Naval Research Laboratory
Washington, DC 20375-5320

NRL/5540/MR—2023/4

Autonomous Cyber Security

Myong Kang
Alexander Velazquez

Center for High Assurance Computer Systems
Information Technology Division

September 12, 2023

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 12-09-2023	**2. REPORT TYPE** NRL Memorandum Report	**3. DATES COVERED** *(From - To)* 10/01/2020 – 09/30/2023

4. TITLE AND SUBTITLE

Autonomous Cyber Security

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Myong Kang and Alexander Velazquez

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
6C14

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/5540/MR--2023/4

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We describe a set of decision-making algorithms (centralized, decentralized, and hybrid) for a multi agent system and present the Factored-Value (FV) Monte Carlo Tree Search (MCTS) Hybrid Cost Max-Plus algorithm for Autonomous Cyber Security. Our proposed algorithm consists of a two-step process. In the first step, each agent uses MCTS to find its best individual actions, taking\ into account cost. Each agent presents its most promising actions to the team. In the second step, the Hybrid Cost Max-Plus algorithm is used for joint action selection. This Hybrid Cost Max-Plus algorithm improves upon the known centralized and distributed Max-Plus algorithms without cost by including the cost of actions in the interactions of agents. The Max-Plus algorithm uses the framework of Coordination Graphs, which exploit dependencies among agents to decompose the global payoff function as the sum of local terms. Our proposed FV-MCTS-Hybrid-Cost-Max-Plus algorithm is online, anytime, distributed, and scalable in terms of the number of agents and their interactions. Our contribution competes with state of art methods and algorithms that use MCTS and Max-Plus to exploit the locality of agent interactions for planning and acting.

15. SUBJECT TERMS

MCTS	Multi agent system	Real time decision making
Distributed coordination	Planning algorithm	

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Myong Kang
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	U	26	**19b. TELEPHONE NUMBER** *(include area code)* (202) 767-0555

This page intentionally left blank.

CONTENTS

This page intentionally left blank.

Autonomous Cyber Security

Final Report

Myong Kang, Alexander Velazquez
Center for High Assurance Computer Systems
U.S. Naval Research Laboratory
Washington, DC, USA
{myong.kang, alexander.velazquez}@nrl.navy.mil

Abstract—**We describe a set of decision-making algorithms (centralized, decentralized, and hybrid) for a multi agent system and present the Factored-Value (FV) Monte Carlo Tree Search (MCTS) Hybrid Cost Max-Plus algorithm for Autonomous Cyber Security. Our proposed algorithm consists of a two-step process. In the first step, each agent uses MCTS to find its best individual actions, taking into account cost. Each agent presents its most promising actions to the team. In the second step, the Hybrid Cost Max-Plus algorithm is used for joint action selection. This Hybrid Cost Max-Plus algorithm improves upon the known centralized and distributed Max-Plus algorithms without cost by including the cost of actions in the interactions of agents. The Max-Plus algorithm uses the framework of Coordination Graphs, which exploit dependencies among agents to decompose the global payoff function as the sum of local terms. Our proposed FV-MCTS-Hybrid-Cost-Max-Plus algorithm is online, anytime, distributed, and scalable in terms of the number of agents and their interactions. Our contribution competes with state of art methods and algorithms that use MCTS and Max-Plus to exploit the locality of agent interactions for planning and acting.**

Keywords— MCTS, Multi Agent System, Real Time Decision Making, Distributed Coordination, Planning Algorithm.

I. INTRODUCTION

Today's military relies on complex computer systems to support diverse, dynamic, and agile missions in contested and rapidly changing environments. The computer systems themselves should be agile and resilient enough to guarantee the success of the missions they support. Adversaries are able to launch cyberattacks that are dynamic, fast-paced, and high-volume, while cyber responses are human-speed. In current cyber defense systems, most system adaptation and recovery processes are ad-hoc, manual, and slow. Keeping pace with existing and emerging cybersecurity threats is a challenging task, especially without the benefit of relying on the human expertise of system administrators and cyber warriors. The objective of this project is to develop enabling technologies for resilient computer systems that can autonomously protect themselves and recover from cyberattacks and system failures. Specifically, we will develop a cyber security machine learning framework that can be used to develop systems that can intelligently and autonomously protect themselves and recover from cyberattacks and system failures while ensuring mission continuity.

IT systems typically consist of multiple subsystems that work together. Seeking independent solutions for each subsystem would yield suboptimal behavior. Autonomous Cyber Security (ACS) seeks to develop algorithms to solve the real-time multi agent decision-making problem in a collaborative setting. We specifically seek to develop *anytime* algorithms for real-time decision making. Anytime algorithms return a result at any time and the longer the algorithm runs, the better the result will be until the optimal result is reached. Not only the payoff but also the cost of an action is an important factor (benefit to mission vs. disruption to mission by taking a particular action) and this must be taken into account in the algorithms.

Manuscript approved September 12, 2023.

We consider a cooperative multi agent decision-making problem where all agents share a common goal [1-8]. Decision making for large action spaces in Multi Agent Systems (MAS) can be computationally expensive and sometimes intractable. Fortunately, in many situations, an agent only needs to coordinate its actions with a few agents and acts independently with respect to the remaining agents. Factorizing the action space into that of a team of agents that coordinate their decisions can make the problem tractable. Coordination is crucial for effective decision making in Cooperative Multi Agent Systems (CMAS) with a shared objective. Each agent is making a local decision and, at the same time, contributing to the globally optimal solution for the team.

A key aspect in implementing a policy for CMAS is to coordinate the decision making of the agents. This coordination problem could be solved using a centralized coordinator that decides what action each agent should take and then communicate the selected actions to the agents. This solution is not robust due to the possibility of malfunction of the centralized coordinator or potential communication problems among agents. Therefore, we will consider a hybrid solution that combines *centralized* and *decentralized* (*distributed)* coordination[1]. In the absence of a central unit (i.e., coordinator), the agents must coordinate their actions toward achieving the shared goal by using a distributed algorithm. The algorithm we describe in this paper considers the cost of actions as a constraint for decision making.

An additional requirement for the algorithm is that it must be an *anytime* algorithm. If the algorithm stops due to the budget constraints (time, cost of actions, etc.), we would like the joint action with the highest payoff to be reported.

A Multi agent Markov Decision Process (MMDP) generalizes the single Markov Decision Process (MDP) model, using the framework of MAS, with the notion of multiple cooperative agents that have their own action sets (as defined in Definition 1, p.962 [13]). Decision making in a MMDP can be either centralized or decentralized. In centralized algorithms, a single decision maker prescribes the actions for all agents to take, while in decentralized algorithms, there is a decision maker for each agent. Our contribution belongs to the *Constraint* Multi agent Markov Decision Process (CMMDP) domain focusing on *online planning* ([13] and references therein). Obtaining a solution for CMMDP is more challenging than MMDP because there are resource constraints. These resource constraints affect the decision making of the CMAS.

The solution of a MMDP is, in general, based on the framework of the Coordination Graphs (CG) algorithm and the Variable Elimination (VE) algorithm that can be used in this regard [1-7,12,13]. This VE algorithm operates by eliminating agents one by one by performing a local maximization step and has exponential complexity with the induced tree width (the size of the largest clique generated during node elimination). Unfortunately, the VE algorithm is not an anytime algorithm. A solution of the *anytime* MMDP is the Max-Plus algorithm, provided in a centralized and a decentralized version in [3-5]. The Max-Plus algorithm is a payoff propagation algorithm where agents exchange appropriate payoff messages over the CG, and they can compute their individual actions based on the convergence of this approach. Little is known about the hybrid version of the Max-Plus algorithm with constraints, which is the focus of this paper.

The purpose of our work is to provide a hybrid solution for a constrained anytime scalable decision-making algorithm (centralized and distributed coordination) in MAS. Online methods for

[1] *Decentralized* and *Distributed* are often used interchangeably despite describing two distinct phenomena. *Decentralized* refers to the action execution while *distributed* refers to location of the algorithm implementation for decision-making agents.

decision making present an alternative strategy, where the agents must interleave the planning and execution of their actions.

Monte Carlo Tree Search (MCTS) is an anytime method, and it is a common approach for online planning. Dealing with an enormous decision space in the context of MCTS is an open problem that is the subject of ongoing research. The limitations of the MCTS algorithm are scalability and the large branching factor (failing to scale suitably when the branching factor grows past a certain threshold). To date, there are two noteworthy approaches, [1] and [2], for scalable coordination of agents by combining three elements: a) online planning using MCTS, b) factored representation with CG, and c) centralized Max-Plus method for joint action selection. In [1], the cost of actions is ignored, and the Max-Plus algorithm is not distributed. In [2], the cost of actions and anytime aspect are missing. The authors apply MCTS to solve Multi agent POMDP model, which is centralized. The solution is a state-of-the-art algorithm to solve Multi agent POMDP. However, it becomes intractable as the coupling degree of the agents increases. The proposed methods are called Factored-Value MCTS with Max-Plus (FV-MCTS-Max-Plus) and FV-POMCP in [1] and [2] respectively.

According to our knowledge, this is the first paper dealing with hybrid factored-value representation using MCTS for planning. This is a new approach and here is the rationale for this two-step decision-making process:

1) Depending on the situation (state), some actions may be more relevant than other actions. Local-level decision making provides an ordering of the actions, from high probability to low probability, in terms of their effectiveness based on the current situation. Thus, it is a very useful step to increase the probability of the anytime algorithm finding the optimal solution early.

2) Network segmentation is always possible in a dynamic environment because global-level optimization in multi agent environment requires communication within the network. Communication may not be possible in some situations (e.g., under cyberattacks or network malfunction). In such situations, the algorithm may not achieve the global optimal solution; thus, the local optimal solution obtained in the first step may be the best solution.

