Autonomous Hierarchical Multi-Level Clustering for Multi-UAV Systems

Jonathan Ponniah*
San Jose State University, San Jose, CA 95192

Mirco Theile†, Or D. Dantsker‡, and Marco Caccamo §
Technical University of Munich, Garching, Germany

Abstract

The strategy of clustering is introduced to enable coordination in decentralized multi-UAV systems. Each cluster is an organized unit comprised of several cluster-members and one cluster-head. We propose the concept of multi-level clustering, in which cluster-heads form successively higher-level clusters, resulting in a tree-shaped hierarchy. Multi-level clustering provides a mechanism for aggregating local states and disseminating the information needed for system coordination. Related work shows that aggregate information is beneficial for efficient UAV path planning using reinforcement learning. We propose rules for scalable multi-level cluster-formation, taking into consideration the computational and communication loads associated with cluster maintenance and information aggregation and dissemination. The viability of the proposed concept is demonstrated in preliminary simulations. The scenarios considered examine the effects of agent motion, take-off, and landing on multi-level clustering. The simulation results show that multi-level clustering is robust to the dynamics of multi-UAV environments.

I. Introduction

Autonomous multi-UAV systems are increasingly relevant in many emerging applications that lack pre-existing wireless infrastructure. Usually, these applications involve spatially-distributed, time-varying "points-of-interest" that must visited under certain time or energy constraints, with the objective of sending UAVs to as many points as possible subject to the given constraints. Examples include wildfire surveillance, in which UAVs must locate burning trees in a forest fire, or post-disaster provisioning of wireless connectivity in which UAVs act as temporary access points to first responders in the aftermath of an event. Points-of-interest correspond to burning trees in the first scenario and first responders in the second.

Ideally, any proposed system should scale with the number of UAVs and distribution of interest-points. In general, communication and planning problems in multi-agent systems are not scaleable. Without pre-existing wireless infrastructure, the UAVs themselves must discover multi-hop routes over which to exchange messages, a computationally expensive task since the network topology at any instant, depends on the changing relative positions and velocities of the UAVs. Moreover, single-agent planning problems are challenging in their own right, making the multi-agent problem even more daunting.

The class of applications described above, modeled by spatially-distributed points-of-interest, has a particular structure that simplifies the problem complexity. Consider the example of wildfire surveillance. For the UAVs to effectively track the spread of fire propelled either by wind or the tree density, it suffices intuitively for each UAV to

*Assistant Professor, Department of Electrical Engineering. jonathan.ponniah@sjsu.edu
†Ph.D. Student, Department of Mechanical Engineering, mirco.theile@tum.de
‡Researcher, Department of Mechanical Engineering, or.dantsker@tum.de
§Professor, Department of Mechanical Engineering, mcaccamo@tum.de

know the precise positions of the burning trees in its vicinity but only the aggregated fire intensity in relatively remote regions. UAVs can thus independently survey/monitor the events in their regions, provided the situation in other regions remains under control; spreading fires should first attract the UAVs nearby, and when large enough, the UAVs that are further away. For the system to respond in a coordinated way to changes in the distribution of interest-points, each agent requires the profile of aggregated states calibrated to itself, instead of the raw local states of all other agents.

The strategy of only disseminating the information needed for multi-agent coordination has previously appeared in the robotics literature without the extra assumption of spatially-distributed interest-points. Instead, the basic, unvarnished Markov Decision Process (MDP) is used, under the assumption that all intra-UAV communication is direct (point-to-point) and limited. The other extreme is the ad-hoc networking literature, which largely ignores application-specific features of the transmitted data and focuses instead on connecting any pairs of agents through multi-hop routes.

This paper is part of a body of work that aims to bridge both extremes by narrowing the class of applications to those that can be modeled by spatially-distributed points-of-interest while also allowing multi-hop routing for communication. The key idea is the notion of a ”cluster” that simultaneously reduces the planning and communication problems' complexity.

A cluster of agents, UAVs in this case, consists of cluster-members and a duly elected cluster-head connected through a ”star” topology, where the cluster-head schedules all communication between its cluster members (intra-cluster communication). The notion of a cluster also appears in the VANET (Vehicular Ad-Hoc Networking) literature as a tool for preventing vehicular collisions by designating as cluster-head, the vehicle with the most stable communication channels between itself and the cluster members. When there are more agents than can reasonably fit into one cluster, agents divide themselves into multiple independent clusters with their neighbors.

This paper envisions a different role for agent clusters, namely as the primary mechanism for information aggregation and dissemination in decentralized systems. The concept of multi-level clustering is proposed to connect all agents under a single hierarchical tree and prevent independent, disconnected clusters. In its most rudimentary form, multi-level clustering unfolds in the following way: first agents organize themselves into clusters and elect cluster-heads; then the cluster-heads organize themselves into ”macro-clusters” (or level-2 clusters) and elect level-2 cluster-heads; the process repeats for higher levels until all the agents are finally connected in a tree of arbitrary depth. Cluster-heads orchestrate all intra-cluster communication at each level and serve as gateways, regulating the information passing in and out of their clusters (inter-cluster communication). It is essential to recognize that clustering merely responds to the agent trajectories without directly affecting them. As far as the authors are aware, multi-level clustering has never been proposed before; clustering algorithms in the literature are all single-level.

Multi-level clustering is difficult because UAVs are highly mobile; UAVs may drift out of range, effectively ending their participation in a functioning unit. Changes that occur at any level of the communication tree can also affect the proper functioning of clusters at other levels. To preserve cluster integrity, much of the literature is invested in finding the optimal (most stable) clustering *before* run-time. This paper proposes a fundamentally different philosophy, based on periodic verification *during* run-time. The idea is to accept a ”quick-and-dirty” initial clustering, one that is likely to be sub-optimal, even crudely so, intending to successively improve this clustering over periodically scheduled maintenance/verification phases. Under the right conditions, the initial clustering may converge to the optimal clustering in steady-state.

