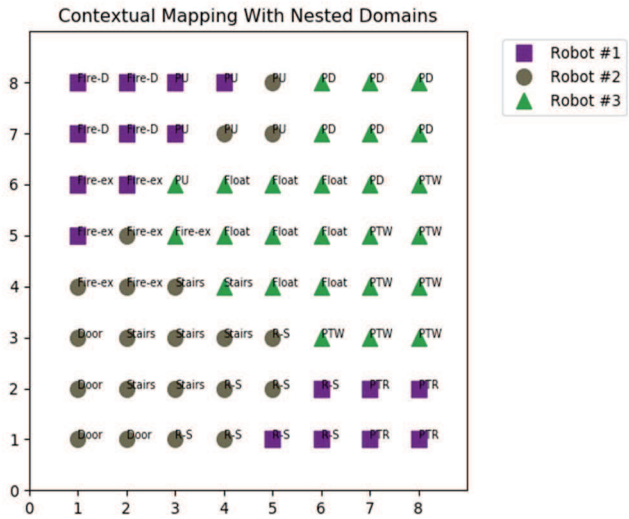


Figure 5. Contextual map with nested domains for each of the three specialized robotic agents available. See TABLE I for abbreviations and TABLE II for nested robotic domain details.

III. RESULTS

The effectiveness of the SOCM for solving the task allocation problem was evaluated by administering several different generalization tests. The first one involved allocating the previously learned task of "Opening a door" to the best matching robot. The trained network was presented with an exact replica of the input vector (23 dimensions) that contained both the symbol code and attribute code with associated features (see TABLE I) for the "Opening a door" task. The network finds the BMU for the task and identifies the associated nested robotic domain. The network identifies, based on capability, that Robot #2 is the best suited available specialized agent for the task. This is reflected visually in Fig. 6 marking the BMU (red "X") within the domain of Robot #2 (gray circle). It is important to note that by comparing TABLE I and TABLE II, Robot #2 shares the most, but not all, features with the "Opening a door" task, making it the best suited available agent.

To show the robustness of the SOCM, the second and third tests involved allocating novel tasks that the network was not trained on. The second test involved presenting the network with an input vector containing only the symbol code for the "Climbing stairs" task and none of the associated required hardware (an attribute code of all zeros). The SOCM identifies the BMU for the task and the associated nested domain, that of Robot #2. This is reflected visually in Fig. 7 marking the BMU within the domain of Robot #2. Again, it is important to note that by comparing TABLE I and TABLE II, Robot #2 only shares some common features with the "Climbing stairs" task, but is still the best suited available agent based on both the network's outputted solution and an external check.

Finally, the third test involved presenting the network with a completely novel task that the network was not trained with to determine if it was able to provide a proper allocation. The vector that was presented to the network contained an all zero symbol code and only the following attribute code information regarding the hardware required to solve the task: fast, battery life>3 hours, and is aquatic. The network determines that this task is most similar to the "Person Drowning" task sharing many, but not all, features (see TABLE I). The BMU is identified in the nested domain of Robot #3 (green triangle), as reflected in Fig. 8.

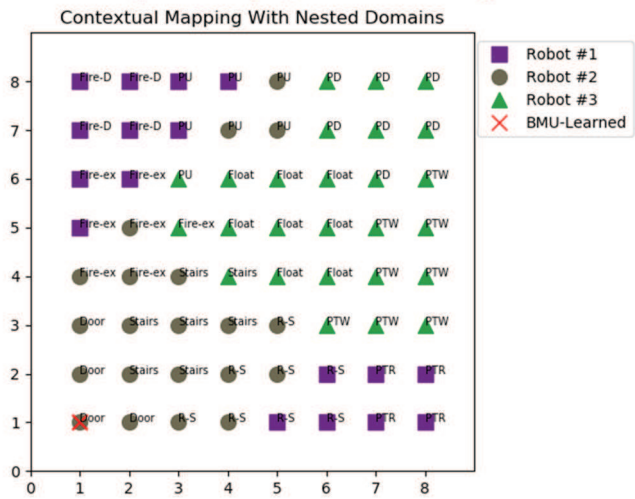


Figure 6. Contextual map with nested robotic domains for allocation of a previously learned task. Identification of BMU and associated domain for the learned "Opening a door" task (red "X"). See TABLE I for abbreviations and TABLE II for nested robotic domain details.