We are in favor of fully hybrid coordination for multi agent decision making for the following reasons:

a) In centralized structures, the central coordinator takes joint observations from all agents and makes joint decisions for all agents. Each agent takes an action based on the decision of the central coordinator. Failure or malfunction of the central coordinator is equivalent to the malfunction of the whole MAS.

b) The central coordinator needs to communicate with each agent to exchange information, which dramatically increases the communication overhead at the single coordinator. This may degrade the scalability as well as the robustness of MAS.

c) In a centralized setting (with a centralized coordinator), the agents are not allowed to exchange information with each other related to the state transition or reward. A hybrid coordination approach could allow local and correlated decisions for agents that can communicate.

Our contribution is three-fold:

1) We consider a budget constraint approach where a cost is associated with each action. Different actions consume different amounts of resources, which can be potentially correlated to the global payoff for the team [19]. In such a setting, the aim of the local decision maker is to optimize his decision at any time under a cost and budget constraint. Therefore, the global team reward at each time step is obtained after subtracting the total cost incurred in examining the cost of the local actions. In this way, we extend previous works [1-3] on centralized coordination where only the time was the budget constraint.

2) We develop hybrid (i.e., centralized and distributed) coordination for the Max-Plus algorithm [3,4], where an agent computes and sends updated messages after it receives new and different messages from one of its neighbors[2]. The messages are sent in parallel, which offers some computational advantages over the sequential execution of the previous centralized coordination algorithms [1,2].

3) We develop a new FV-MCTS-Hybrid Cost-Max-Plus method with a two-step decision-making process. Our contribution is the development of the theoretical framework for combining Monte Carlo Tree Search (MCTS) with a Hybrid Cost Max-Plus algorithm for decision making and execution. The proposed method is a suboptimal solution for Dec-POMDPs. The exact solution of a Dec-POMDP is known to be intractable and Non-deterministic EXPonential (NEXP) complete, even for only two agents [14].

The outline of the paper is as follows. In Section II, we present related work. The mathematical background is introduced in Section III. Both variants of the Max-Plus algorithm without cost are considered and revised in Section IV. In Section V, we present the new distributed Max-Plus algorithms with cost. In Section VI, we consider the Max-Plus Hybrid algorithm with cost. The proposed method FV-MCTS-Hybrid Cost-Max-Plus is presented in Section VII. In Section VIII we present our conclusions and future work.

II. Related Work

The problem of behavior coordination in multi agent decision making is a hot research topic being addressed in different communities such as Game Theory, Reinforcement Learning, Decision Theory, Cybersecurity, Constraint Programming, Control and Robotics to name a few [16-24]. Of particular interest is the optimal solution for distributed agents sharing a Global Reward [1,6,8,12,15]. A taxonomy of distributed and local optimization algorithms for multi agent global behavior is given in Fig. 1.

In this paper, we focus on centralized and distributed decision-making coordination where the agents interact with each other while making decisions. In such CMAS, the agents are constrained in their decisions by the cost of actions. Little is known about how the cost of actions influence these decisions and what distributed coordination algorithms can be used to solve real-time scenarios for distributed planning when the agents share a common goal.

[2] In the context of the Max-Plus algorithm the neighbor of agent i is the agent j connected on the edge (i, j)

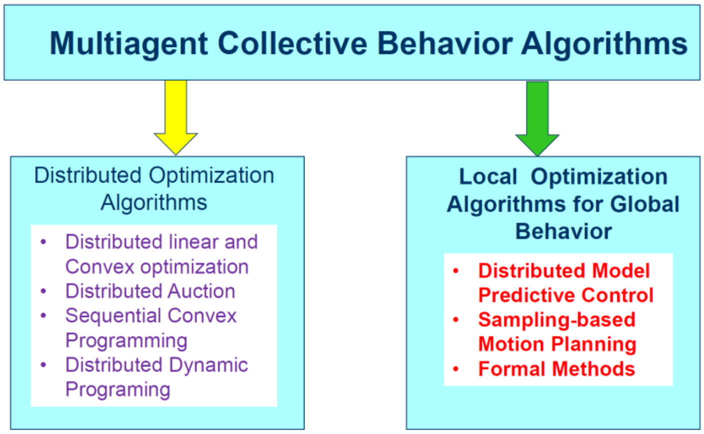

Multiagent Collective Behavior Algorithms
Distributed Optimization Algorithms
• Distributed linear and Convex optimization
• Distributed Auction
• Sequential Convex Programming
• Distributed Dynamic Programing
Local Optimization Algorithms for Global Behavior
• Distributed Model Predictive Control
• Sampling-based Motion Planning
• Formal Methods

Fig. 1. Distributed and local optimization algorithms.

The idea of using the MCTS algorithm to exploit the MAS structure was first presented in [2] in the context of a multi agent POMDP model, which is centralized. The elegant algorithm proposed in [2] uses an exact method of solving the Coordination Graph (CG), namely Variable Elimination (VE) [3-5,12,13]. Unfortunately, the Factored-Value Partially Observable Monte Carlo Planning (FV-POMCP) method proposed in [2] is not *anytime*. This aspect was reviewed in [1], where MCTS was used in conjunction with a *centralized* implementation of the Max-Plus algorithm for joint action selection. The same approach was used in [10].

Our solution for the distributed coordination of multi agent planning is different from [1], [2], and [10], all of which use a centralized approach. In addition, those presented solutions do not consider the cost of actions. In our solution, we use the MCTS algorithm with cost [19] only in the first stage to select a small, ordered set of promising actions for each agent that will be presented to the team in the next stage. In the next stage, our solution uses the Hybrid Cost Max-Plus algorithm, which we also introduce in this paper.

The MCTS approach in [1] and [2] is used for two reasons:

1) To perform a global simulation from the current global state to move agents to the next global state, where the structure of the CG does not change in time.

2) To modify the joint action selection for every agent only during the last step of the algorithm (after reaching the convergence of the centralized Max-Plus algorithm).

In addition, our hybrid solution performs better than selecting actions according to a uniform distribution, as proposed in [10]. Other notable works [3-5,12,13] propose to use the CG with no MCTS extension. The Dec-MCTS approach used in [8] cannot overcome the exponentially large action space induced by naïve application of MCTS. The algorithm proposed in [8] is convergent in some conditions.

III. Mathematical Background

We consider a cooperative multi agent *planning*[3] problem with a system of N agents, a finite horizon T, and a discrete time approach. In each time step $t, 1 \leq t \leq T$, every agent $i, 1 \leq i \leq N$, takes an individual action a_i from its set of actions $\mathcal{A}_i$. The selected joint action for the team is

[3] We use *planning* and *decision making* interchangeably.

the vector of team actions $\boldsymbol{a} = (a_1, a_2,\ldots, a_i,\ldots, a_N)$ and the associated payoff function of the team is given by $Q(\boldsymbol{a})$, not including the cost of the joint action. The classical coordination problem [6] is to find the global optimal joint action $\boldsymbol{a}^* = (a_1^*, a_2^*, \ldots, a_i^*, \ldots, a_N^*)$ that maximizes the global payoff function $Q(\boldsymbol{a})$, i.e.:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Q(\boldsymbol{a}) \tag{1}$$

A naïve approach would enumerate all possible joint actions of all agents and select the one that maximizes the global payoff function $Q(\boldsymbol{a})$. However, this is obviously impractical because the global team action space $\mathcal{A} = \times\, \mathcal{A}_i$ is a N fold cartesian product which grows exponentially as the number of agents N increases. Since the payoff function $Q(\boldsymbol{a})$ used in equation (1) does not consider the cost of actions, it is difficult to apply this model to some real-world problems, since the cost of actions is not negligeable in some application domains [19].

We further consider the discrete variable $\mathcal{S}_i$, the local state of agent i. The global state of the system is denoted by $\mathcal{S}$, which is factored across the agents $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_N$. At time t, the MAS is in global state $\boldsymbol{s} \in \mathcal{S}$ and in the next time step $t+1$, the system will transition to new global state $\boldsymbol{s'}$. The transition probability to the next state $\boldsymbol{s'}$ is $p(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a})$. We consider the simplest settings for factorizing the action space (FAS), where the state of the system is completely observed and available for deciding an action for each component in the action space factorization. The above FAS can be modeled using a MMDP [26].

The cost function c_i of taking an individual action a_i from its set of actions $\mathcal{A}_i$ maps an action a_i to a real number. The cost of the global joint optimal action $\boldsymbol{a}$ of the team $\mathcal{C}(\boldsymbol{a})$ is also a real-valued function. When two agents i and j interact with each other, the cost incurred due to their interaction is denoted by $\mathcal{C}_{i,j}$, which maps a pair of actions (a_i, a_j) to a real number. For the distributed version, we assume that only interconnected agents on an edge coordinate their local actions at a particular time t.

Following the approach from [1], the overall cost, $\mathcal{C}(\boldsymbol{a})$, in CMAS is linear in N, the number of agents, and in $\mathcal{N}$, the number of pair interactions (i, j):

$$\mathcal{C}(\boldsymbol{a}) = \sum_i^N c_i + \sum_{(i,j)}^{\mathcal{N}} \mathcal{C}_{i,j} \tag{2}$$

We define global payoff (a.k.a. Global Reward) of the team as $R(\boldsymbol{a}) = Q(\boldsymbol{a}) - \mathcal{C}(\boldsymbol{a})$, where $Q(\boldsymbol{a})$ is the *benefit to mission* accomplished by the team of agents, and $\mathcal{C}(\boldsymbol{a})$ is the cost of the global joint action.

The new coordination problem is to find the cost optimal joint action $\boldsymbol{a}^*$ that maximizes the Global Reward of the team with cost, i.e.:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} R(\boldsymbol{a}) \tag{3}$$

The reward (payoff) function with cost for an action of agent i is defined as $R_i(a_i) = Q_i(a_i) - c_i$ [3] and, for two agents that are connected on the edge (i, j), the reward function is defined as $R_{i,j}(a_i, a_j) = Q_{ij}(a_i, a_j) - \mathcal{C}_{i,j}$, where the joint payoff values are given by $Q_{ij}(a_i, a_j)$ and the associated joint cost values are $\mathcal{C}_{i,j}$.