To be viable, multi-level clustering includes mechanisms that allow clusters to merge or split, supporting the organized transfer of agents from one cluster to the next, and that keep agents connected to the tree even when they leave their clusters. These mechanisms allow the system to adapt and adjust, to maintain a stable superstructure that connects all UAVs despite trajectory-induced changes to the exposed topology. This paper provides a high-level introduction of the algorithm and preliminary simulation results to whet the appetite for a more in-depth simulation and analysis to follow.

The rest of the paper is organized as follows. Section II provides some background along with an overview of the relevant literature, Section III introduces some of the basic concepts of multi-level clustering, Section IV includes preliminary simulation results that demonstrate the viability of multi-level clustering. Section V concludes with a summary of future work.

II. Background and Related Literature

The objective of multi-level clustering is to reduce the complexity of multi-agent motion planning, by disseminating minimally required local state information for global coordination. System states evolve according to some Markov Decision Process (MDP).[1] Other components of an MDP include actions, "state transition probabilities", and a reward function which assigns a real-valued "reward" to each state-action pair. State-transition probabilities correspond to the system dynamics, and describe the probability of moving to some state in the next time-instant given the state and selected action in the present time-instant. The system is Markovian in the sense that the future only depends on the past through the present.

A "policy" is a rule for mapping specific states to assigned actions. For example, UAVs could have a policy of visiting burning trees within some fixed radius of their current position, provided that the intensity of the fires in remote regions remains beneath some threshold. Every policy will have an expected discounted reward accrued over its operating lifetime. The optimal policy, by definition, achieves the maximum discounted reward.

Reinforcement learning[2] is a way of "solving" the optimal policy without knowledge of the state-transition probabilities. Some relevant applications of reinforcement learning in multi-UAV systems with coordination constraints, include wildfire surveillance[3,4] and data harvesting.[5] In many cases, finding the optimal policy is computationally expensive,[6] having exponential complexity in the state-space dimension.[7]

In certain applications, state-spaces have "locally-interactive" structure; the optimal action of any particular agent depends on a combination of the specific events in its immediate vicinity and the aggregate events occurring elsewhere. This structure can be exploited,[8] to find approximately optimal policies with faster convergence times.[9] One such approach that appears in the literature is called state decomposition.[8] It is shown that compressed global information combined with detailed local information can be used effectively for single UAV path planning,[10] as well as multi-UAV path planning.[5] In the multi-UAV problem, agents can infer the global view by communicating with the other agents through a multi-level hierarchy.

State decomposition also simplifies the communication problem.[11] If the state-space is locally-interactive, then each agent need only know the aggregate states of remote agents instead of the precise local states of every individual agent.[12] Exploiting this structure reduces the communication burden on the system as a whole.[13] Strategies based on this approach appear in the literature for general MDPs,[14,15] contingent on the assumption that any locally-interactive structure is known a priori.[16] In practice, finding this structure is a non-trivial exercise that must be repeated for each application, since the state-space itself is application-specific. Augmenting the traditional MDP with spatially-distributed interest-points, has the advantage of making this structure explicit, which simplifies the planning problem. The multi-level clustering communication protocol in this paper is proposed with these augmented state spaces in mind.

An alternative approach (not taken here) is to separate the communication problem from the planning problem; Ad-Hoc Networks (MANETs) have been studied extensively in the literature,[17] outside the context of motion-planning. There are several ways of classifying ad-hoc networking protocols in the literature. Protocols are pro-active,[18] when routing tables are consistently maintained, and reactive when routing tables are updated as needed.[19] Link-state routing,[20] allows each agent to see the entire communication graph, whereas distance-vector routing[21] only reveals the next hops to each agent. These protocols, which prioritize discovery of shortest routes, are not designed to enable coordination in multi-agent systems or disseminate state information.

The notion of clustering,[22] has also been studied in vehicular ad-hoc networks (VANETs).[23] Clusters consist of agents that share relatively stable communication channels for low-bandwidth, delay-sensitive VANET applications like collision avoidance[24] and vehicle platooning.[25] Metrics for creating stable clusters usually depend on the agents' relative positions and velocities,[26] and do not incorporate a "system objective" such as tracking the spread of wildfire. The literature appears confined to single-level clustering, where each cluster is independent and effectively disconnected from the others.

III. Concept

The following section describes some fundamental clustering rules and concepts that ensure disconnected groups of decentralized, asynchronous agents can form a stable functioning network, despite the differences in their take-off/landing times and the dynamics of their motion. The proposed ”hierarchical multi-level clustering” approach, takes agents from this initial uncoordinated state, to a steady-state in which full system-wide coordination is achievable.

A. Decentralized to Hierarchical

At the moment of spawning, which occurs at take-off, each agent finds itself alone; it knows of no other neighbors and belongs to no cluster. Hence, the immediate tasks are to both discover whether any neighboring agents or agent clusters exist in the vicinity and to alert these agents of its presence. Newly spawned agents accomplish both tasks by first broadcasting ”hello” messages that include their positions while listening for other agents' ”hello” messages, sharing this information. From the ”hello” messages, agents create lists of their neighboring agents and clusters.

The next task is to either initiate cluster formation with neighboring unclustered agents or, if available, join a neighboring cluster that is not at capacity. After successfully joining or creating a cluster, coordination of communication is handled by a cluster-head. The cluster-head of this level-1 cluster is chosen according to its close proximity to the centroid of the agents' positions in the cluster. Cluster-heads repeat the decentralized process, searching for other same level cluster-heads to create a cluster of one level higher, or joining existing clusters that are exactly one level higher. Repeated, this yields the desired hierarchy, in which cluster-heads can schedule internal communication, leading to more efficient synchronous communication schemes.

B. Clustering Rules

We define rules for the proposed multi-level clustering to improve the applicability of the clustering in real-world environments. Their rationale and resulting consequences are elaborated in the following.

