AIAA SciTech Forum
23-27 January 2023, National Harbor, MD & Online
AIAA SCITECH 2023 Forum

10.2514/6.2023-2658

Sequential Cooperative Reinforcement Learning of Mothership Routing Problem

Jaeho Lee[*], Junwoo Park[†], Jayden Dongwoo Lee[‡], and Hyochoong Bang[§]
Korea Advanced Institute of Science and Technology, Daejeon, 34141, Korea

This paper proposes a sequential cooperative reinforcement learning for the vehicle routing problem which possess a moving depot (mothership) where the depot is bound for a designated destination. Drones which are considered as delivery vehicles and deployed from the moving depot should convey packages to the known customer within operation time of the mothership. Challenges here are to assign which drones to which customer at what point in the path of the mothership, as well as to determine the best route of the mothership and those of the drones that minimizes cumulative distance. Our solution approach is to separate constitutive rolls into respective agents instead of controlling everything in a centralized fashion using single policy network. Then the key contribution of this work is to train the collection of policies in a sequential cooperative fashion using alternated interaction among them. The suggested algorithm represents each policy well trained with increasing reward as iteration continues, moreover more stable than simultaneous reinforcement learning for each agent. The proposed design prevents affecting training of other agent. Also, algorithm makes proper gradient backpropagation of reward possible. On the vehicle routing problem with mothership, our approach outperforms a solution with simultaneous learning of two policy agent in solution quality and training time with computational cost.

I. Introduction

The vehicle routing problem (VRP) is classified as a combinatorial optimization and integer programming problem that has been studied in the field of computer science. The problem is NP-hard whose solution approaches are often limited or even not tractable within polynomial time at all. Many exact and heuristic algorithms have been proposed but obtaining better and faster solutions is still a challenging task. The classical VRP problem is posed as a determination of a set of routes such that all customers' requirements and operational constraints are satisfied and some notion of the global transportation cost is minimized. Each vehicle departs from a certain node and arrives at another called the depot. Depending on the applications of interest, the VRP has many variations and specializations according to the constraints and conditions. One of the popular variances is Dynamic VRP (DVRP) [1], and the node of this problem have dynamic elements and changes over timesteps. In a DVRP, unlike other variants of VRP, some information may be available after several timesteps which make the original route design ineffective afterwards. Therefore, isolating the static and the dynamic information in this problem is a core strategy. Re-planning from the original route is inevitable whenever new information is observed from the dynamic elements. For an overview of the DVRP and its' variances, see, for examples, [2–5].

The VRP with mothership, which is moving depot, is one of the rising problems in DVRP. It reflects the near future where drones are aggressively utilized. The biggest shortcoming of the drone as a delivery vehicle is capacity that only one package is carriable in a single departure. To cover areas with many customers with this disadvantage, the aid of multiple drones and a mothership that moves along them is essential. In this paper, we call this problem VRP with mothership and focus on it. The objective is combined optimization of drones and a mothership to reduce the transportation cost which comprise traveling distance or time. The development starts from the paper [6] by Murray and Chu, where combined truck and drone model is considered which drone is utilized for inefficient delivery locations. The work from Ulmer and Thomas [7] improves the model where truck and drones are dispatched. Other papers are more concentrated on drone delivery [8–10]. However, these papers use heuristic optimization methods to solve the problem. Despite the mathematical proof that the approaches converge to the (sub)optimized solution, one of the biggest

[*]Master Student, Department of Aerospace Engineering, KAIST, 16jhlee@kaist.ac.kr, Student Member of AIAA.
[†]Ph.D. Candidate, Department of Aerospace Engineering, KAIST, junwoopark@kaist.ac.kr, Student Member of AIAA.
[‡]Ph.D. Candidate, Department of Aerospace Engineering, KAIST, cin6474@kaist.ac.kr, Student Member of AIAA.
[§]Professor, Department of Aerospace Engineering, KAIST, hcbang@kaist.ac.kr, Member of AIAA.

disadvantages of such heuristic approaches is computational cost they consume. Generally, the computation time is (exponentially) proportional to the problem dimension and complexity. Therefore, they are not suited for dynamic problems which require an immediate reaction to the varying conditions. Moreover, they also show bad scalability.

As another perspective, without hand-engineered reasoning, reinforcement learning with neural networks is a compelling option. Deep reinforcement learning gives a huge milestone in approaching various problems. After a seminal work [11] by Mnih, which plays Atari games with deep reinforcement learning, huge advances in deep reinforcement learning arose such as [12–14]. Naturally, there are some attempts to apply reinforcement learning to VRP with mothership. After Nazari [15] applied reinforcement learning for traveling salesman problem (TSP) and VRP, Chen and Ulmer [16], utilize deep Q learning to solve VRP with drones and trucks setup. They called these kind of systems as fleet and optimized each truck and drone with static depot. They concentrated on choosing the location of delivery for each drone. This paper shows deep reinforcement learning can be applied to dynamic vehicle routing. One step further in reinforcement learning, considering each delivery vehicle as individual decision maker, multi-agent reinforcement learning can be a solution to the DVRP. The root of multi-agent reinforcement learning is by Ming's work [17], where agents are trained by isolated Q-learning algorithms. Each agent uses partial observation to obtain action. This algorithm works well in a small dimension, not in the large one. Also observation sharing does not considered. Jacob and Gregory developed an Implicit Q-Learning algorithm called COMA [18]by proposing centralized learning with reward shaping. Agents trained by COMA show cooperative working. Many multi-agent algorithms are based on COMA. Rashid proposed another centralized learning from IQL algorithm [19], where mixing network works for centralized learning. However, training VRP with mothership by multi-agent reinforcement learning algorithm is time consuming and hard to shape reward for all agents. Since, multi agent algorithm assumes that every agent has same roles or similar tasks. However, the agents in VRP with mothership has different tasks and policy. Ideal reward shaping with algorithm does not guarantee proper gradient backpropagation. Despite the analogy between MARL and the problem of interest in this study, direct application is avoided due to the poor performance, which agent does not know their role. Also, there are more better ways to train heterogeneous agents.