In general, in the Max-Plus algorithm, each agent calculates the edge reward by dispatching messages to its neighbors [6]. Including the cost c_i for action a_i (assuming the cost of the action is not dependent on local state) and the cost for joint action $\mathcal{C}_{i,j}$, then a general message sent from agent i to agent j (which is taking action a_j) is a scalar valued function of the action space of the receiving agent j as given below:

$$\mu_{ij}(a_j) = Q_i(a_i) - c_i + \sum_{k\in\Gamma(i)\backslash j} \mu_{ki}(a_i) + Q_{ij}(a_i, a_j) - \mathcal{C}_{i,j} \qquad (4)$$

The only variable in the right part of (4) is the action a_i. For convergence, the local agent i will select his local action a_i (from his set of local actions $\mathcal{A}_i$) to send the maximum payoff value Q_i given by (4) to its neighbor j, producing the message:

$$\mu_{ij}(a_j) = \max_{a_i}\{Q_i(a_i) - c_i + \sum_{k\in\Gamma(i)\backslash j} \mu_{ki}(a_i) \ + \ Q_{ij}(a_i, a_j) - \mathcal{C}_{i,j}\} \qquad (5)$$

The message $\mu_{ij}(a_j)$ can be regarded as a local payoff function of the agent i. Due to the cost, the scalar value given by (5) could become negative. In this case, the contribution to the edge payoff due to the other agents will be defined as $[\mu_{ij}(a_j)]^+ = \max[0, \mu_{ij}(a_j)]$. Finally, each agent individually computes its optimal action:

$$a_i^* = \arg\max_{a_i}\{0, [Q_i(a_i) - c(a_i) + \sum_{k\in\Gamma(i)} \mu_{ki}(a_i)]\} \qquad (6)$$

If the edge payoff is 0, then the agent does not need to take any new action and keeps the old action.

IV. Both Variants of no Cost Max-Plus Algoritms

The Max-Plus algorithm is a popular method for computing the maximum posteriori configuration in an (unnormalized) undirected graph model. This algorithm is analogous to the belief propagation (BP) or sum product algorithm in Bayesian Networks [5,7]. In this algorithm, two agents i and j iteratively send information about their locally optimized payoff values. Previous versions of this algorithm [1,2] did not include the cost of actions and the benefit to the mission for a specific agent.

A. Centralized Max-Plus Algorithm without Cost

The centralized version of the Max-Plus algorithm operates using iterations in a sequential manner. The central coordinator picks an agent i and starts the process. In every iteration, each agent i computes and sends a message μ_{ij} to the neighbor j (connected on an edge) in a predefined order. This process continues until all messages converge to a fixed point, or until a "deadline" signal is received (either from an external source or from an internal timing signal). Then, the most recent joint action is reported. For anytime extension, we update the joint action when it improves upon the best value found so far.

A coordination graph (CG) is a graph $G = (V, E)$ where each node V represents an agent and each edge E defines the dependency between two agents: $N = |V|$ and $\mathcal{N} = |E|$. The Max-Plus algorithm is scalable in terms of the number of agents N and their number of local connections $\mathcal{N}$ in the CG representation [3]. In general, for any CG, we can *factor* the global payoff function *value* as:

$$Q(\boldsymbol{a}) = \sum_{i\epsilon V} Q_i\,(a_i) + \sum_{(i,j)\epsilon E} Q_{ij}(a_i, a_j) \qquad (7)$$

where $Q_i(a_i)$ denotes a local payoff for agent i and it is only based on its individual action when contributing to the system individually.

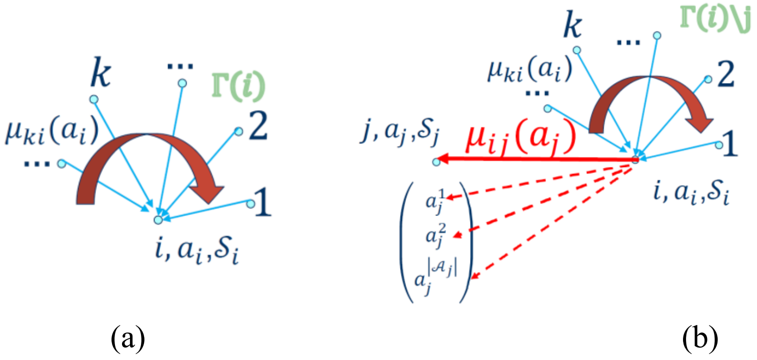
Considering an edge (i, j) for agents i and j, a local joint payoff function Q_{ij} takes a pair of actions $\left(a_i, a_j\right)$ as input and provides a real number as output. In the Max-Plus algorithm without cost, in each time step t, each agent i that is in its local state $\mathcal{S}_i$ takes action a_i by collecting the payoffs from its neighbors as in Fig. 2a. This agent sends a local message μ_{ij} that maps an action a_j of an agent j to a real number as in Fig. 2b.

In Fig. 2b, the transmitted message $\mu_{ij}\colon \mathcal{A}_j \rightarrow \mathbb{R}$ from agent i to agent j in response to the action a_j taken by agent j contains information about the payoff of agent i that is local payoff $Q_i(a_i)$, joint payoff $Q_{ij}\left(a_i, a_j\right)$[4] and the cumulated payoff from all his neighbors $\Gamma(i)\backslash j$ of node i except node j as given below:

$$\mu_{ij}\left(a_j\right) = Q_i(a_i) \ + \sum_{k\in\Gamma(i)\backslash j} \mu_{ki}(a_i) + Q_{ij}\left(a_i, a_j\right) \qquad (8)$$

The first term in relation (8) is the local payoff of the agent i when it takes the local action a_i that is independent of the action a_j of its neighbor j. The second term of (8) shows agent i is collecting all the payoffs from its neighbors $\mu_{ki}(a_i)$ (except neighbor j) when all of its neighbors are observing the action a_i. Agent i is sends its collected payoff only to its neighbor j. The joint payoff $Q_{ij}\left(a_i, a_j\right)$ is shared for both agents. Each agent contributes half of the payoff:

$$\frac{1}{2} Q_{ij}(a_i, a_j) = \frac{1}{2} Q_{ji}(a_j, a_i)$$



(a) (b)

Fig. 2. Interaction on edge between two agents in centralized Max-Plus algorithm.
Fig. 2a: Agent i receives all its neighbors' payoffs. Fig. 2b: Agent i sends the message $\mu_{i,j}$ to agent j.

Agent i can send payoff messages for any action of agent j as in Fig. 2b (dotted arrows). The agents keep exchanging messages at each iteration until they converge. It is proven that for tree structured graphs (no loops), the message updates converge to a *fixed point* after a finite number of iterations [6].

As an example, a coordination graph with 6 agents is shown in Fig. 3. From the perspective of agent 2, agents 1, 3, 4, 5, and 6 are leaves (children), while agents 1, 2, 3, and 4 are roots (parents). The messages transmitted "up" to parents are different from messages sent "down" to their children. Calculating relation (7) is different for different types of agents. If an agent is a leaf, it can respond immediately. However, if an agent is a parent, it waits for its children to send their payoffs.

[4] We assume commutative joint payoff functions $Q_{i,j}\left(a_i, a_j\right) = Q_{j,i}(a_j, a_i)$

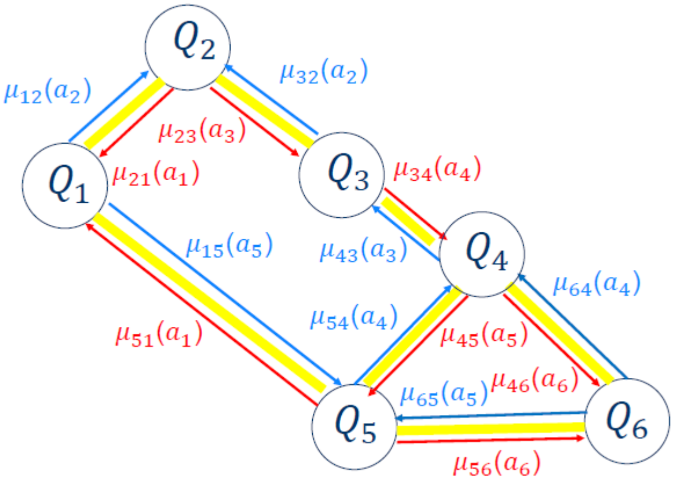


Fig. 3. Coordination Graph with 6 agents. Agent 3 sends messages $\mu_{32}(a_2)$, $\mu_{34}(a_4)$ to its neighbors, agents 2 and 4.

As mentioned before, the Max-Plus centralized coordination problem without cost is to find the optimal joint action $\boldsymbol{a}^*$ that maximizes $Q(\boldsymbol{a})$, defined in (9) as:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} Q(\boldsymbol{a}) \tag{9}$$

Please note that cumulated payoff of the agent i from all its neighbors in (8) is already included in the local payoff of agent j in (7). The exact solution of (9) is the Variable Elimination (VE) algorithm, which unfortunately is not an anytime algorithm.

B. Distributed Iterative Max-Plus Algorithm without Cost

The distributed version of the Max-Plus algorithm without cost is much more complex in terms of implementation and requires an increased number of iterations because, without the central coordinator, the agents do not have access to the factored global payoff function given by (7). The Global Reward should be evaluated within the MAS and communicated to all agents. Therefore, the agents participating in the distributed algorithm will have enhanced capabilities compared to the agents in the centralized algorithm. We need to distinguish among evaluated Global Reward G in distributed MAS. Instantaneous Global reward at iteration m, G_m, and initial desired Global Reward, G_0, is presented to all agents at initial time step of the horizon, $t = 0$. We will assume a distributed synchronous coordination.