1. Every agent can only be a cluster-head of one cluster.

This rule's rationale is that a cluster-head performs tasks for maintenance and cluster verification to maintain and improve the clustering and tasks for aggregating and disseminating mission-critical information to its members and superiors. These tasks require computational and communication capacities. If agents are allowed to be cluster-heads of multiple levels of clusters, their onboard hardware needs to be provisioned for the worst-case computational and communication load. This worst-case does not have an upper bound if the number of agents and the operating area are unbounded, because an agent could be a cluster-head of an unbounded number of clusters. Furthermore, if the number of agents and operating area are bounded, the worst-case load can still be significantly higher than the average case load. In both cases agents are required to be drastically over-provisioned in terms of computational hardware and communication resources. To avoid this, we do not allow agents to be cluster-heads of multiple clusters, balancing the loads among all agents.

2. Every cluster-head has to be a member of a level-1 cluster.

This rule's rationale is that every agent, cluster-head or idle, still conducts its regular mission tasks. Thus cluster-heads have to be members of level-1 clusters to receive and report the same level of aggregate information. This allows for homogeneous multi-agent systems, simplifying many planning and control problems.

3. A cluster-head of any level has to be a level-1 member of its own sub-tree.

The tree in this rule refers to all agents that are connected through multi-level clustering. The sub-tree of one cluster-head contains all of its descendants. This rule is an extension to *Rule 2*, adding that the cluster-head has to be a leaf in its sub-tree. If agents are non-moving, this rule is enforced by through *Rule 1* and *Rule 2*, as the agents can only be

selected as cluster-heads if they are idle members of the tree. However, this rule is not automatically enforced through the previous rules if agents are moving as they can change level-1 cluster membership, potentially leaving the sub-tree. Without *Rule 3*, action directives decided by a cluster-head for its cluster may conflict with action directives it receives as level-1 member of an external tree. This conflict creates ambiguity in action selection when applying the concept of joint actions for multi-agent planning. As joint actions will be an essential element of hierarchical planning, which is the ultimate goal of this research direction, *Rule 3* eliminates the potential hurdle.

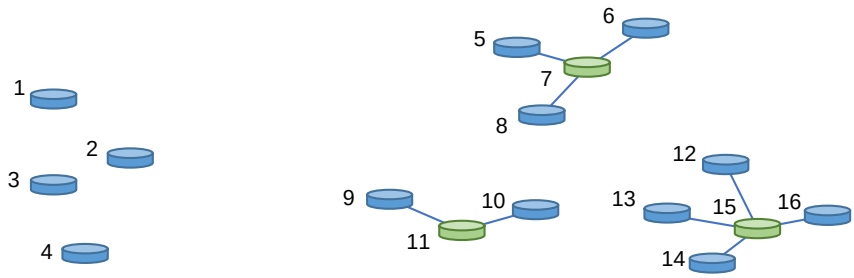
C. Clustering Example and Tree View

Figure 1 illustrates multi-level clustering in two groups of agents. It shows the formation of one higher-level cluster in both groups, starting from an unclustered state in the first group and a level-1 clustered state in the second.

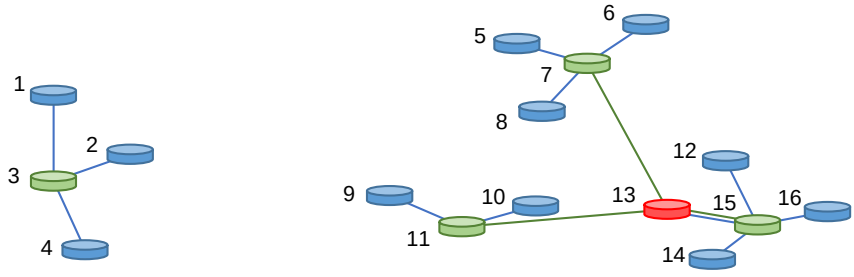
Consider agents 1 to 4 in the unclustered group in Figure 1a and assume, due to their mutual proximity, that each agent has received ”hello” messages from the others. The agents are thus ready to initiate cluster formation. The protocol guarantees that one of the four agents will successfully inform the other three of their impending membership in a new cluster and, given its knowledge of their positions, elevate the agent closest to the centroid as cluster-head. In Figure 1a, agent 3, being the closest agent to the centroid, is elevated to the head of the cluster.

This method, which creates single-level clusters, can be further extended to create multi-level clusters, as shown in Figure 1 for agents 5 to 16. Assume the level-1 cluster heads 7, 11, and 15, marked in green, have exchanged ”hello” messages due to their level-1 proximity, and are ready to form a level-2 cluster. As before, the protocol guarantees that one of the three will inform the others of their impending membership in a new level-2 cluster. However, as stipulated in *Rule 1*, the initiating agent does not designate any of these agents as the level-2 cluster head; these agents already have clusters to manage and are not allowed to manage more. Instead, the initiating agent queries their lists of idle level-1 cluster members (the agents that are not heads of any clusters) and elevates the idle agent closest to the centroid of the level-1 cluster-heads, to the head of the level-2 cluster, which in Figure 1b, turns out to be agent 13.

Multi-level clustering results in a tree-structured hierarchy. Figure 2 shows resulting trees from both groups of agents in Figure 1b. Observe that every node in the tree is also a leaf within the same tree, as stipulated in *Rule 2*.



(a) Two groups of agents being unclustered (left) and level-1 clustered (right).



(b) Two groups of agents being level-1 clustered (left) and level-2 clustered (right).

Figure 1: Multi-level clustering examples.

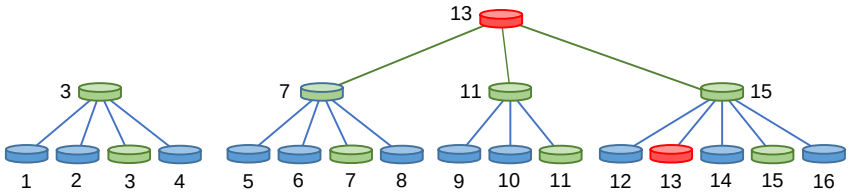


Figure 2: Tree view of the agents in Figure 1b.