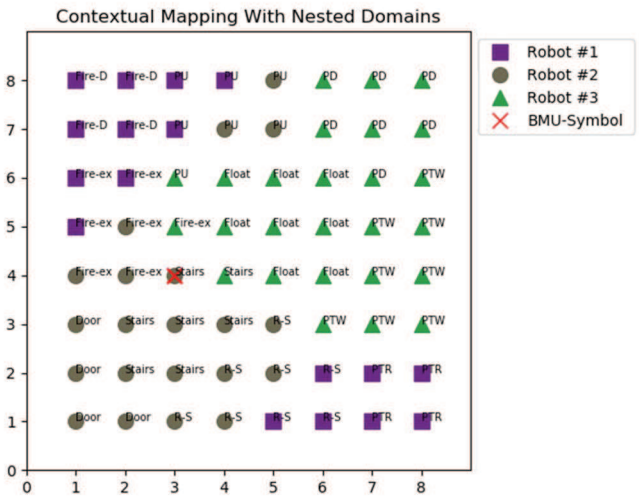


Figure 7. Contextual map with nested robotic domains for allocation of a symbol only task. Identification of BMU and associated domain when given only the symbol for "Climbing stairs" and no hardware features (red "X"). See TABLE I for abbreviations and TABLE II for nested robotic domain details.

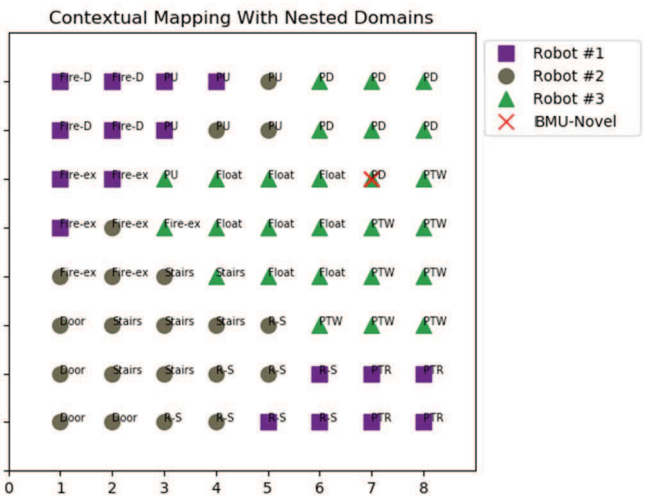


Figure 8. Contextual map with nested robotic domains for allocation of a novel task. Identification of BMU and associated domain when presented with novel task with hardware features of: fast, has battery life>3 hours, and is aquatic (Red "X"). See TABLE I for abbreviations and TABLE II for nested robotic domain details.

IV. Discussion

The identification of an effective mechanism for task allocation that aims at determining which robotic agents are capable of solving a given task in a multi-robot system has been the root of many approaches [20], [21]. With the aforementioned advantages of using a heterogeneous system, we believe that a heterogeneous approach to the task allocation problem merits further exploration. We therefore sought to explore the task allocation problem using heterogeneous system from an artificial neural network standpoint using the SOCM.

As illustrated in our results, the SOCM was able to form a contextual map with clear differentiation between tasks based on the required hardware and the capabilities of robotic agents to solve these tasks. In this alternative approach to the actual task allocation problem, it is important to note the success of the network's ability to differentiate between highly correlated tasks. For example the tasks of "Opening a door" and "Climbing stairs" were highly correlated with only one difference in the hardware required, with the "Opening a door" task additionally requiring an arm (TABLE I). Despite this high degree of similarity, distinct separate clusters were still observed (Fig. 4). This could be in part due to the orthogonal nature of the symbol codes lowering the correlation between these tasks, thus making it easier for the network to identify different BMUs.