The global payoff function $Q(\boldsymbol{a}^*)$ after calculating with (7) of every synchronous time step t must be calculated and stored. A trivial approach would be to centralize the value $Q(\boldsymbol{a}^*)$ to a single agent in every time step. This agent would then inform the other agents each time an improved solution is obtained. However, this method has two drawbacks: (1) the increased of number of messages, and (2) violation of agents' privacy caused by the need to inform a single agent (not necessarily a neighbor) about the joint payoff function $Q_{ij}(a_i, a_j)$, which represents the payoff of coordinated action of two other agents i and j.

An elegant solution to overcome these drawbacks is to use a Spanning Tree (ST), $G_t = (V, E_t)$, where at each time step t, the number of agents is fixed (V), while the number of connections (E_t) depends on the MAS configuration at that time. We still consider the case where all agents are connected (the domain is connected). An example of a ST associated with the CG given in Fig. 3 is represented in Fig. 4. In general, a ST associated with a CG is not unique and should be known by all the agents at every time step t. If the ST is fixed (as we assume for simplicity in this paper) then it will be initialized at the beginning of the Distributed Max-Plus algorithm.

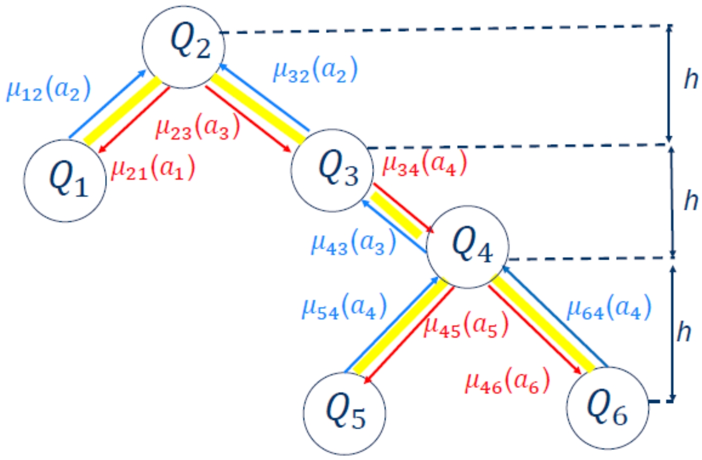


Fig. 4. Example of Spanning Tree graph associated with CG from Fig. 3.

In a ST, each agent receives information from its children, calculates the resulting payoff including its own contribution, and passes it to its parents. For example, in Fig. 4, agent 2 is the *root*, which makes the final calculation based on the payoff received from its children, agents 1 and 3, respectively. Agent 3 is also a parent (*rooted*) and will consider the payoff from child agent 4 in its calculations. Agents 1, 5, and 6 are leaves, which report to their parents.

After calculating the global payoff value $Q(\boldsymbol{a}^*)$ in this ST, agent 2 will communicate this value to other agents 1, 3, 4, 5, and 6 by propagating down in the ST. By this mechanism, all the agents in the distributed MAS will be aware of the global payoff value at every time step t.

Another important issue is knowing the status of every agent and when they will start their local algorithm. Every agent is in *waiting* mode and will initiate the process of calculating the Global Reward in MAS when it *believes* that it will have a significant contribution to it. This happens when an agent has an increased change in its local payoff or/and the edge payoff with its neighbor j.

In the distributed Max-Plus algorithm without cost, agent i at any iteration m may receive four types of messages m_l, $1 \leq l \leq 4$, sent by agent j, and therefore his response to agent j should be matched to the incoming messages. Any agent can initiate the exchange of messages at any time. In addition, we need to distinguish whether agent i is a leaf or root as in Fig. 5a and Fig. 5b.

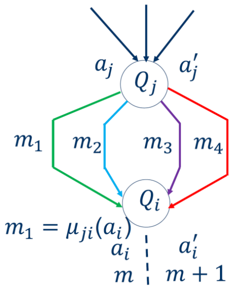


Fig. 5a: leaf (terminal)

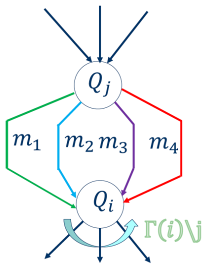


Fig. 5b: root (parent)

The main difference between the leaf (terminal node) and the root (parent) is the computational time. If agent i is a leaf, then it can respond immediately in the next iteration $m+1$ with its optimal action $a_i^{'}$. Otherwise, agent a_i must wait for payoff calculation from its children denoted by $\Gamma(i)\backslash j$ as in Fig. 5b. Please note the difference between the actions and reported payoff as explained below.

The four types of incoming messages received from agent j are explained next.

1) Message $m_1 = \mu_{ji}(a_i)$ (green line in Fig. 5a) is a typical Max-Plus message received from agent j when agent i is taking action a_i . Agent i will respond with message $m_1' = \mu_{ij}(a_j)$ and will start exchanging messages until local convergence is achieved: $m_1 \approx m_1'$ (no improvement of the local reward or no significant difference of the messages exchanged). The final action of agent i is the action a_i' as in (9). Agent j initiated the process when it observed a significant change in the reported payoff from agent i, $m_1 \neq m_1'$. The same process can also be started by agent i, which observed a significant change in the reported payoff of agent j. When local convergence is reached, the action presented by agent i to its neighbors is a_i' (which is not necessary the optimal one a_i^*). Since the local information of agent i is modified and possibly improved at convergence, agent i also believes the team Global Reward has been improved. In this case, after local convergence, agent i will ask for a payoff evaluation in the distributed MAS system and agent j will respond by sending the message m_2. Before asking for this payoff evaluation, the action of agent i is a_i' and its reward is:

$$r_i = Q_i(a_i') + \frac{1}{2} Q_{i,j}(a_i', a_j') \tag{10}$$

since agent i must share the edge reward with agent j, which also contributed to r_i through its action a_j'. In (7), each agent on the edge contributes half of the payoff function $Q_{ij}(a_i, a_j)$.

The Distributed Max-Plus Algorithm runs more iterations compared to the Centralized Max-Plus Algorithm. The centralized version run at most M iterations in order to converge to the fixed point. Usually, this number is small ($M \leq 10$) and depends on the particular application. If it does not converge within M iterations, the Centralized Max-Plus algorithm will stop. For the Distributed Max-Plus version, there are additional hops h as illustrated in Fig. 4. To send the evaluated Global Reward in the particular case illustrated in Fig. 4, which has three hops, there are 6 additional steps necessary (3 upward and 3 downward) to propagate the evaluated Global Reward up to agent 2 from agents 1, 3, 4, 5, and 6. There are another three hops to send the evaluated Global Reward from agent 2 down to agents 1, 3, 4, 5, and 6. For N agents, the worst case scenario is $2(N-1)$ steps, so the distributed Max-Plus algorithm will run at most $1 \leq m \leq M + 2(N-1)$ iterations.

Let us assume that the *anytime* Global Reward value in the distributed coordination Max-Plus algorithm at iteration $m \in \{1,2,\dots,M+2(N-1)\}$ is G_m. Since we assume a distributed synchronous coordination, each agent i is synchronized at iteration m. Each agent will proceed to iteration $m+1$ without a coordinator. There is no possibility that any other agent j will start iteration $m+1$. The rest of the messages (m_2, m_3, m_4) described below are necessary to calculate the anytime Global Reward G_m and the *evaluated* Global Reward in MAS system denoted by G. These two payoff values G_m and G are compared against the desired Global Reward denoted by G_0 presented to all agents at the initial time step of the horizon time step $t = 0$.

The Evaluated Global Reward (anytime G_m or at the end of the horizon time T) will be calculated in three successive phases: a) request message for payoff calculation sent to agent i, b) request message for payoff accumulation in the system and c) request message to calculate the Global Reward of the team that will be shared by all neighbors.

2) Message $m_2 = evaluate\ (j)$ (blue line in Fig. 5a) is a request for payoff evaluation sent to agent i from agent j. Agent i will respond with message $evaluate\ (i)$. This process is illustrated in Fig. 5c and Fig. 5d depending on whether the agent i is a leaf or root. When agent i receives this request and is a leaf (Fig. 5c), it will lock the best action a_i' found so far and will respond with it. In its response to agent j, agent i communicates the best action found so far:

$$a_i' = \arg\max_{a_i}\{Q_i(a_i) + \mu_{ji}(a_i)\} \text{ and his local payoff } r_i = Q_i(a_i) + \frac{1}{2}Q_{i,j}(a_i, a_j) \qquad (11)$$

No further action is required from agent i.

If agent i is a root (Fig. 5d), it will not lock its action a_i'. It will send the message $evaluate\ (i)$ to all of its children. As a root, agent i is asking all of its children to report their payoffs and will then initiate the calculation of the payoff accumulation (which will be explained next). Agent i will lock its individual action after the evaluation.

3) Message m_3 (purple line in Fig. 5e and Fig. 5f) is a request from agent j to calculate the accumulated payoff in the MAS system. Agent j will sent its local reward r_j accumulated so far to agent i as illustrated in Fig. 5e (leaf) and Fig. 5f (root), respectively. If agent i is a leaf, then it will modify its accumulated local reward to $r_i = r_i + r_j$ and will stay in this state until it receives the last message for calculating the Global Reward for the team as in Fig. 5g. If the agent is a root (parent) it will send a request message to calculate the global payoff to all of its children (purple arrows) by sending value $r_i + r_j$ as in Fig. 5f. As a parent, it will receive the payoff for all of its children (red line) and send the accumulated payoff to its parent agent j (red line) in Fig. 5f. In other words, after receiving the message *calculate the accumulated payoff*, agent i will send the information regarding its local payoff either back to neighbor j or to all of its children.