Back to our problem, we develop a framework for solving VRP with mothership using deep reinforcement learning through sequential cooperative reinforcement learning which is a major contribution of this study. For this purpose, we consider the finite Markov decision process (MDP) formulation of multiple interacting agents. The problem of previous studies of solving VRP with reinforcement learning was using only one agent. This kind of modeling often yields extremely huge action space training of which is not effective. Therefore, using single agent requires other creative skills during training or in reward shaping. These methods can draw a good policy network beside disadvantage in not only computational cost but also not scalable. The reward is very sensitive, so variation of MDP formulation or problem states can affect the different results. We propose sequential cooperative reinforcement learning that three policy networks are contained in the environment: drone agent, allocator agent, and mothership agent. Each policy handles respective object. For instance, a drone agent focuses only on delivering packages from the mothership and comes back, in other words, rendezvous point. The allocator agent decides which nodes to deliver and when the drone should start to deliver in current timestep. Lastly, mothership agent determines which route the mothership should take in order to reduce transportation costs of itself and drones. Dividing the problem into each module can reduce the action space and be scalable for changes in the environment. There are two reasons we divide into exactly three agents. First, it is practical to real-world operations. Each module corresponded to each vehicle or device. Second, if we organize system with two agents, one is combined agent of mothership and allocator another is drone, it was almost impossible to train. This is because of action space and gradient backpropagation. The action space of first agent is large and each action represents different functions. The sequence of these actions makes training hard since it is very hard to notice which action makes more rewards. Substituting the object of one module for the other can be easily trained with simple reward shaping. In addition to separate modules, sequential learning follows for a more optimized policy. Using trained modules in an environment when training another module and vice versa gives more stable search for agent. After a few iterations, the entire modules are properly trained but not fully optimized. At the final step, do a simultaneous learning to get global optimum route. Therefore, we called this scheduling method as sequential-cooperative training. As a result, the policy that is learned shows better rewards in sequential training steps than usual simultaneous training at the beginning. This process is simple and computational cost is reduced than ordinary cooperative reinforcement learning. This sequential-cooperative learning is better than simultaneous learning in local convergence and early stopping criteria. Training simultaneously is not a good method since each agent is not trained at the beginning. For example, the allocator policy shows an action but the drone policy is not trained yet, so it does not complete its objective and the wrong reward can be returned even though other agents work well. This makes searching space large and needs more timestep or even failed to train. Therefore, simultaneous learning is hard to apply to the environment that each policy is very important combined system. Also, individual early stopping is another advantage of sequential cooperative reinforcement learning. Early stopping in reinforcement learning is very important to prevent the overfitting of policy. A single agent can be fitted to a single object or another untrained agent, not a global optimization. It is really hard to decide early stopping even though converges, also it is not an optimized policy. We investigated overfitting of the drone agent. In the simultaneous learning, drone agent works well in the early timesteps. As timestep goes on, drone agent trained uncertain searching actions of mothership which is vibrating and starts to vibrate together. This ruins whole episode and returns negative rewards. In the sequential training steps, the reward of each training step is due to only one agent distinct from cooperative training which is influenced by whole agents. Therefore, it is easy to notice that agent meets proper level and it provides when to stop training of one agent. Thus, sequential cooperative reinforcement learning can stop the iteration of a specific module in a separate manner. We design a trick that early stopping criteria of drone agent to complete the training. Finally, there are two reasons we divide whole system into three agents. First, it is practical to real-world operations. Each module corresponds to each vehicle or device. Also, dividing into two agents, one is combined agent of mothership and allocator another is drone, is almost impossible to train. This is because, backpropagation does not works properly since two objectives are combined. More details are explained on later section.

Our numerical experiments indicate that our framework performs well and reward of each module is increased as iteration continues. Moreover, utilizing three agents gives better results and shorten the training time against using only two agents. Another interesting observation of our results is policy of allocator agent. The allocator decides whether delivering or not for certain node according to mothership and destination.

The main contributions of this paper are 1) create a VRP with mothership environment conditions to increase efficiency of routing problems. 2) proposes a scheduling methods for reinforcement learning which is necessary to optimize our routing problem, 3) evidence that scheduling methods release disadvantages of ordinary way to solve. 4) Open-sourced environment and training code* for further optimization.

II. Backgrounds

A. Taxicab geometry

A Taxicab geometry is a form of geometry whose metric is replaced by Manhattan distance. The Manhattan distance represents the distance between two points as a sum of the absolute differences of their Cartesian coordinates. The Manhattan distance, d_1, between two vectors $\mathbf{p}, \mathbf{q}$ in an n - dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes as

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{1}$$

where $\mathbf{p}, \mathbf{q}$ are vectors

$$\mathbf{p} = (p_1, p_2, \cdots, p_n), \mathbf{q} = (q_1, q_2, \cdots, q_n). \tag{2}$$

The Manhattan distance depends on the rotation of the coordinate system, but does not depend on its reflection about a coordinate axis or its translation. Taxicab geometry satisfies all of Hilbert's axioms except for the side-angle-side axiom. In this paper, environment is based on taxicab geometry since fleet operation on urban circumstances which have buildings and blocks have some restricted areas. The taxicab geometry was chosen in order to restrict the situations to discrete action spaces. The main process of this paper must include comparison of scheduling algorithms between using 2 agents and 3 agents. If mothership and drone agent move on a continuous action space, they would require separate training algorithms fundamentally different from that of the allocator agent which is on a discrete action space. To compare the scheduling algorithms, such differences are unacceptable. This forces us to use a discrete action space for mothership and drone agents.

B. Proximal Policy Optimization

This study adopts the proximal policy optimization (PPO) reinforcement learning algorithm [20] as a building block of the proposed sequential cooperative learning mechanism. As the algorithm works in an on-policy fashion, the primary intention is to let each constitutive agent, i.e. drones, the mothership, or the allocator, interact as its up-to-date version. Furthermore, its applicability to the discrete action space is suitable for the routing problem especially when

*Code is available at https://github.com/yaeho/Cooperative-Mothership the taxicab geometry (1) is concerned. It's also worth mentioning that the effectiveness of the algorithm has been verified over a number of fields such as motion planning [21], logistics [22] and vehicle routing problems [23]. The ease of implementation due to its simple structure should be the last appealing feature. As a brief summary of PPO algorithm, this subsection describes only the core points that make it powerful while leaving the thorough derivation and the detailed explanation of it to the original paper [20, 24] and the referenced therein.