IV. Clustering Scenario Examples

The algorithm used for the autonomous hierarchical multi-level clustering is a prototype that will be improved and elaborated in detail in follow-up work. The basic principle of the algorithm is that, when unclustered, agents alternate *join* and *create* phases. In the *join* phase, agents try to *join* existing clusters which are close enough and not at capacity. If the agent does not find a cluster to *join*, it switches to the *create* phase in which it tries to find other unclustered agents in their *create* phase to form a new cluster. When a new cluster is formed, the agent closest to the centroid is elected as cluster-head, receiving a *cluster-token*. This *cluster-token* contains the necessary cluster information such as the member list and higher cluster heads. The idea of the *cluster-token* is that it can be passed to another agent if a better cluster head is chosen or the current token holder de-spawns.

Once the cluster-head is elected, receiving the *cluster-token*, this agent starts another *join-create* cycle to *join* or *create* higher-level clusters. Furthermore, the cluster management, such as choosing a new cluster-head, evicting agents that surpass the distance threshold to the cluster-head, rebalancing sub-clusters, and more, can be conducted by the cluster-head directly. This way, the initially fully decentralized clustering converts to an autonomous hierarchy with substructure management.

This section shows four scenario examples of clustering under different motion and spawning patterns. The first example shows stationary agents that are not spawning. The second adds spawning and de-spawning to the stationary agents, while the third adds spiral motion to the non-spawning agents. The fourth example combines the spiral motion with the spawning and de-spawning behavior, being closest to a real-world scenario.

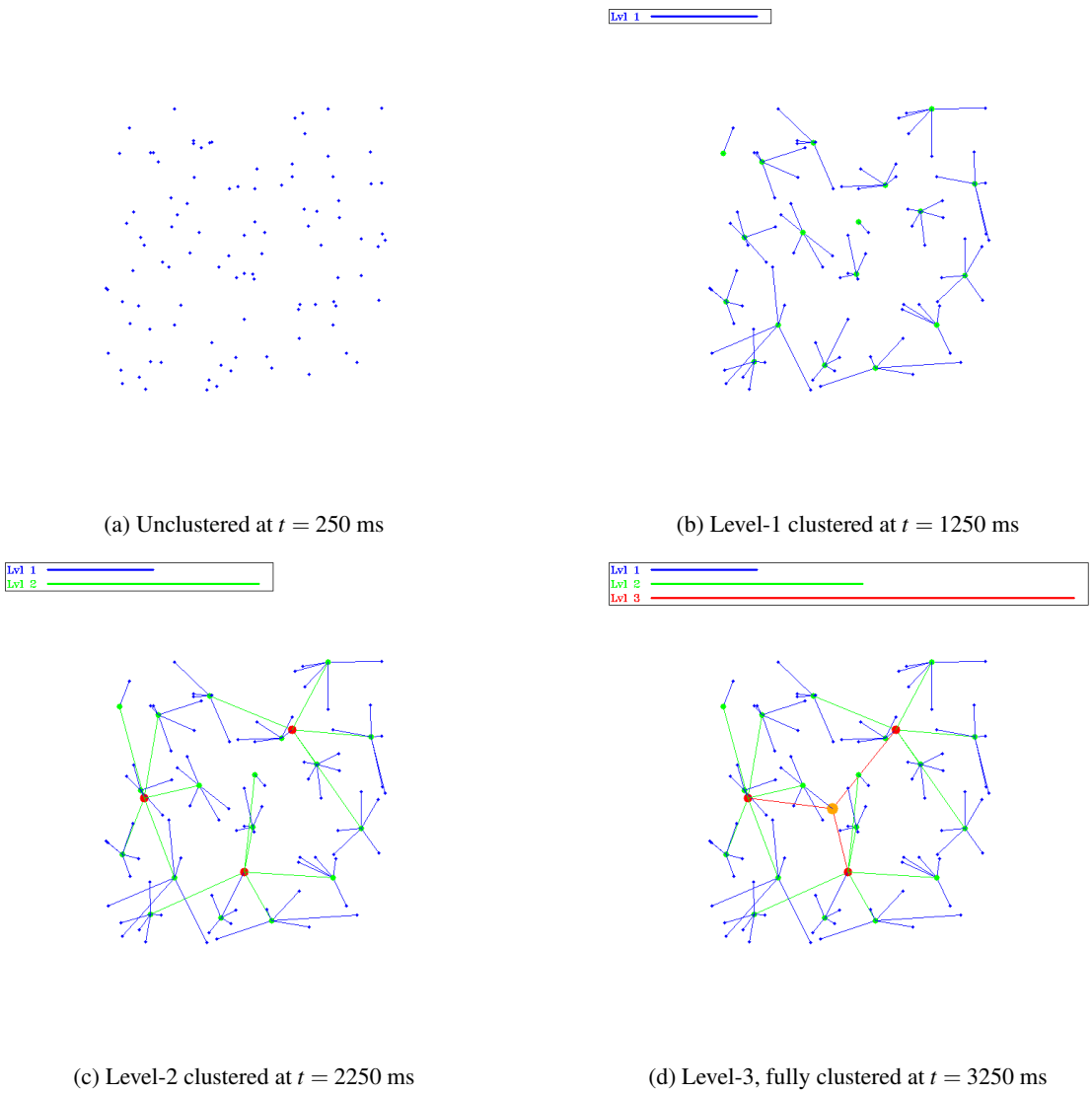
A. Non-spawning stationary

The scenes in Figure 3 show the clustering of the agents at different times during the simulation. The blue points are idle agents, i.e., agents that are not cluster-heads. Conversely, green points are level-1 cluster-heads, red points level-2, and orange points level-3 cluster-heads. Connecting lines indicate cluster membership with blue lines connecting idle blue agents with their green cluster-head, green lines connecting green level-1 cluster-heads with red level-2 cluster-heads, and analogously for higher levels. The legends on top show the different distance thresholds for the cluster levels.

The distance thresholds double for each increment in cluster level in all the scenarios, starting at 15m for level-1. The agents perform their *join-create* tasks at 10Hz and switch from the *join* phase to the *create* phase after 1 s and vice versa after 2 s. This guarantees that *create* phases of unclustered agents or cluster-heads without higher cluster membership overlap. The cluster capacity is set to 6.