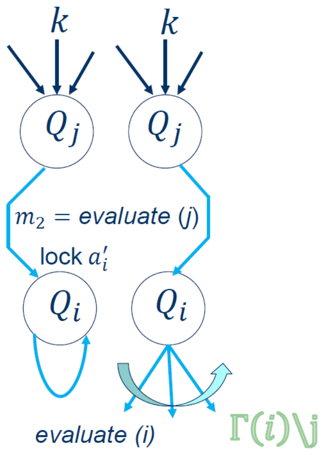


Fig. 5c: leaf Fig. 5d: root

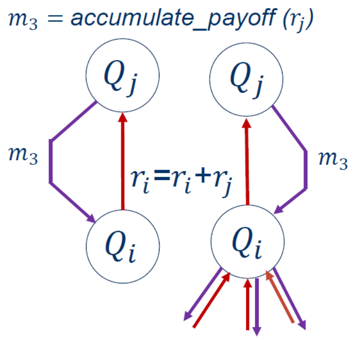


Fig. 5e: leaf Fig. 5f: root

Note that when agent i receives the accumulated rewards from all of its children, it might choose a different action a_i' found so far in MAS. Since this action is fixed in iteration m, the agent must unlock this action for calculating the global payoff of the MAS for the next iteration $m+1$ of the Distributed Max-Plus algorithm.

4) The last message m_4 (red line in Fig. 5a and Fig. 5g) is the message sent by agent j to agent i, and it contains the information about the evaluated Global Reward G of the team the has been calculated so far. Agent i was not aware of this value when it noticed an increase in its local payoff and it believed that it could make a significant contribution to the MAS evaluated Global Reward. Agent i will check whether it must continue its local process of finding new optimal actions or not. To do so, it must compare the evaluated Global Reward G against instantaneous Global Reward G_m and finally to compare it against the imposed Global Reward G_0. This mechanism is illustrated Fig. 5g. and it is called "*anytime*".

After receiving message m_4, agent i will lock the action and will calculate its contribution to the team by calculating instantaneous Global Reward G_m. If its contribution to the team's instantaneous Global Reward G_m does not cause it to increase (NO in Fig. 5g), i.e., the best action found so far a_i' does not increase the team Global Reward G_m, it will ask for help from its children. Agent i will unlock a_i' and will repeat the optimization process. If its contribution increases the team Global Reward G_m (YES in Fig. 5g) at iteration m, then its selected action a_i' will be reported optimal a_i^* and the evaluated Global Reward G will become G_m. This instantaneous value will be communicated to its neighbors. In this case, its belief that it could contribute to the instantaneous Global Reward G_m was true. Now, agent i needs to know if the team should continue or stop. The anytime algorithm will STOP if the resources of agent i are exceeded, or if a deadline message arrived, or if the desired Global Reward is met. Checking for these exit criteria is a very important step. As mentioned before, it will allow the algorithm to escape from local maxima and the team will move forward to achieve the desired Global Reward assignment G_0 presented to all agents at initial time step of the horizon $t=0$.

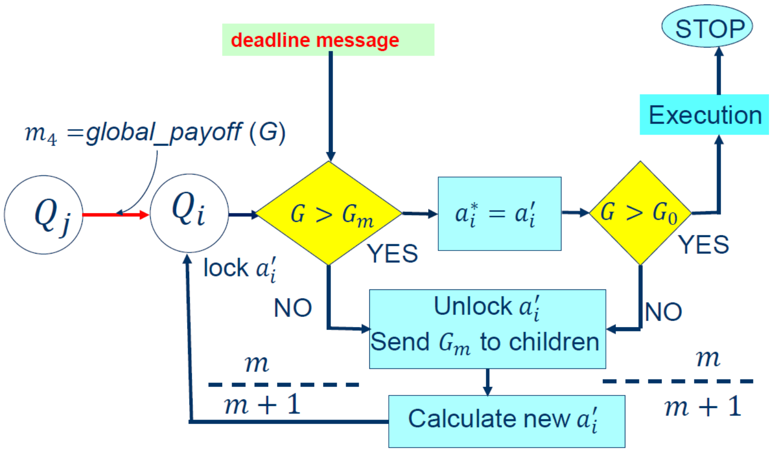


Fig. 5g: Evaluated Global Reward and the instantaneous Global Reward by the root agent of ST.

If the instantaneous Global Reward G_m and the evaluated Global Reward G are both greater than the desired Global Reward G_0 (YES option) then agent 2 (see Fig. 4) will stop its work and communicate its decision to the rest of the team. The actions could be executed if it is required. If not (NO option) then the agent will continue its optimization process. Therefore, the team will work together in distributed coordination until time T. This is a major difference in contrast with the centralized coordination version of Max-Plus. In the distributed implementation, in addition to the anytime aspect, each member of the team must check if the desired Global Reward G_0 is achieved. Therefore, in the distributed version, the agents require additional capabilities compared to the centralized version.

V. Distributed Max-Plus Algorithm with Cost

The iterative Max-Plus algorithm with limited budget and coordination graphs is a scalable and anytime algorithm, which will provide the best action a_i^* for any agent and the best global joint action $\boldsymbol{a}^*$**,** which will maximize the *factored* Global Reward *value* at a given global state $\boldsymbol{s}$ described below:

$$R(\boldsymbol{a}) = \sum\nolimits_{i\epsilon V} R\,(a_i) + \sum\nolimits_{(i,j)\epsilon E} R\big(a_i, a_j\big) \tag{12}$$

The Cost Distributed Max-Plus algorithm runs individually for every agent i and works with Spanning Tree (ST) coordination graphs. This feature is a major advantage over CG as used in previous works [1], [2], because some local interactions could become disconnected due to unreliable communication channels. The only requirement for our algorithm is for the domain to be "connected" regardless of the number of edges. At the onset of the algorithm, we will provide the following information to every agent i:

- The spanning tree coordination graph $G_t = (V, E_t)$, where each node V represents an agent, the number of agents is N, and each edge E_t defines the dependency between two agents at time step t**.** The number of all edges is $\mathcal{N}$.
- The costs of actions c_i for agent i and the costs $\mathcal{C}_{i,j}$ for any a pair of actions $\big(a_i, a_j\big)$.
- Number of iterations of the Max-Plus algorithm $1 \leq m \leq M + 2(N-1)$, horizon time limit T, $1 \leq t \leq T$.
- Actions for each agent i, A_i, $a_i \epsilon A_i$.
- We set the Global Reward in the Cost Distributed Max-Plus algorithm to the minimum value: $G_m = -\infty$.
- In the system for any agent: $\mu_{ij} = \mu_{ji} = 0$, for any $(i,j) \in E$, $a_i \in A_i$, $a_j \in A_j$ and for any agent i , $r_i = 0$ and $R(\boldsymbol{a}) = G_m = -\infty$.

The algorithm is described below. First, we would like to reach the convergence or the fixed point of the algorithm by calculating μ_{ij}, which are required for Global Reward. After that, the local payoff is calculated.

```
WHILE the fixed point is not reached, number of Max-Plus iterations $M$, horizon time $T$,
and cost budget are not reached for any agent // Root agent is evaluating this condition.
Wait for an incoming message $m_l$
 IF $m_1$
     FOR any iteration $m$, $1 \leq m \leq M + 2(N-1)$
          FOR all neighbors $j \in \Gamma(i)$
               a. compute with (4) and (5)
                  $\mu_{ij}\big(a_j\big) = Q_i(a_i) - c_i + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) + Q_{ij}\big(a_i, a_j\big) - \mathcal{C}_{i,j}$
               b. normalize the message $\mu_{ij}\big(a_j\big)$ for
                  convergence
               c. send the message $m_1 = \mu_{ij}\big(a_j\big)$ to the
                  agent $j$ if different from previous
               d. check if $\mu_{ij}\big(a_j\big)$ is closed the previous
                  message value
```

```
            END FOR all neighbors.
        Calculate the optimal individual action
          $a_i' = \arg\max_{a_i}\{0, [Q_i(a_i) - c_i + \sum_{k\in\Gamma(i)} \mu_{ki}(a_i)]\}$
        Determine $\boldsymbol{a}'$ the optimal global action so far including all previous $a_i'$
      END FOR any iteration
END IF $m_1$
IF $m_2$
      Lock $a_i'$, set $r_i = 0$ send the evaluation request to all children.
          IF agent $i$ is a leaf, then initiate the accumulation payoff
          END IF
END IF $m_2$
IF $m_3$
      $r_i = r_i + r_j$      // add payoff of child $r_j$
      IF the agent $i$ is a root send the global payoff to all children.
      ELSE send the global payoff to parent.
END IF $m_3$
IF $m_4$
          Calculate the evaluated global reward
          Use anytime:
              IF $R \geq r$
              THEN $a_i^* = a_i'$ ; $r = R(a_i')$
              ELSE $a_i^* = a_i'$
END IF $m_4$
END WHILE
Return the global reward $G = R(\boldsymbol{a})$.
```

VI. Hybrid Max-Plus Algorithm with Cost

In the Cost Hybrid Max-Plus algorithm, every agent i may receive a vector segmentation status message $\boldsymbol{s} = [1,0]$. Message $\boldsymbol{s}$ is a segmentation message (represented with a *dotted* red arrow Fig. 6) indicating the communication status to the centralized coordinator for all agents in the system. If $\boldsymbol{s} = 1$ for all agents at time step t, then no segmentation has occurred in the system for all N agents. All agents will continue in their centralized setting by using CG. However, if $\boldsymbol{s} = 0$ for some $N_2 < N$ agents at time step t, then a segmentation process has occurred. The group of N agents will split into two groups $N_1 + N_2 = N$. The N_1 agents will continue under the supervision of centralized coordinator and N_2 agents will execute the distributed Max-Plus algorithm. Later on, the above two groups may regroup again under centralized supervision.