Let's think of a surrogate objective function $L(s_t, a_t, \theta_{\text{old}}, \theta)$ [24] denoted as

$$L(s_t, a_t, \theta_{\text{old}}, \theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}(s_t, a_t), \tag{3}$$

instead of a gradient of some general notion of (discounted) return, which should be the expectation of gradient of log probability over taken paths. Here, the calculation of the advantage $\hat{A}$ is based on the prior parameters θ_{old}, i.e., $\hat{A} = A^{\pi_{\theta_{\text{old}}}}$, and the ratio denotes a relative improvements compared to the one before the update. Then a policy that maximizes (3) should intuitively yields a better one in terms of stochastic performance while preventing overestimation, and thus the catastrophic collapse of a performance. Thus, solving the problem

$$\theta^* = \arg\max_\theta \mathop{\mathbb{E}}_{s,a\sim\theta_{\text{old}}} \left[L(s_t, a_t, \theta_{\text{old}}, \theta)\right] \tag{4}$$

was the core idea of the trust region policy optimization (TRPO) algorithm [24] while the optimization domain is limited for the safe and monotonic improvements using the Kullback-Leibler divergence as

$$\mathop{\mathbb{E}}_{s\sim\pi_{\theta_{\text{old}}}} \left[D_{\text{KL}}\left(\pi_\theta(\cdot|s) \| \pi_{\theta_{\text{old}}}(\cdot|s)\right)\right] < \delta, \tag{5}$$

with δ being a design parameter. Despite the theoretical soundness of the method, however, realization of it is certainly not an easy task. The TRPO approximates the objective along with its constraint using second order Taylor expansion, and solves the problem by the method of Lagrangian duality [25], yet the process is still way too much complex involving the calculation of Hessian-based Fisher information matrix (FIM).

PPO improves TRPO [24] in a practical way by introducing a clipping parameter ϵ as

$$L(s_t, a_t, \theta_{\text{old}}, \theta) = \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}(s_t, a_t), \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \hat{A}(s_t, a_t)\right), \tag{6}$$

where the operator $\text{clip}(\cdot, l, u)$ for given parameters l, u such that $l < u$ denotes

$$\text{clip}(x, l, u) = \begin{cases} l, & \text{if } x < l \\ u, & \text{if } x > u \\ x, & \text{otherwise,} \end{cases} \tag{7}$$

yielding an extremely efficient calculation of (4). The calculation of the advantage is often realized using multiple time steps, e.g., minibatch, aided by the principle of general advantage estimation (GAE) [26]. It is well-supported by numerical studies that even the simple choice of ϵ, e.g., 0.2, provides stable learning, so that the automatic regularization is already supported by the minimum number of design parameters. With the help of first-order approximation supported by the simplest numerical trick, PPO improves both the sample and computation efficiency, and mitigates the performance collapse problem. Note that advanced variants of (6) often includes entropy bonus and/or squared error of state value function with that of the target.

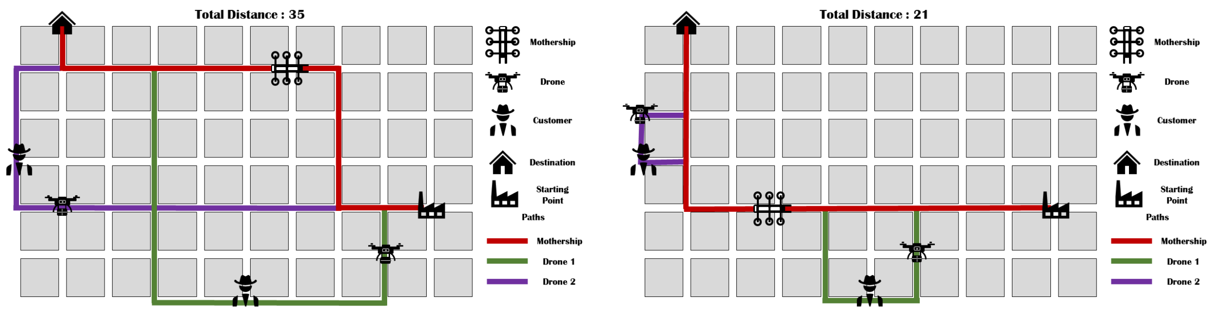
III. Problem Definition

This section explains about the problem we handled. The main object is utilizing heterogeneous fleet in VRP to minimize total distance of the whole system. Therefore, we designed the conditions of heterogeneous fleet and markov decision process model to optimize the problem.

A. Heterogeneous Fleet

We consider heterogeneous fleet which consists of a mothership and drones. The mothership plays as a moving depot which gives packages and recharging to the drones. The whole system has mothership, the moving depot, allocator, and drones similar to the same-day delivery except static depot. The mothership has unlimited capacity due to work as moving depot and responsibility for certain small area. There is allocating drone to customer agent which is called allocator. The mothership and allocator receive same observation data to decide action.

In contrast to mothership, drones have limited battery capacity and payload weight. The one thing important is that drones are subject to weight limits. Nevertheless, in this paper, we suppose that packages of customers request are within the weight limit since there are statistics that between 75 and 90 percent of Amazon deliveries could technically be handled by the drone [27]. Therefore, in this study we assume that drone delivers a package to the location and came back to the mothership without certain payload limitation. Also, usual drone has high speed, so drone flies faster than mothership.



(a) Not optimized Route

(b) Optimized Route

Fig. 1 Figure (a) shows non optimal route. Cumulative travel range of drone is long. Figure (b) shows sub-optimal route. The cumulative distance of drones and mothership is much shorter than previous non optimal route.