Many approaches to task allocation involve the decomposition of larger tasks into smaller subtasks prior to allocation, such as with some market-based approaches [22]. Therefore, the incorporation of an effective mechanism for differentiating between tasks can be considered necessary to allocation. In regards to the actual allocation of tasks, we show that nested domains of heterogeneous robots are easily created on top of the contextual map forming allocation regions. The formation of nested domains for approaching this problem is in line with previous work where a centralized algorithm is used to form allocation regions for each agent based on robot speed profiles [23].

The use of the SOCM can be considered as fast as any classical classification technique. The application of nested domains in combination with the contextual map allow for the identification of the BMU and allocation of learned tasks to the most suitable robotic agents within fractions of a second on an average desktop computer. It took an average of 0.009 seconds to perform task allocation, excluding training, for the three cases presented here. Furthermore the robustness of the SOCM was validated with reasonable allocation of a task with the minimal information of just a symbol, and with a completely novel task.

While the SOCM has shown success with the task allocation problem, some questions still remain. For instance, in the condition where the task is unsolvable due to no agent having the needed capabilities would the SOCM still allocate an agent? From additional experiments conducted within the current framework, the SOCM would still allocate to the next suitable agent based on any remaining similarities between task requirements and agent capabilities, even though the agent would not be able to fully solve the task. In the search-and-rescue scenarios considered here, this is not seen entirely as a limitation. For example, in the case of a person drowning, an agent might not have the capability to dispense oxygen or an arm to lift the person out of the water. However, it is better to send an agent that might not be fully capable to save a drowning person than none at all. However, we believe that this is a situation-based problem and future refinements are required to incorporate when an agent should still be allocated despite a mismatch in required capabilities and task requirements.

V. Conclusion

The overall aim of this paper was to examine the application of the SOCM as a mechanism for task allocation in a heterogeneous unmanned robotic system. This work shows that the applied SOCM encapsulates a true heterogeneous system with inter-variability amongst its members for addressing the task allocation problem. Even though the representation considered for the task allocation problem in this initial experimental study focuses on relatively simple tasks, it clearly shows the potential of the application of SOCM to develop formal associations between tasks and robotic agents that must share common characteristics.

More complex tasks may have many potential solutions and require further cooperation of multiple robots [21]. One potential solution would be to incorporate a k-winners-take-all (*k*WTA) rule to produce a more distributed classification [24]. Furthermore, Monte-Carlo or bootstrap methods could be used to assess the stability of quantization in the SOCM [25]. This would allow the selection of more reliable maps, which is an important consideration for applications to sensitive problems such as search-and-rescue, where increased reliability is an essential asset for such dynamic problems. Finally, we aim to incorporate weighted inputs (instead of just binary) to put emphasis on specific features that may be more critical for solving a given task [17].

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support from Department of National Defence of Canada toward this research under the Innovation for Defence Excellence and Security (IDEaS) program, and from the Ontario Graduates Scholarship (OGS).

REFERENCES

[1] L. E. Parker, "Multiple Mobile Robot Systems," in *Springer Handbook of Robotics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 921–941.

[2] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Rob. Res.*, vol. 32, no. 12, pp. 1495–1512, Oct. 2013.

[3] A. Nikitenko, E. Lavendelis, M. Ekmanis, and R. Rumba, "Task Allocation Methods for Homogeneous Multi-Robot Systems: Feed Pushing Case Study," *Autom. Control Comput. Sci.*, vol. 52, no. 5, pp. 371–381, Sep. 2018.

[4] D. Di Paola, D. Naso, and B. Turchiano, "Consensus-based robust decentralized task assignment for heterogeneous robot networks," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 4711–4716.

[5] E. Lavendelis, A. Liekna, A. Nikitenko, A. Grabovskis, and J. Grundspenkis, "Multi-Agent Robotic System Architecture for Effective Task Allocation and Management," in *Proceedings of the 11th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA '12)*, Cambridge, UK, pp. 167-174, Feb. 2012.