The first scenario is showing 100 agents that are spawned at time $t = 0$ s, starting their autonomous clustering algorithms simultaneously. Their positions are randomly sampled uniformly in a 40m $\times$ 40m area. At $t = 250$ ms, the agents are still unclustered since they are trying to join existing level-1 clusters. As all of them spawned simultaneously, no clusters exist. At time $t = 1250$ ms all agents switched to their *create* phase and created the first level of clusters. The cluster-heads are indicated in green. In contrast to higher-level clusters, level-1 cluster-heads are also members of their headed clusters, explaining that at max five blue lines are visible per cluster. At this time, the green agents finalized forming their clusters and try joining potentially existing level-2 clusters.

At time $t = 2250$ ms level-2 clusters formed, promoting idle level-1 members to level-2 cluster-heads. The level-2 cluster-heads are still members in their previous level-1 clusters showing a two-way blue and green line to one of the level-1 cluster heads for each level-2 cluster-head (red). At $t = 3250$ ms, one level-3 cluster is formed, again promoting



(a) Unclustered at $t = 250$ ms

(b) Level-1 clustered at $t = 1250$ ms

(c) Level-2 clustered at $t = 2250$ ms

(d) Level-3, fully clustered at $t = 3250$ ms

Figure 3: Stationary non-spawning clustering.

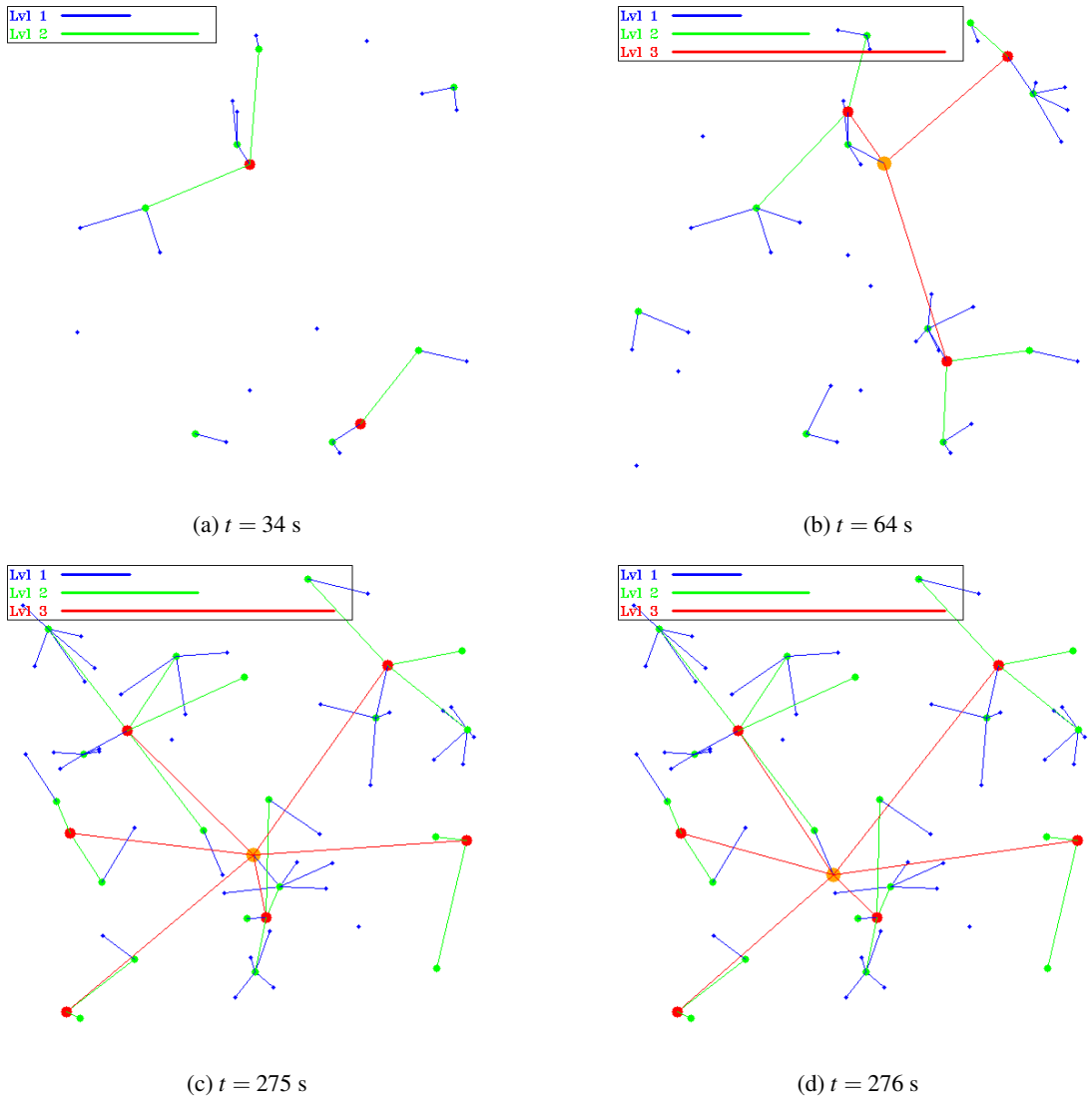
an idle level-1 member to the cluster-head. From now on, the level-3 cluster-head is trying to join or create a level-4 cluster but will not find any other level-3 cluster-heads to create a new level-4 cluster.

This scenario shows how the proposed clustering is intended to work in general. The simultaneous spawning of the agents made it very simple to form a tree that contains every agent. The following examples complicate the scenario to challenge the algorithm's reaction to dynamic environments.

B. Spawning stationary

In this scenario, the reaction to the agents' spawning and de-spawning is shown. For this the spawning area is increased to 100m $\times$ 100m. Agents are spawned, i.e., taking off, periodically with intervals between spawns uniformly sampled from $[1000, 2000]$ ms. Each agent receives a lifetime sampled from $[75, 100]$ s after which the agent de-spawns, i.e., lands. Due to the spawning and de-spawning intervals, it is expected to have an average number of active agents around 58.