B. Markov Decision Process Model

Markov decision process (MDP) model is designed for VRP with mothership. An MDP is a discrete-time stochastic control process. It provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. [28].

Decision Point A decision point is when the decision made. In this model, decision occurs every timestep. A mothership agent, allocator agent, drone agents decide in a row.

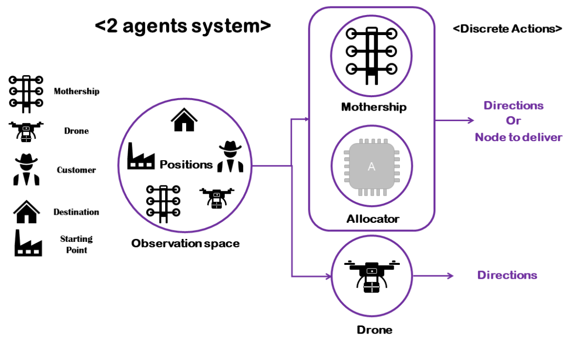
State The state at a decision point provides useful information needed to make the decision. In this model, the state S^t at the *t-th* timestep includes location of mothership, destination of mothership and customers (packages). Therefore, we notate the state S^t as a tuple $S^t = (P_c, P_{c_d}, P_p)$, the mothership location at *t-th* timestep P_c, the mothership destination P_{c_d} and the location of customers P_p. In order to separate delivered or not, we add a mask to packages at state.

Actions As sequential cooperative reinforcement learning, there are three agents in our model. Also there is the order of action, which mothership agent actions first based on observation state. After than, allocator agent decides (actions) which customers (packages) should be delivered. Lastly, drone agents actions according to their observations. We represent the action at a each decision points (timesteps) $A^t = (D^t_c, D^t_a, D^t_{d_i})$ of mothership, allocator and drone agent i at *t-th* timestep as D^t_c, D^t_a and $D^t_{d_i}$ accordingly. The action space of mothership and drones are 5 dimensional discrete space each number represents the cardinal points and last one means stay. Allocator agent have one more dimensions than the number of customers. This means which customer should drone have to deliver at that timestep. If allocator outputs 0 means there is nothing to begin delivery at this moment. The comparison environment which comprise 2 agent the mothership and drone, has different action space. In this environment the mothership agent must decide which customer to deliver, the role of allocator agent. Therefore, mothership action space has 5 more discrete space than the number of customers. The drone agent has same discrete action space with drone agent in 3 agent environment.

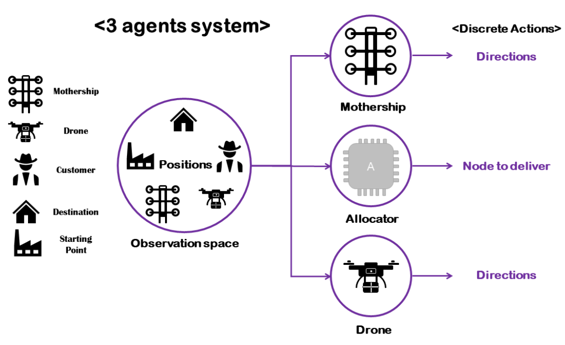
Before MDP update states from action of each agents, model checks following conditions. If state does not meets simple three conditions than process will be stopped.

1) Each vehicles of fleet does not get out from the desired environmental space. For instance, the space limitation
2) Process iterates until mothership arrives at the destination of mothership
3) Process iterates only about the fixed timesteps

Reward The reward of the each agent have different shape. The basement logic of reward shaping is receiving same expected reward on every random episodes. For instance, because of random initialization of environment, each episode has different customer locations and mothership location. If reward function is consist of absolute strict value, the maximum reward of each state is mismatched. This causes bias to the training algorithm and makes algorithm fails. Therefore, we designed the drone to get a same reward in each delivery. Dividing constant reward according to the distance makes same reward for every delivery. The distance is decided when the drone starts delivery. The carrier agent in 2 agent system has combined reward function of carrier and allocator. The carrier agent receive a reward when it arrives at the destination. This reward is made of the number of packages are delivered and whole moved distance of agents in thje environment. The allocator agent has additional reward function which gets constant reward from ordering delivery to drone until come back. The constant reward is settled from delivering distance of a drone when drone came back. This can transfer the signal efficiently to the policy network about the allocator action. The penalty is given when the agent gets out of the environment space.



(a) 2 agents system



(b) 3 agents system

Fig. 2 Figure (a) shows 2 agents system markov decision process. Figure (b) shows 3 agents system markov decision process. Both have same observation spaces.

IV. Training Method

The overall process of VRP with mothership can be described as three parts. One is for drone agents which deliver a package to the customers. Other is allocator agents that allocate drones to deliver packages to the customers. If they do not allocate any packages to drones, it means do not go delivery at this moment. Another is the mothership the moving depot. Mothership can adjust routes friendly to drones and an allocator. The cooperation of these three modules can reduce the transportation costs or time. The proposed idea came from this modulization. This is a kind of multi-agent problem, but there is limitations utilizing multi-agent reinforcement algorithms. Multi-agent algorithms are usually used for training same type of policy which roles similar tasks. However, our environment has distinct agent which is allocator agent. The characteristic of allocator agent is totally different with drone or mothership agent. Therefore, it is hard to send a correct reward signal to allocator policy network by multi-agent reinforcement learning algorithms. Also, these algorithms require a large timestep since every agent does not know what is their function in the environment. It can be trained if spending tremendous episodes which is inefficient and rare probability to be trained.