[6] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized Stochastic Policies for Task Allocation in Swarms of Robots," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 927–937, 2009.

[7] A. Khamis *et al.*, "Cooperative Robots and Sensor Networks," *Stud. Comput. Intell.*, vol. 604, 2015.

[8] X. Lin, D. Soergel, and G. Marchionini, "A self-organizing semantic map for information retrieval," in *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '91*, pp. 262–269, 1991.

[9] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, Jan. 2013.

[10] G. A. de Barreto, A. F. R. Araújo, and H. J. Ritter, "Self-Organizing Feature Maps for Modeling and Control of Robotic Manipulators," *J. Intell. Robot. Syst.*, vol. 36, no. 4, pp. 407–450, 2003.

[11] J. Himberg, J. Ahola, E. Alhoniemi, J. Vesanto, and O. Simula, "The Self-Organizing Map as a Tool in Knowledge Engineering," 2001, pp. 38–65.

[12] S. Haykin *et al.*, *Neural Networks and Learning Machines Third Edition*. 2009.

[13] D. Zhu, X. Cao, B. Sun, and C. Luo, "Biologically Inspired Self-Organizing Map Applied to Task Assignment and Path Planning of an AUV System," *IEEE Trans. Cogn. Dev. Syst.*, vol. 10, no. 2, pp. 304–313, Jun. 2018.

[14] A. Zhu and S. X. Yang, "A Neural Network Approach to Dynamic Task Assignment of Multirobots," *IEEE Trans. Neural Networks*, vol. 17, no. 5, pp. 1278–1287, Sep. 2006.

[15] D. Zhu, H. Huang, and S. X. Yang, "Dynamic Task Assignment and Path Planning of Multi-AUV System Based on an Improved Self-Organizing Map and Velocity Synthesis Method in Three-Dimensional Underwater Workspace," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 504–514, Apr. 2013.

[16] M. Chen and D. Zhu, "A Workload Balanced Algorithm for Task Assignment and Path Planning of Inhomogeneous Autonomous Underwater Vehicle System," *IEEE Trans. Cogn. Dev. Syst.*, 2018.

[17] H. Ritter and T. Kohonen, "Self-Organizing Semantic Maps," *Biological cybernetics*, vol. 61, no. 4, pp. 241-254, 1989.

[18] B. P. Gerkey, M. J. Mataric, and M. J. Matari, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *Int. J. Rob. Res.*, vol. 23, 2004, pp. 939-954.

[19] J. Tian, M.H. Azarian, and M. Pecht "Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm," in *Proceedings of the European Conference of the Prognostics and Health Management Society, pp. 1-9, 2014.*

[20] R. Zlot and A. Stentz, "Complex Task Allocation For Multiple Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1515–1522, 2005.

[21] O. Al-Buraiki, P. Payeur, "Agent-Task Assignation Based on Target Characteristics for a Swarm of Specialized Agents," in *Proceedings of the 13th Annual IEEE International Systems Conference*, Orlando, FL, pp. 268-275, Apr. 2019.

[22] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-Based Multirobot Coordination: A Survey and Analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, Jul. 2006.

[23] J. Camilo, G. Higuera, and G. Dudek, "Fair subdivision of Multi-Robot Tasks," *Robot. Autom. (ICRA), 2013 IEEE Int. Conf.*, pp. 3014–2019, 2013.

[24] S. Chartier, G. Giguere, D. Langlois, and R. Sioufi, "Bidirectional Associative Memories, Self-Organizing Maps and k-Winners-Take-All: Uniting feature extraction and topological principles," in *2009 International Joint Conference on Neural Networks*, pp. 503–510, 2009.

[25] M. Cottrell, E. de Bodt, and M. Verleysen, "A Statistical Tool to Assess the Reliability of Self-Organizing Maps," in *Advances in Self-Organising Maps*, London: Springer London, pp. 7–14, 2001.