The simulation samples in Figure 4 are similar to the previous ones, showing a greater area. Since no agent is present at the start of the simulation at $t = 34$ s, only some agents spawned and formed small clusters up to level-2. At



(a) $t = 34$ s (b) $t = 64$ s

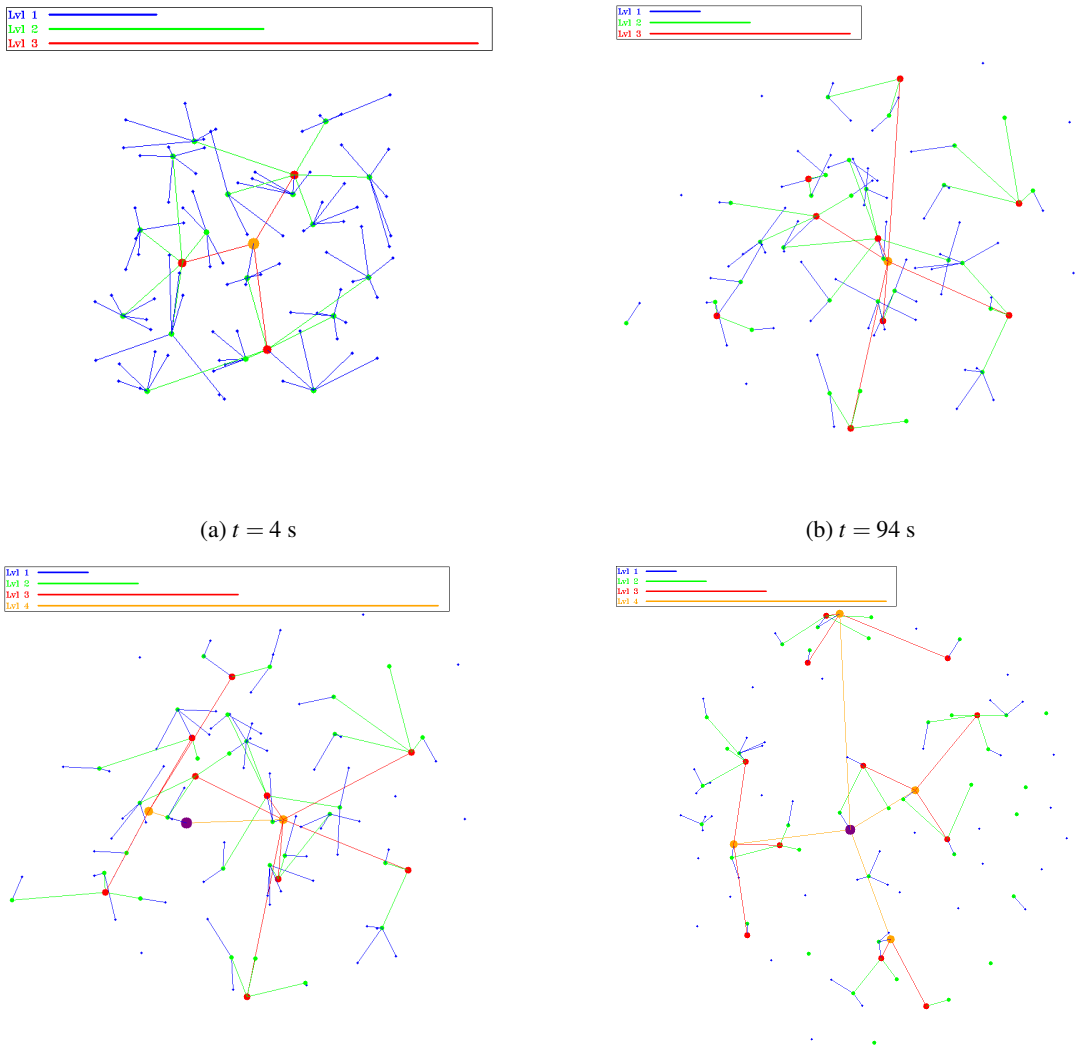
(c) $t = 275$ s (d) $t = 276$ s

Figure 4: Stationary spawning clustering.

$t = 64$ s, all agents that spawned are still present, as the lowest lifetime value is 75 s. More agents spawned, allowing larger clusters, including a level-3 cluster, connecting a majority of the agents.

Later during the simulation, a stable number of active agents is established, with agents spawning and de-spawning at similar intervals. At time $t = 275$ s nearly all agents are clustered together. Note that none of the agents at that time point were present in the previous times, yielding very different clusters. To show how the algorithm reacts to cluster-heads de-spawning, the next time sample at $t = 276$ s shows the clustering after the level-3 cluster-head (orange) from the previous time sample de-spawned. When de-spawning, a cluster-head chooses a new cluster-head from the idle level-1 cluster members. The new selection is based on proximity to the centroid of the cluster members. It can be seen that no other cluster is affected by the cluster-head transition.

This scenario intends to show how well the algorithm handles a varying number of agents, which is crucial when dealing with dynamic real-world scenarios since UAVs have limited battery capacity. We show that for our proposed clustering, it is not necessary to assume that all participating agents are active simultaneously.