There is another possible method to train our environment. Using single agent reinforcement learning algorithm to each agent. This method has high uncertainty since this is deeply combined environment. The training speed is different according to the type of agent. Some agents can be trained quickly others are slowly. Difference of the training speed induces failure of the training. The quickly trained agent works properly in the initial timesteps but they trains the random searching movements ,which is untrained, of other agents and ruins whole system. This harms system gradually, finally system fails. To prevent this occurrence, early stopping method is mandatory. However, the reward is deducted from the actions of multiple agents. This means it is hard to specify the proper stopping point of the each

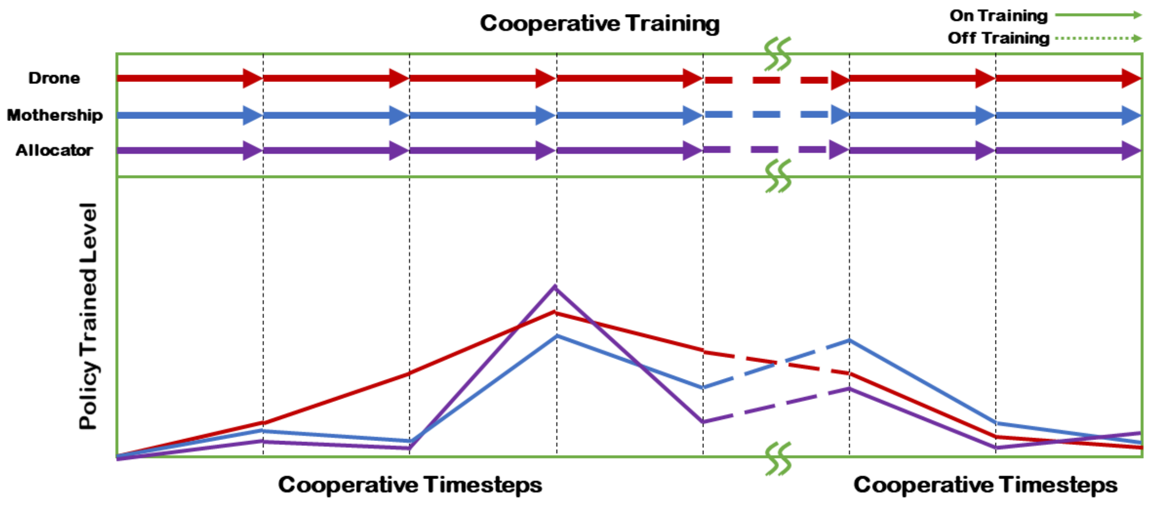


Fig. 3 Cooperative training algorithm. Each agent is trained with single agent reinforcement learning algorithm. In the early timesteps, total reward increases and each agent trained properly. However, as timestep goes on, some early trained agents train random searching action and system broken.

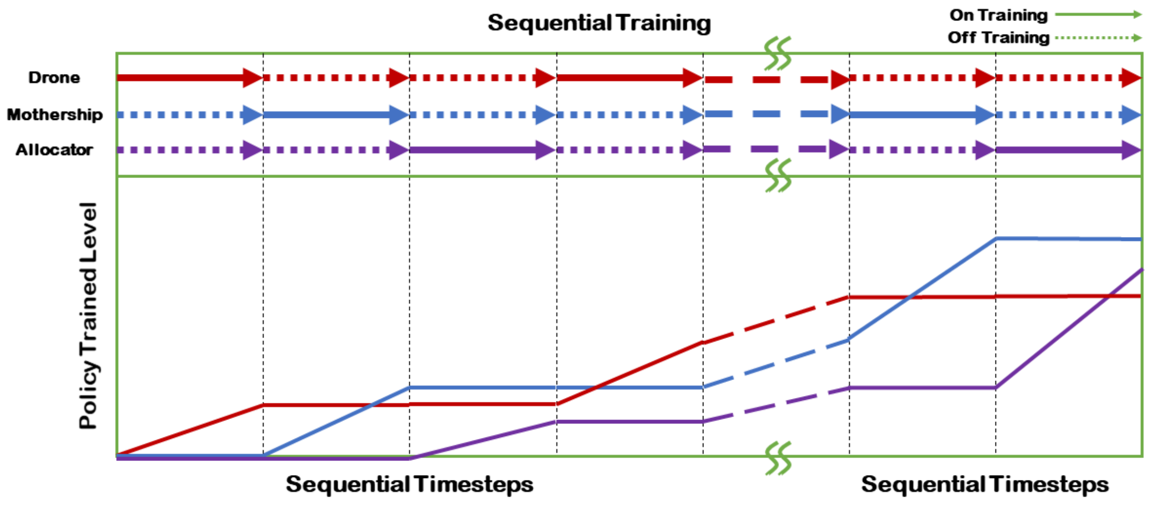


Fig. 4 Sequential training algorithm. Each agent is trained with single agent reinforcement learning algorithm. Only one type of agent is trained in a pahse. It does not destroy the whole system, but it has disadvantages that impossible to get a global optimum point.

agent. Therefore, sequential cooperative learning algorithm is proposed.

Contrary to the multi-agent reinforcement algorithm, we propose sequential reinforcement learning. Each agent is trained with the single agent reinforcement algorithm in a row. Every agent of our environment has important role in one united system, so without any single module system could not work properly. Using trained policies when training one module and same for training another modules. Through iteration process, each module is trained several times not continuously. Process have sequence of training. For instance, drone agent the most simple agent is trained first. Then, using drone agent, training allocator agent is followed. Next, mothership agent will be trained using drone and allocator agent. Go back to the training process of drone agent and these process continues, all agents are learned gradually. This is sequential training.

The critical disadvantage of utilizing only sequential training method is impossible to reach a global optimum point. Only one agent is trained in a timestep, so agent get a optimum point at that condition. Therefore, we add a cooperative training after the sequential training to get a global optimum. These combined method can prevent disadvantages of previous two algorithms. Sequential training before the cooperative training prevents each agent to learn a random searching untrained movements. This is because, only one agent is trained in a specific phase. In this phase, the reward changing is only influenced by a agent. This makes possible to create an early stopping criteria of some agents. The later training, the cooperative training, makes whole agents to reach global optimum point through the combined training.