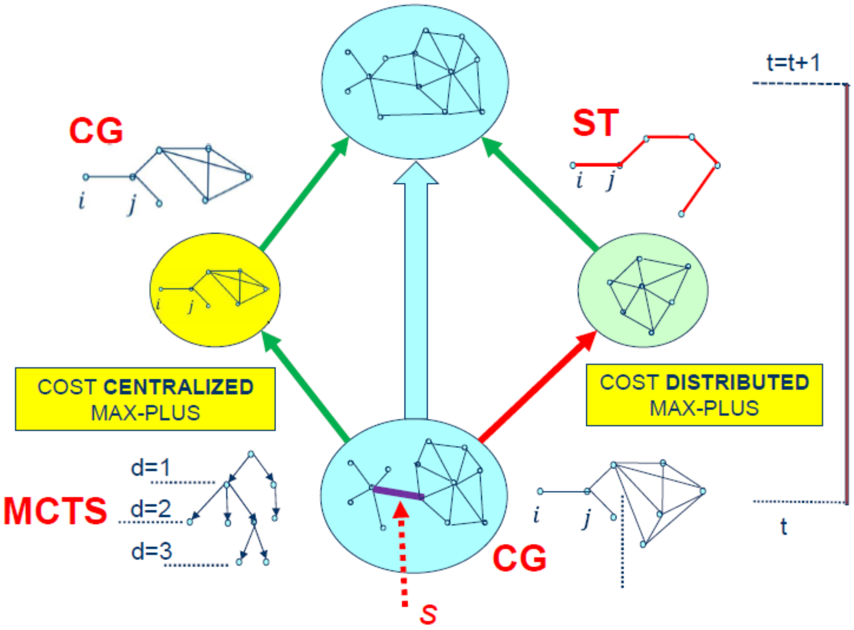


Fig. 6. Illustration of the segmentation process setting for Hybrid Factored-Value Cost Max-Plus method.

For $N = 6$ agents from our example in Fig. 3, the system will split into two groups by removing the connection between agents Q_3 , Q_4 and Q_1 , Q_4, respectively, when the segmentation message $\boldsymbol{s} = 0$ occurs. The group N_2 is formed by the agents Q_1, Q_2, Q_3 selected to be in a Spanning Tree configuration (decentralized) [4]. The group N_1 is formed by agents Q_4, Q_5, Q_6 selected to be in the centralized CG setting [3]. We have split them into equal groups for illustrative and numerical experimentation purposes. The group N_1 that is still able to communicate with the centralized coordinator will continue in their centralized setting (CG) (left green arrow in Fig. 6). The other group N_2 that lost communication with the coordinator will continue in a distributed Max-Plus algorithm (right red arrow in Fig. 6) in a Spanning Tree (ST) setting [4]. Later, at time step $t = t + 1$, if $\boldsymbol{s} = 1$ for all agents, then all agents will regroup again.

Another message received only by agents in group N_2 that lost connection with the central coordinator is $\boldsymbol{s} = 0$. Message $\boldsymbol{s} = 0$ is used by distributed Max-Plus algorithm and it tells each agent what part of the distributed Max-Plus algorithm should be executed. For example, it could be $m_1 = \mu_{ji}(a_i)$, which is a typical Max-Plus message (line 23 in the algorithm below). When the message is a request to calculate the accumulated payoff from agent j (line 30 in algorithm below), then the agent i will calculate the accumulated payoff in the MAS. The last form of incoming message (line 39) is the message sent by agent j to agent i, and it contains the information about the evaluated Global Reward of the team calculated so far.

We present pseudocode of the proposed (hybrid with cost) algorithm below. As inputs of our proposed algorithm, we provide the number of agents N, CG configuration, and the following information about every agent i:

- Actions for each agent i, A_i, $a_i \epsilon A_i$.
- Initialization by centralized coordinator for each agent: $\mu_{ij} = \mu_{ji} = 0$, for any $(i, j) \in E$, $a_i \in A_i$, $a_j \in A_j$ and for each agent i, $r_i = 0$ and $R(\boldsymbol{a})$.
- The costs of actions c_i for each agent i and the costs $\mathcal{C}_{i,j}$ for any pair of actions (a_i, a_j).

```
Wait for segmentation message s as in Fig. 6.
IF an agent receives the segmentation message s = 1 Go to the Cost Centralized Max-Plus
  algorithm given below: // All agents that receive s = 1
  WHILE the fixed point is not reached, time and cost budget are not reached // the centralized
  coordinator is evaluating this condition
    DO for any iteration $m$
      FOR any agent $i$
        FOR all neighbors $j \in \Gamma(i)$
          a. compute $\mu_{ij}(a_j) = Q_i(a_i) - c_i + \sum_{k\in\Gamma(i)\backslash j}\mu_{ki}(a_i) + Q_{ij}(a_i, a_j) - \mathcal{C}_{i,j}$
          b. normalize the message $\mu_{ij}(a_j)$
          c. send the message $\mu_{ij}(a_j)$ to the agent $j$
          d. check if $\mu_{ij}(a_j)$ is closed to the previous message (equivalent to reaching the
             convergence)
        END FOR all neighbors
        Calculate by centralized coordinator
                  $a_i^* = \arg\max_{a_i}\{0, [Q_i(a_i) - c(a_i) + \sum_{k\in\Gamma(i)}\mu_{ki}(a_i)]\}$
        Determine $\boldsymbol{a}^*$, the optimal global action so far including all previous $a_i'$
        // Use anytime:
        IF $R(\boldsymbol{a}) \geq r$ THEN $a_i^* = a_i'$ ; $r = R(a_i')$
        ELSE $a_i^* = a_i'$
        END IF
      END FOR every agent $i$
    END DO for any iteration $m$
  END WHILE
  Return the global reward $R(\boldsymbol{a})$
ELSE IF // All agents that receive s = 0
  Provide the Spanning Tree with algorithm in [18] and Go To the decentralized Max-Plus
  WHILE the fixed point is not reached, horizon time $T$, and cost budget are not reached
                                                          // Root agent is evaluating this condition
        IF $m_1$ = (regular Max Plus typical message given by (4))
          FOR any iteration $l$, $1 \leq l \leq M + 2(N-1)$
              FOR all neighbors $j \in \Gamma(i)$
                a. compute $\mu_{ij}(a_j)$
                   $\mu_{ij}(a_j) = Q_i(a_i) - c_i + \sum_{k\in\Gamma(i)\backslash j}\mu_{ki}(a_i) + Q_{ij}(a_i, a_j) - \mathcal{C}_{i,j}$
                b. normalize the message $\mu_{ij}(a_j)$ for convergence
                c. send the message $m_1 = \mu_{ij}(a_j)$ to the agent $j$ if different from previous
                d. check if $\mu_{ij}(a_j)$ is closed the previous message value
              END FOR // for all neighbors.
            Calculate $a_i'$ with (6) the optimal individual action
                   $a_i' = \arg\max_{a_i}\{0, [Q_i(a_i) - c(a_i) + \sum_{k\in\Gamma(i)}\mu_{ki}(a_i)]\}$
            Determine $\boldsymbol{a}'$ the optimal global action so far including all previous $a_i'$
          END FOR // all iterations
        ELSE IF $m_2 =$ (a request for payoff evaluation)
          Lock $a_i'$, set $r_i = 0$, send the evaluation request to all children.
        IF agent $i$ is a leaf initiate the accumulation payoff
      END IF
      ELSE IF $m_3$ = (request to calculate the accumulated payoff denoted by $r_i$ for agent $i$)
          $r_i = r_i + r_j$   // add payoff of child $r_j$
          IF the agent $i$ is a root send the global payoff to all children
          ELSE send the global payoff to parent
          END IF
      ELSE IF $m_4 =$ (evaluate Global Reward)
          Calculate the evaluated global reward
          Use anytime:
            IF $R \geq r$
                $a_i^* = a_i^{'}$ and $r = R(a_i^{'})$
            ELSE
                $a_i^* = a_i^{'}$
            END IF
        END IF // (message $s = 0$)
    END WHILE
    Return the best joint actions $\boldsymbol{a}^*$ and the global reward $R(\boldsymbol{a})$ accumulated so far
  END IF // (segmentation message $\boldsymbol{s} = 1$ or 0)
```

In the distributed Max-Plus algorithm, the anytime extension is more sophisticated. Hence, evaluation of the distributed joint action is only triggered by an agent when it is assumed to be worthwhile in this setup. Messages are sent in parallel, a factor that results in a computational advantage over sequential execution in centralized Max-Plus algorithms. However, a distributed Max-Plus algorithm is more complicated than a centralized Max-Plus algorithm because each agent must individually determine whether the system converges or whether to report its action.

Each agent in a distributed Max-Plus algorithm starts the propagation of an evaluation message over a spanning tree to compute its local contributions to global payoff, and then sends it to the parent node. In turn, a parent node accumulates all payoffs of its children and after adding its own contribution, sends the result to its own parent nodes before summing these payoffs together for all nodes.

VII. Factored-Value MCTS Hybrid Cost Max-Plus Method

Our main contribution is a new method for planning and acting called Factored-Value MCTS Hybrid Cost Max-Plus, which is depicted in Fig. 7. The method is a two-phase decision-making process:

I) In Phase I (Fig. 7, green box), the MCTS algorithm is run independently (MCTS depth steps are denoted by d) by each agent, producing a small, ordered set of its best potential actions. (We do not present the MCTS results [19] here. We focus only on the Cost Max-Plus algorithm.) For every agent in the state s, at time step t, the initial set of actions $\mathcal{A}_i$ is now reduced to the new set $\mathcal{A}_i^k = \{a_i^1, \dots, a_i^k\}$ where $k < |\mathcal{A}_i|$, as in Fig. 7, where an example is illustrated for three actions: $k = 3$. At the end of Phase I, every agent will present this ordered selected set to the team. After Phase I, the segmentation message may occur and Phase I can restart later at any time $t \leq T$ when the agents are regrouped in state s' as in Fig. 6 and Fig. 7. This segmentation process is represented by a red feedback connection in Fig. 7.