(a) $t = 4$ s

(b) $t = 94$ s

(c) $t = 98$ s

(d) $t = 200$ s

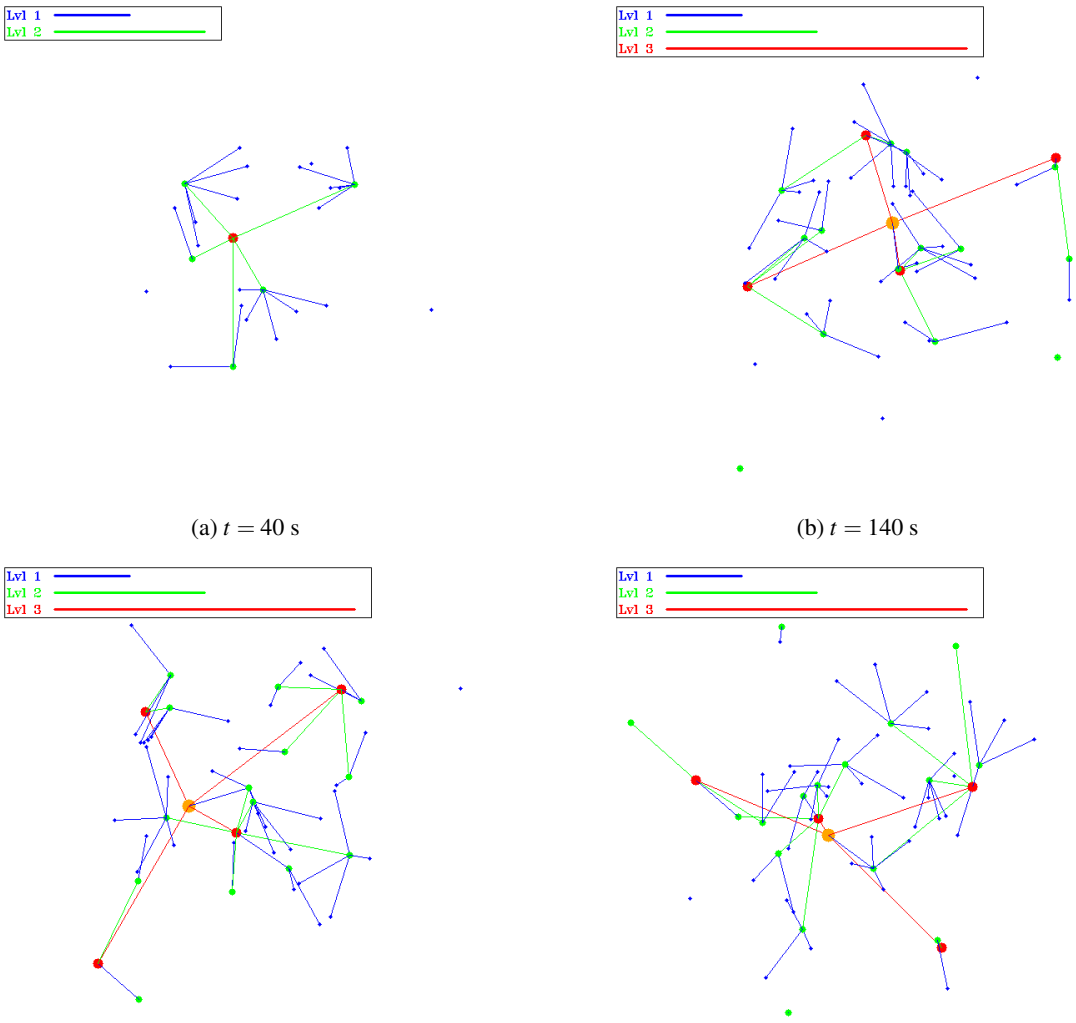
Figure 5: Non-spawning spiral movement clustering.

C. Non-spawning spiraling

This scenario introduces motion to the agents, evaluating the handling of agents leaving or entering cluster radii. The motion chosen is a counter-clockwise spiraling motion around the agent's spawning location. Consequently, some agents move in the same direction and others in different directions depending on their starting positions. The traveling speed is randomly sampled for each agent, and the starting position is sampled as in the non-spawning stationary case in Figure 3. The agents' lifetime is set to infinite, and all agents are spawned at the start of the simulation.

Four time samples from the simulation are shown in Figure 5 similar to the previous scenarios. The difference in this Figure is that the scale changes for different time samples as agents spiral outwards. At $t = 4$ s all agents are clustered as in Figure 3. They slowly start spiraling outwards.

At $t = 94$ s, the agents spiraled out so far that too many level-2 clusters are necessary, yielding nine level-2 cluster heads. These cannot be clustered in one single level-3 cluster, showing three disconnected trees. At $t = 98$ s, this problem is solved by the main tree evicting the top level-2 cluster-head, allowing the right level-2 cluster-head to join. The three disconnected clusters form a new level-3 cluster, whose cluster-head creates a new level-4 cluster together with the cluster-head of the first level-3 cluster. One idle agent in the middle is elected to level-4 cluster-head (purple).



(a) $t = 40$ s

(b) $t = 140$ s

(c) $t = 200$ s

(d) $t = 260$ s

Figure 6: Spawning spiral movement clustering.

After 200 s, the agents are getting more sparse and disconnect slowly, showing many disconnected agents. However, a majority of agents are still connected to the main tree.

The scenario shows that handling motion, especially relative motion, is quite complicated for clustering. We show that our proposed algorithm can handle relative motion. However, a current shortcoming is that sparse agents cannot be handled well. In future work, this will be addressed to connect as many agents as possible.

D. Spawning spiraling

The spawning spiraling scenario combines the spawning behavior of Figure 4 and the motion of Figure 5. In contrast to the non-spawning scenario in 5, the agents are expected to remain in some area because they de-spawn. Therefore, the scale of the graphics can remain unchanging.

At $t = 40$ s, some agents spawned, starting their spiraling movement. One level-2 cluster formed, containing all but three agents. As in the stationary spawning scenario, at $t = 140$ s, all agents are replaced, but the clustering structure remained and grew further. A level-3 cluster was able to form connecting most, but some sparse agents on the

outskirts. At time $t = 200$ s all but one agent are connected to the main tree of which some disconnect at time $t = 260$ s. Throughout all time samples, a majority of the agents are connected to the main tree.

This scenario best approximates a real-world flight of UAVs, including relative motion as well as take-off and landing. After an initial ramp up time, a relatively constant number of active agents can be observed as the de-spawning agents are replaced. Therefore, a scenario like this can demonstrate how well the clustering can behave in a dynamic, mostly dense scenario. The proposed algorithm can handle the scenario very well except for sparse outskirts.