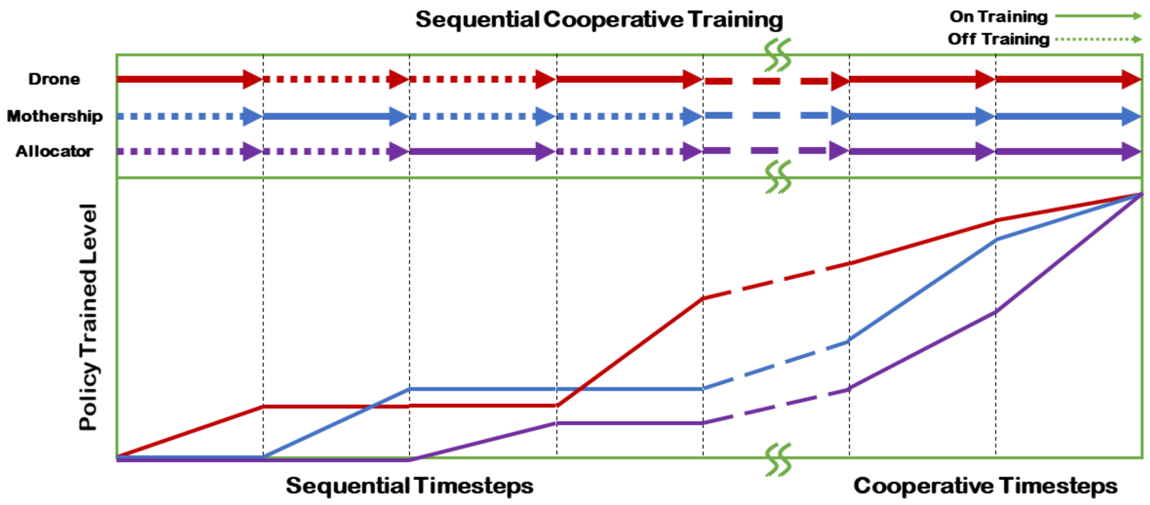


Fig. 5 Cooperative training algorithm. Each agent is trained with sequential learning algorithm in the early timesteps. Only one type of agent is trained in a phase. In the later timestpes, cooperative training makes agents to reach a global optimum point

V. Experiments

We designed experiments to verify our algorithms. The experiment held in 1 mothership, 4 drones, 10 packages environment. The other conditions can be solved by the same algorithm. We set 6 different algorithms to compare the performances. The experiments are classified by the number of agents and 3 algorithms which are cooperative, sequential and sequential cooperative. The reward function of each experiments are same. For instance, combined mothership agent gets a reward that sum of separated mothership and allocator reward function. The observation space is array of mothership, package and drone locations. The package locations are marked when drone got a order to delivery that package. The network architecture is made of fully connected layers and tanh activate follows. The experimental hyperparameters are represented below.

Hyperparameter	Value	Hyperparameter	Value
Total timesteps	10^7	γ	0.95
Cooperative timesteps	7×10^6	Learning rate critic	10^-3
Sequential timesteps	3×10^6	Learning rate actor	3×10^-4
Sequential training frequency	3×10^5	Update epochs	80
Max episode length	10^3	Action σ	0.6

The Figure 6 shows the running average of each experiment. There are three characteristics to focus on. First, in cooperative experiments they reached high result level, but reward decreased after timesteps. This is because of drone agents which failed to delivery missions. In the early timesteps, drone moved properly to its destination. However, in the later timesteps, drone trained untrained random movement of mothership agent and it began to vibrate. This causes decreasing in reward value and finally whole system went wrong. On the other hand, sequential training remained reward level. This is because of 2 additional tricks. Sequential training prevents gradient backpropagation of training about untrained other policies. Each agent trains learned policy at least. This minimizes random movement and unaccountable movements. Also, early stopping criteria for drone can be applied in sequential training. In the drone training steps, the reward change is only influenced by alternation of drone policy. Therefore, reward in drone training steps leads us to early stopping criteria. This is shown in the Figure 7. The drone loss in sequential training remained constant after early stopping of drone policy. Also according to Figure 8 the mothership loss of cooperative learning increased after beginning of cooperative learning. This also shows training going broken gradually.

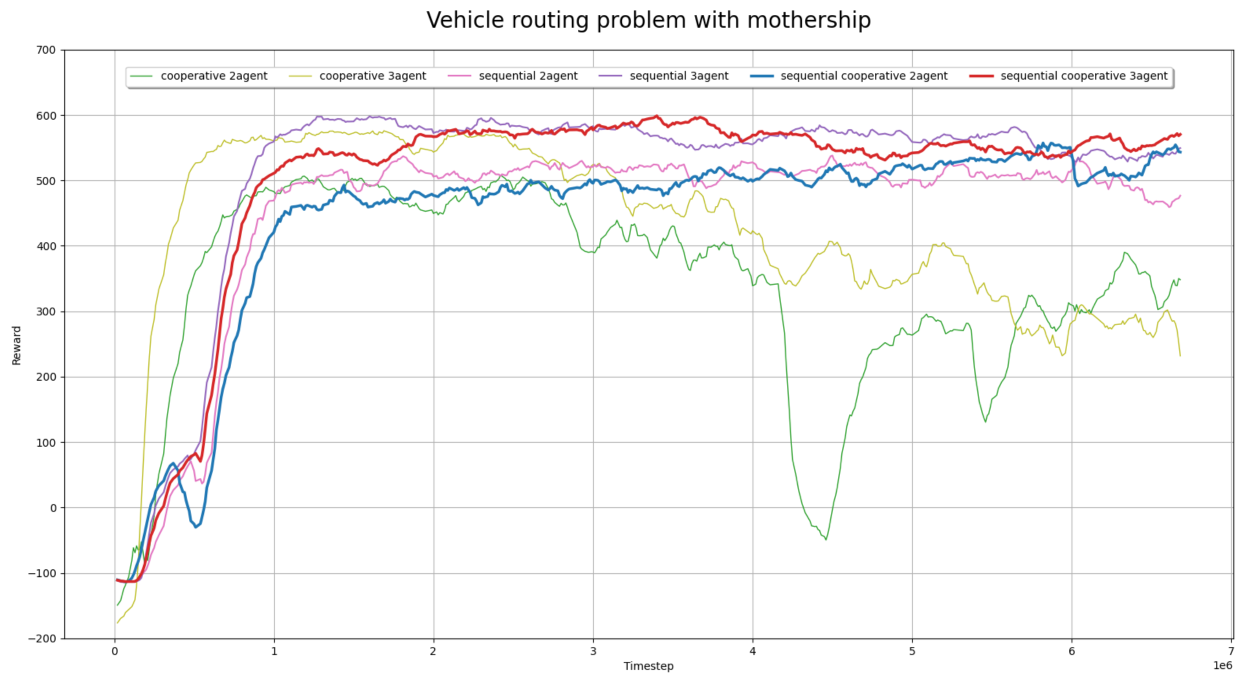


Fig. 6 The average reward of each experiment. The sequential learning remains the policy trained level but cooperative learning is broken as timestep goes.