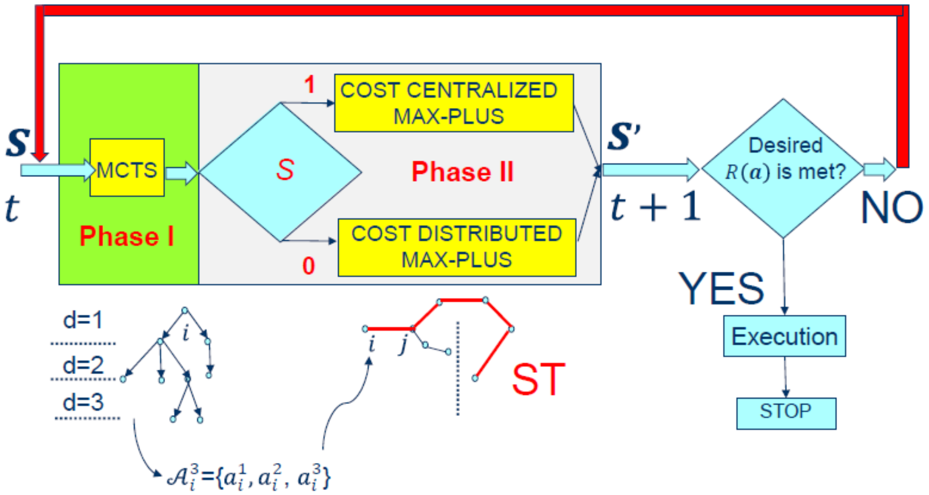

S
t
MCTS
Phase I
1
S
0
COST CENTRALIZED MAX-PLUS
Phase II
COST DISTRIBUTED MAX-PLUS
S'
$t+1$
Desired $R(a)$ is met?
NO
YES
Execution
STOP
d=1
d=2
d=3
i
i
j
ST
$\mathcal{A}_i^3=\{a_i^1, a_i^2, a_i^3\}$

Fig. 7. Factored-Value MCTS Hybrid Cost Max-Plus method.

By reducing the branching factor $|\mathcal{A}_i|$ for every agent, the number of actions per state $\mathcal{A}_i$ will be substantially reduced, since the possible action space for every agent has a cardinality that is exponential in the time horizon. Our approach in reducing the branching factor $|\mathcal{A}_i|$ for every agent is better than contracting the sample space at random as in [8] or choosing the actions using the uniform probability distribution as in [12], since the most promising actions will be allocated the highest probability for planning and execution.

II) In Phase II, when the segmentation message $\boldsymbol{s}$ occurs, the initial group of agents will be split into two subgroups. If $\boldsymbol{s} = 1$ (line 2 of the algorithm in Section V), then a small group of agents will continue the decision-making process as in Fig. 2 by using the Cost Centralized Max-Plus algorithm. In the same period of time, the other subgroup, which lost their connection to the centralized coordinator ($\boldsymbol{s} = 0$), will use the Cost Distributed Max-Plus algorithm described by lines 21 through 49 in the pseudocode. Please note that the number of iterations of running the cost Max-Plus algorithm in a centralized and decentralized setting could be different, as shown in our experiments. In any case, the results will be collected at time step $t + 1$.

At time step t, the MAS is in global state $\boldsymbol{s} \in \mathcal{S}$ and in the next time step $t + 1$ the system will transition to the new global state $\boldsymbol{s}'$. The transition probability to the next state $\boldsymbol{s}'$ is $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = 1$ (i.e., it is certain) if the desired Global Reward $R(\boldsymbol{a})$ is not met by the team as in Fig. 7 (NO option). If $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = 0$, the MAS will stay in the same state $\boldsymbol{s}$ or $\boldsymbol{s}'$ with $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = 0$ if the desired Global Reward is met (YES option). In the latter case, MAS agents execute the best joint action $\boldsymbol{a}^* = (a_1^*, a_2^*, \dots, a_i^*, \dots, a_N^*)$ found so far if it is required by the team, and the algorithm will stop.

Our approach improves upon the centralized MCTS solution proposed in [1] where, by simulation of the global state $\boldsymbol{s}$, the final coordinated actions of the team are found and therefore there is no guarantee that the team will move to the next global state.

In this paper, we only present the numerical results of the Payoff Value and Convergence of reaching the fixed point to understand the performance characteristics of the hybrid algorithm. Convergence is a major concern for any online approach and reaching the imposed payoff value is related to the efficiency of the proposed algorithm. Our results are given below in the form of graphical plots in Fig. 8 (Cost Distributed Max-Plus algorithm) and Fig. 9 (Cost Centralized Max-Plus algorithm), respectively. The combination of reached Payoff value and the convergence of the algorithms produce the four main scenarios of decentralized deterministic, decentralized randomized, centralized deterministic, and centralized randomized.

In the plots, the red lines represent deterministic payoff (i.e., taking actions in deterministic order) while the blue lines correspond to randomized actions. The reason for considering randomized actions (uniform distribution) is to compare with sequential decision-making systems. The payoff value range is represented on each plot by the red scale on the left vertical axis, while the convergence scale is represented by the green scale on the right vertical axis. In a three node MAS described as above configured in both distributed and centralized algorithms, the splitting segmentation process depends on real case at time t.

We consider the CG setting similar to Fig. 3 with 6 agents. When a segmentation message ***s*** occurs between agents Q_3 and Q_4 (red arrow) the group of 6 agents will be split in two groups where there is no connection between the agents Q_3 and Q_4. The first group of agents, Q_1, Q_2, Q_3, will be configured in a Spanning Tree configuration (distributed or decentralized) where the results are plotted in Fig. 8. The second group with the agents Q_4, Q_5, Q_6 (centralized in CG settings) will have the payoff value and the convergency performance plotted, respectively, as in Fig. 9. We kept the same cost of actions.

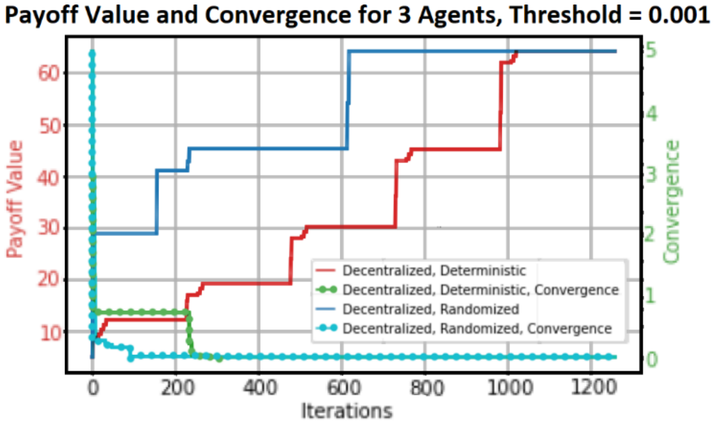


Fig. 8. Performance of Cost Distributed Max-Plus algorithm.

Each plot contains four graphical structures along with a convergence threshold. For a convergence threshold of 0.001, the centralized algorithm achieves the maximum payoff at about 374 iterations while the decentralized algorithm achieves the maximum payoff at about 1023 iterations. However, the distributed algorithm achieves convergence at about 215 iterations while the centralized algorithm achieves the convergence at about 344 iterations. Also note the different maximum payoff values. The maximum payoff value for the decentralized case is about 67, while the maximum payoff value for the centralized scenario is about 74.

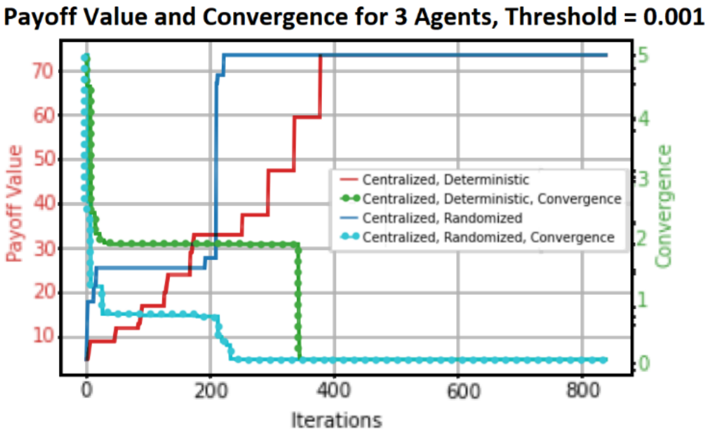


Fig. 9. Performance of Cost Centralized Max-Plus algorithm.

The reason why the payoff value is higher in the centralized scenario is that the centralized coordinator knows and can control the Global Reward (GR) or the payoff with a "global view" of all agents in the MAS. In the distributed case, local agents have only a "partial view" of the local environment. We noticed that the convergence of the distributed algorithm for the same number of agents is faster than the centralized one, however it takes more time to reach GR.

In a centralized setting (central coordinator), the agents are not allowed to exchange information with each other. A fully distributed coordination approach for MAS could allow local and correlated decisions for agents that can communicate. With only local rewards, the decentralized approach achieves less than the maximum network-wide reward that can be achieved by the centralized case.

In conclusion: (a) we have noticed that the convergence in both settings depends on number of total actions of agents in the system, (b) the distributed algorithm converges faster than the centralized algorithm, (c) the centralized algorithm arrives at a maximum payoff before the distributed algorithm, and (d) the centralized algorithm achieves higher maximum payoff than the distributed algorithm.