V. Conclusion and Future Work

This paper proposed the idea of multi-level clustering to enable communication and coordination in decentralized systems of autonomous UAVs. From an initial state without any knowledge of their neighbors, agents form tentative clusters by broadcasting "hello" messages then subsequently electing cluster-heads to schedule intra-cluster activity. Cluster-heads use the same strategy to discover neighbors and form higher level clusters. If no neighbors are detected, clusters periodically toggle between join and create phases, appropriately time-shifted to guarantee overlap with any other disconnected clusters moving in range. This iterative process of cluster formation and discovery continues until all agents are connected and synchronized into a single hierarchical tree.

Some basic clustering rules were proposed for maintaining hierarchical integrity in the system and individual cluster independence. These rules were simulated for different scenarios involving stationary/non-stationary agents with spawning/no-spawning capabilities. The simulation outcomes were described and discussed.

Future work will explore cluster maintenance strategies, such as cluster size balancing and eviction policies to improve multi-level clustering. We will introduce performance metrics to assess and quantifiably weigh the merits of each strategy. These metrics will evaluate connectivity among the agents, communication resources needed for cluster formation and maintenance, and indicators for efficient dissemination of aggregate state information.

Acknowledgments

Marco Caccamo was supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.

References

[1]Bellman, R., "A Markovian Decision Process," *Journal of Mathematics and Mechanics*, Vol. 6, No. 5, 1957, pp. 679–684.

[2]Kim, D. K., Liu, M., Omidshafiei, S., Lopez-Cot, S., Riemer, M., Habibi, G., Tesauro, G., Mourad, S., Campbell, M., and How, J. P., "Learning Hierarchical Teaching Policies for Cooperative Agents," 2019.

[3]Julian, K. D. and Kochenderfer, M. J., "Distributed Wildfire Surveillance with Autonomous Aircraft Using Deep Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 8, 2019, pp. 1768–1778.

[4]Haksar, R. N. and Schwager, M., "Distributed Deep Reinforcement Learning for Fighting Forest Fires with a Network of Aerial Robots," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1067–1074.

[5]Bayerlein, H., Theile, M., Caccamo, M., and Gesbert, D., "Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning," 2020.

[6]Pineau, J., Gordon, G., and Thrun, S., "Point-based Value Iteration: An Anytime Algorithm for POMDPs," *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003, pp. 1025–1030.

[7]Spaan, M. T. J. and Vlassis, N., "Perseus: Randomized Point-based Value Iteration for POMDPs," *J. Artif. Int. Res.*, Vol. 24, No. 1, Aug. 2005, pp. 195–220.

[8]Osband, I. and Van Roy, B., "Near-optimal reinforcement learning in factored MDPs," *Advances in Neural Information Processing Systems*, Vol. 1, No. January, 2014, pp. 604–612.

[9]Foka, A. and Trahanias, P., "Real-time hierarchical POMDPs for autonomous robot navigation," *Robotics and Autonomous Systems*, Vol. 55, No. 7, 2007, pp. 561 – 571.

[10]Theile, M., Bayerlein, H., Nai, R., Gesbert, D., and Caccamo, M., "UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning," 2020.

[11]Wu, F., Zilberstein, S., and Chen, X., "Multi-Agent Online Planning with Communication," *Proceedings of the Nineteenth International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'09, AAAI Press, 2009, p. 321–328.

[12]Stone, P. and Veloso, M., "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork," *Artificial Intelligence*, Vol. 110, No. 2, 1999, pp. 241 – 273.

[13]Roth, M., Simmons, R., and Veloso, M., "Reasoning about Joint Beliefs for Execution-Time Communication Decisions," *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, Association for Computing Machinery, New York, NY, USA, 2005, p. 786–793.

[14]Oliehoek, F. A., Whiteson, S., and Spaan, M. T., "Approximate solutions for factored Dec-POMDPs with many agents," *Belgian/Netherlands Artificial Intelligence Conference*, 2013, pp. 340–341.

[15]Nair, R., Tambe, M., Roth, M., and Yokoo, M., "Communications for improving policy computation in distributed POMDPs," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, 2004, pp. 1098–1105.

[16]Boutilier, C., "Planning, Learning and Coordination in Multiagent Decision Processes," *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '96, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996, p. 195–210.

[17]Hu, Y.-C., Johnson, D. B., and Perrig, A., "SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, Vol. 1, No. 1, 2003, pp. 175 – 192.

[18]Hu, Y.-C., Perrig, A., and Johnson, D. B., "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," *Proceedings of the 2nd ACM Workshop on Wireless Security*, WiSe '03, Association for Computing Machinery, New York, NY, USA, 2003, p. 30–40.

[19]Hu, Y.-C., Perrig, A., and Johnson, D. B., "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Wireless Networks*, Vol. 11, No. 1, 2005, pp. 21–38.

[20]Jacquet, P., "Optimized Link State Routing Protocol (OLSR)," RFC 3561, RFC Editor, October 2003.

[21]Perkins, C. E. and Bhagwat, P., "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *SIGCOMM Comput. Commun. Rev.*, Vol. 24, No. 4, Oct. 1994, pp. 234–244.

[22]Rawashdeh, Z. Y. and Mahmud, S. M., "A novel algorithm to form stable clusters in vehicular ad hoc networks on highways," *Eurasip Journal on Wireless Communications and Networking*, Vol. 2012, No. 1, 2012, pp. 15.

[23]Vodopivec, S., Bester, J., and Kos, A., "A Survey on Clustering Algorithms for Vehicular Ad-Hoc Networks," 07 2012, pp. 52–56.

[24]Rossi, G. V., Fan, Z., Chin, W. H., and Leung, K. K., "Stable clustering for Ad-Hoc vehicle networking," *IEEE Wireless Communications and Networking Conference, WCNC*, 2017, pp. 1–6.

[25]Hassanabadi, B., Shea, C., Zhang, L., and Valaee, S., "Clustering in Vehicular Ad Hoc Networks using Affinity Propagation," *Ad Hoc Networks*, Vol. 13, No. PART B, 2014, pp. 535–548.

[26]Wei Fan, Yan Shi, Shanzhi Chen, and Longhao Zou, "A mobility metrics based dynamic clustering algorithm for VANETs," *IET International Conference on Communication Technology and Application (ICCTA 2011)*, 2011, pp. 752–756.