Next, the 3 agents show better performances than using 2 agents. Even though using same reward functions, the gradient backpropagation did not work properly in the combined agent. The agent stores sequence of actions and rewards. There are blended reward about cost due to mothership movement and allocator movement. This makes agent difficult to distinguish accurate actions. Therefore, 3 agents gave higher rewards than 2 agents. Figure 8 shows that the loss of mothership converged in 3 agent experiments but increased in 2 agent experiments. Finally, using cooperative learning gave better performances. After beginning of cooperative timesteps the reward of the episode was increased. Sometimes it reduced a bit, but it had increasing tendency. Reaching global optimum point requires hyperparameter tuning only for cooperative learning.

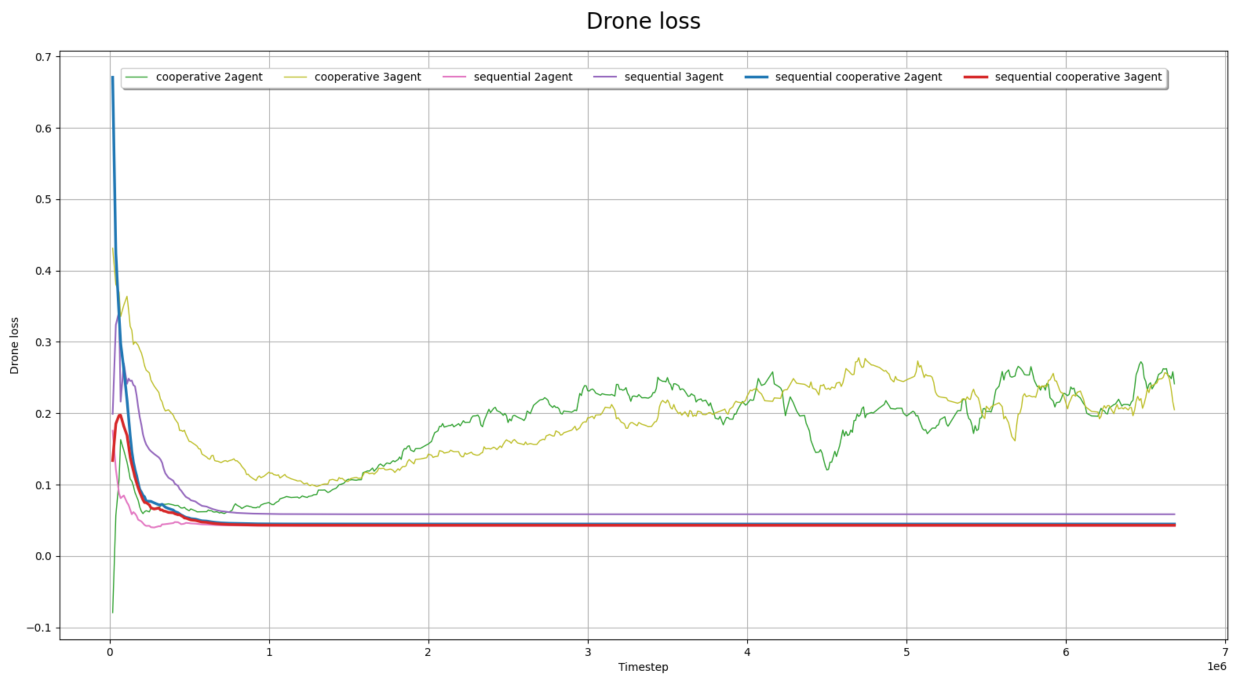


Fig. 7 The drone loss of experiments. The drone loss increases in cooperative experiments. However, in the sequential experiments there exists early stopping criteria and sequential algorithm to prevent divergence.