VIII. PROJECT OUTCOMES

This project resulted in a number of publications, which are listed here:

- P. Cotae, M. Kang and A. Velazquez, "A Scalable Real-Time Multiagent Decision Making Algorithm with Cost," *2021 IEEE Symposium on Computers and Communications (ISCC)*, Athens, Greece, September 2021.
    - We presented an algorithm for centralized coordination of a multiagent system in which the team makes a collaborative decision to maximize the global payoff. Our algorithm can be applied to real-time multiagent decision making problems in a collaborative setting and includes a cost factor for the planning and execution of actions.
- P. Cotae, M. Kang and A. Velazquez, "A Scalable Real-Time Distributed Multiagent Decision Making Algorithm with Cost," *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, January 2022.
    - We presented a distributed algorithm for a multiagent system in a collaborative setting. We revised the distributed Max-Plus algorithm by presenting a new Cost Distributed

    - Max-Plus (CD Max-Plus) algorithm that includes the cost of actions in the interactions of agents, named Factored Value-MCTS-CD Max-Plus.

- A. Velazquez, B. Montrose, M. Li, J. Luo, M. H. Kang, S. Patra and D. S. Nau, "ACRS4SDN: An Autonomous Cyber Response System for Software-defined Networks," *Defense Technical Information Center (DTIC)*, April 2022.
    - We introduced ACRS4SDN, a system to monitor for, and quickly respond to attacks and failures that may occur in a software-defined network (SDN) using automated acting and planning. We validated the performance of the system through experimentation on a real SDN across a series of cyberattack scenarios.
- A. Velazquez, J. T. Mathews, R. R. F. Lopes, T. Braun and F. Free-Nelson, "Toward autonomous cyber defense for protected core networking," *2023 International Conference on Military Communication and Information Systems (ICMCIS)*, Skopje, North Macedonia, May 2023.
    - We discussed the motivation for adding autonomous cyber defense capabilities to a type of coalition military network, called Protected Core Networking (PCN), and we outlined a path toward implementing these capabilities using existing reference architectures, frameworks, and enabling technologies, in order to adapt autonomous cyber defense concepts to the PCN context.
- P. Cotae, N. E. A. Reindorf, M. Kang and A. Velazquez, " A Hybrid Collaborative Multi Agent Decision Making Algorithm With Factored-Value Max-Plus," *2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, Istanbul, Turkey, July 2023.
    - We presented a real-time multiagent decision-making algorithm that combines a centralized algorithm and a distributed algorithm, which is especially useful in dynamic environments where network segmentation is unavoidable. Our algorithm has online, anytime, and scalable properties.

Additionally, a number of invited talks were given throughout the course of this project:

- M. Kang, "Autonomous Intelligent Resilient Security," presented at the *2nd Annual Technical Exchange on Autonomous Cyber Defense for Tactical Systems*, virtual, February 2022.
- A. Velazquez, "Autonomous Cyber Response for Protected Core Networking," presented to the *NATO Science & Technology Organization (STO) Information Systems Technology (IST) 162 Research Task Group (RTG) on Cyber Monitoring & Detection for Military Systems*, virtual, January 2023.
- A. Velazquez, "Autonomous Cyber Response for Software-defined Tactical Networks," presented at the *3rd Annual Technical Exchange on Autonomous Cyber Defense for Tactical Systems*, virtual, February 2023.
- A. Velazquez, " Toward autonomous cyber defense for protected core networking," presented at the *2023 International Conference on Military Communication and Information Systems (ICMCIS)*, Skopje, North Macedonia, May 2023.

IX. Conclusion and Future Work

We focused on the real-time multi agent decision-making problem with cost factor by using the Coordination Graphs and Spanning Tree settings. We introduced the Hybrid Cost Max-Plus algorithm for the first time. Our proposed method, Factored-Value MCTS Hybrid Cost Max-Plus, is online, anytime, distributed, and scalable in the number of agents and local interactions, which makes it particularly suitable for real world applications such as cybersecurity [19].

There are many future directions to improve the Global Reward with cost of MAS based on the locality of agent interactions. One future direction is using the recent advances in Deep Reinforcement Learning that have demonstrated the great potential of neural networks for function approximation in handling a large state space.

Another direction is inspired by Game Theory using “regret minimization techniques”. When the team decides a global optimal action, one or more agents may have “regret” in choosing their previous actions. Minimizing the counterfactual regret is equivalent to maximizing the Global Reward. The key idea is that information state qualitatively represents the data changing at a given node. Yet another direction is the information statistical approach for maximizing the information from a given state of MAS.

References

[1] Choudhury, Shushman, Jayesh K. Gupta, Peter Morales, and Mykel J. Kochenderfer. "Scalable Anytime Planning for Multi-Agent MDPs." *arXiv preprint arXiv:2101.04788* (2021).

[2] Amato, Christopher, and Frans Oliehoek. "Scalable planning and learning for multi agent POMDPs." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1. 2015.

[3] P. Cotae, M. Kang and A. Velazquez, "A Scalable Real-Time Multiagent Decision Making Algorithm with Cost," *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021, pp. 1-6, doi: 10.1109/ISCC53001.2021.9631510.

[4] P. Cotae, M. Kang and A. Velazquez, "A Scalable Real-Time Distributed Multiagent Decision Making Algorithm with Cost," *IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* 2022, pp. 745-746, doi: 10.1109/CCNC49033.2022.9700566.

[5] Kok, Jelle R., and Nikos Vlassis. "Collaborative multi agent reinforcement learning by payoff propagation." *Journal of Machine Learning Research* 7 (2006): 1789-1828

[6] Kok, Jelle R., and Nikos Vlassis. "Using the max-plus algorithm for multi agent decision making in coordination graphs." In *Robot Soccer World Cup*, pp. 1-12. Springer, Berlin, Heidelberg, 2005.

[7] Vlassis, Nikos, Rainout Elhorst, and Jelle R. Kok. "Anytime algorithms for multi agent decision making using coordination graphs." In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 1, pp. 953-957. IEEE, 2004.

[8] Best, G., Cliff, OLM., Patten, T., Mett, R.R. and Fitch, R., 2019. Dec-MCTS: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, *38*(2-3), pp. 316-337.

[9] Wainwright, Martin, Tommi Jaakkola, and Alan Willsky. "Tree consistency and bounds on the performance of the max-product algorithm and its generalizations." *Statistics and computing* 14, no. 2 (2004): 143-166.

[10] Landgren, Peter, Vaibhav Srivastava, and Naomi Enrich Leonard. "Distributed cooperative decision making in multi-agent multi-armed bandits." *Automatica* 125 (2020): 109445.

[11] Böhmer, Wendelin, Vitaly Kurin, and Shimon Whiteson. "Deep coordination graphs." In *International Conference on Machine Learning*, pp. 980-991. PMLR, 2020.

[12] Gupta, Jayesh Kumar. Modularity and Coordination for Planning and Reinforcement Learning. PhD thesis Stanford University, 2020.

[13] de Nijs, Frits, Erwin Walraven, Mathijs De Weerdt, and Matthijs Spaan. "Constrained multi agent Markov decision processes: A taxonomy of problems and algorithms." *Journal of Artificial Intelligence Research* 70 (2021): 955-1001.

[14] Guestrin, Carlos, Daphne Koller, and Ronald Parr. "Multi agent Planning with Factored MDPs." In *NIPS*, vol. 1, pp. 1523-1530. 2001.

[15] Guestrin, Carlos, Michail Lagoudakis, and Ronald Parr. "Coordinated reinforcement learning." In *ICML*, vol. 2, pp. 227-234. 2002.

[16] Bernstein, Daniel S., Robert Givan, Neil Immerman, and Shlomo Zilberstein. "The complexity of decentralized control of Markov decision processes." *Mathematics of operations research* 27, no. 4 (2002): 819-840.

[17] Revach, Guy, Nir Greshler, and Nahum Shimkin. "Planning for Cooperative Multiple Agents with Sparse Interaction Constraints." (2020).

[18] Pettie, Seth, and Vijaya Ramachandran. "An optimal minimum spanning tree algorithm." *Journal of the ACM (JACM)* 49, no. 1 (2002): 16-34.

[19] Patra, Sunandita, A. Velazquez, Myong Kang, and Dana Nau. "Using online planning and acting to recover from cyberattacks on software-defined networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, pp. 15377-15384. 2021.

[20] Czech, Johannes. "Distributed Methods for Reinforcement Learning Survey." In *Reinforcement Learning Algorithms: Analysis and Applications*, pp. 151-161. Springer, Cham, 2021.

[21] Li, Ruoxi, Sunandita Patra, and Dana S. Nau. "Decentralized Refinement Planning and Acting." In *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 31, pp. 225-233. 2021.

[22] Hayes, Conor F., Mathieu Reymond, Diederik M. Roijers, Enda Howley, and Patrick Mannion. "Risk Aware and Multi-Objective Decision Making with Distributional Monte Carlo Tree Search." *arXiv preprint arXiv:2102.00966* (2021).

[23] Rossi, Federico, Saptarshi Bandyopadhyay, Michael T. Wolf, and Marco Pavone. "Multi-Agent Algorithms for Collective Behavior: A structural and application-focused atlas." arXiv preprint arXiv:2103.11067 (2021).

[24] Grover, Divya, and Christos Dimitrakakis. "Adaptive Belief Discretization for POMDP Planning." arXiv:2104.07276 (2021).

[25] Fioretto, Ferdinando, Enrico Pontelli, and William Yeoh. "Distributed constraint optimization problems and applications: A survey." *Journal of Artificial Intelligence Research* 61 (2018): 623-698.

[26] Mahajan, Anuj, Mikayel Samvelyan, Lei Mao, Viktor Makoviychuk, Animesh Garg, Jean Kossaifi, Shimon Whiteson, Yuke Zhu, and Animashree Anandkumar. "Reinforcement Learning in Factored Action Spaces using Tensor Decompositions." *arXiv preprint arXiv:2110.14*