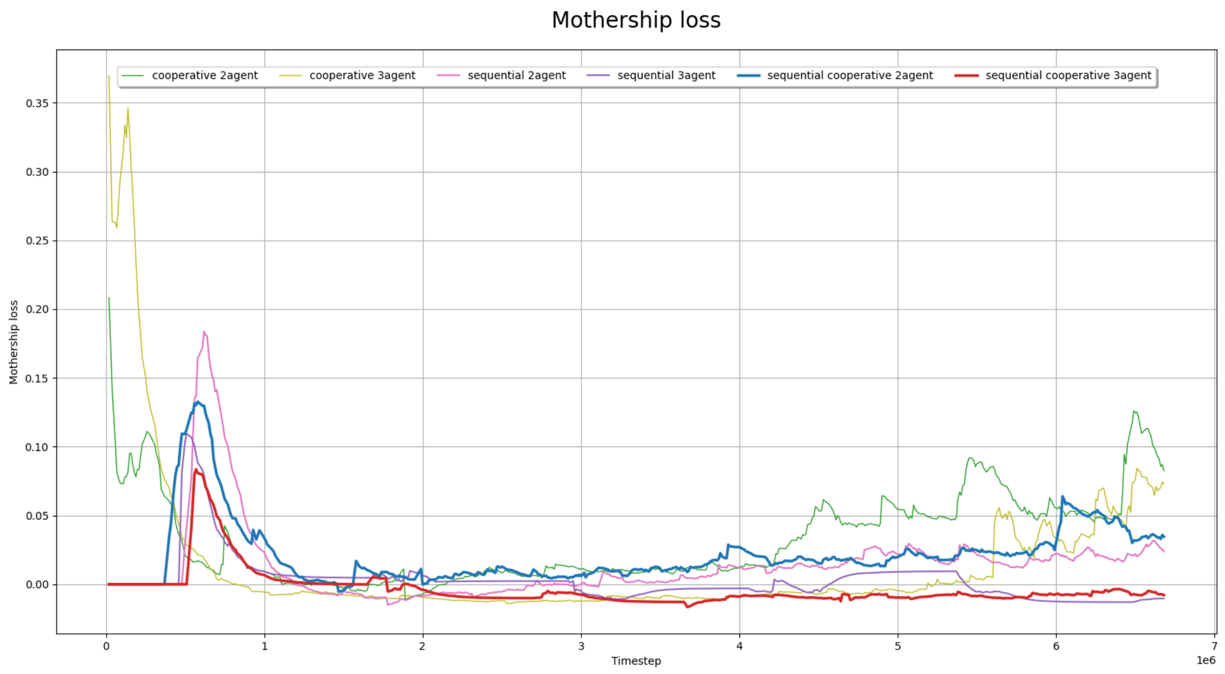


Fig. 8 The mothership loss of the experiments. The mothership loss of 3 agent and sequential cooperative training converges. In the 2 agent experiments, combined agent of mothership and allocator causes diverging of mothership loss.

VI. Conclusions and Future works

The sequential cooperative algorithm showed better performances than cooperative algorithm and sequential algorithm. Proposed algorithm supplements cooperative reinforcement learning in heterogeneous multi-agent environment which has critical disadvantage to diverge. The sequential algorithm has advantages that requires low computational cost and insensitive to other agents or hyperparameters. The one shortcoming, impossible to reach global optimum point is solved by adding cooperative learning after sequential training. Our algorithm can be utilized also in other heterogeneous multi-agent problems not only in vehicle routing problem with mothership since simple and small computation cost.

We found that network architecture is important for solving our problem to get a global optimum. Our algorithm solves the environment not in perfect way. Therefore, design a policy and critic network architecture to acquire a global optimum and fast converging is our future work. Also, logic of hyperparameter tuning in cooperative learning after sequential learning is required.

References

[1] Adewumi, A. O., and Adeleke, O. J., "A survey of recent advances in vehicle routing problems," *International Journal of System Assurance Engineering and Management*, Vol. 9, No. 1, 2018, pp. 155–172.

[2] Pillac, V., Guéret, C., and Medaglia, A., "Dynamic vehicle routing problems: state of the art and prospects," 2011.

[3] Larsen, A., Madsen, O. B., and Solomon, M. M., "Recent developments in dynamic vehicle routing systems," *The vehicle routing problem: Latest advances and new challenges*, Springer, 2008, pp. 199–218.

[4] Hvattum, L. M., Løkketangen, A., and Laporte, G., "A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems," *Networks: An International Journal*, Vol. 49, No. 4, 2007, pp. 330–340.

[5] Euchi, J., Yassine, A., and Chabchoub, H., "The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach," *Swarm and Evolutionary Computation*, Vol. 21, 2015, pp. 41–53.

[6] Murray, C. C., and Chu, A. G., "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, Vol. 54, 2015, pp. 86–109.

[7] Ulmer, M. W., and Thomas, B. W., "Same-day delivery with heterogeneous fleets of drones and vehicles," *Networks*, Vol. 72, No. 4, 2018, pp. 475–505.

[8] Campbell, J. F., Sweeney, D., and Zhang, J., "Strategic design for delivery with trucks and drones," *Supply Chain Analytics Report SCMA (04 2017)*, 2017.

[9] Carlsson, J. G., and Song, S., "Coordinated logistics with a truck and a drone," *Management Science*, Vol. 64, No. 9, 2018, pp. 4052–4069.

[10] Dayarian, I., Savelsbergh, M., and Clarke, J.-P., "Same-day delivery with drone resupply," *Transportation Science*, Vol. 54, No. 1, 2020, pp. 229–249.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[12] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[13] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous methods for deep reinforcement learning," *International conference on machine learning*, PMLR, 2016, pp. 1928–1937.

[14] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] Nazari, M., Oroojlooy, A., Snyder, L., and Takác, M., "Reinforcement learning for solving the vehicle routing problem," *Advances in neural information processing systems*, Vol. 31, 2018.

[16] Chen, X., Ulmer, M. W., and Thomas, B. W., "Deep Q-learning for same-day delivery with a heterogeneous fleet of vehicles and drones," *arXiv preprint arXiv:1910.11901*, 2019.

[17] Tan, M., "Multi-agent reinforcement learning: Independent vs. cooperative agents," *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[18] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S., "Counterfactual multi-agent policy gradients," *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32, 2018.

[19] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S., "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," *International Conference on Machine Learning*, PMLR, 2018, pp. 4295–4304.

[20] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] Kim, S., Park, J., Yun, J.-K., and Seo, J., "Motion planning by reinforcement learning for an unmanned aerial vehicle in virtual open space with static obstacles," *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, IEEE, 2020, pp. 784–787.

[22] Iriondo, A., Lazkano, E., Susperregi, L., Urain, J., Fernandez, A., and Molina, J., "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Applied Sciences*, Vol. 9, No. 2, 2019, p. 348.

[23] Holler, J., Vuorio, R., Qin, Z., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C., and Ye, J., "Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem," *2019 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2019, pp. 1090–1095.

[24] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P., "Trust region policy optimization," *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

[25] Boyd, S., Boyd, S. P., and Vandenberghe, L., *Convex optimization*, Cambridge university press, 2004.

[26] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[27] D'Onfro, J., "Amazon's new delivery drone will start shipping packages in a matter of months," https://www.forbes.com/sites/jilliandonfro/2019/06/05/amazon-new-delivery-drone-remars-warehouse-robots-alexa-prediction/?sh=32f382fb145f, 2019. [accessed: 06.05.2019].

[28] Wikipedia, "Amazon's new delivery drone will start shipping packages in a matter of months," https://en.wikipedia.org/wiki/Markov_decision_process, 2006. [accessed: 04.22.2006].