games

MDPI

Article

Collaborative Cost Multi-Agent Decision-Making Algorithm with Factored-Value Monte Carlo Tree Search and Max-Plus †

Nii-Emil Alexander-Reindorf [1] and Paul Cotae [2,*]

[1] Department of Computer Science, School of Engineering and Applied Sciences, The University of the District of Columbia, Washington, DC 20008, USA; niiemil.alexanderrei@udc.edu
[2] Department of Electrical and Computer Engineering, School of Engineering and Applied Sciences, The University of the District of Columbia, Washington, DC 20008, USA
* Correspondence: pcotae@udc.edu; Tel.: +1-210-396-0004 or +1-202-274-6290
† This paper is an extended paper of our paper published in 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkiye, 4–7 July 2023, pp. 438–443, doi: 10.1109/BlackSeaCom58138.2023.10299698.

Abstract: In this paper, we describe the Factored Value MCTS Hybrid Cost-Max-Plus algorithm, a collection of decision-making algorithms (centralized, decentralized, and hybrid) for a multi-agent system in a collaborative setting that considers action costs. Our proposed algorithm is made up of two steps. In the first step, each agent searches for the best individual actions with the lowest cost using the Monte Carlo Tree Search (MCTS) algorithm. Each agent's most promising activities are chosen and presented to the team. The Hybrid Cost Max-Plus method is utilized for joint action selection in the second step. The Hybrid Cost Max-Plus algorithm improves the well-known centralized and distributed Max-Plus algorithm by incorporating the cost of actions in agent interactions. The Max-Plus algorithm employed the Coordination Graph framework, which exploits agent dependencies to decompose the global payoff function as the sum of local terms. In terms of the number of agents and their interactions, the suggested Factored Value MCTS-Hybrid Cost-Max-Plus method is online, anytime, distributed, and scalable. Our contribution competes with state-of-the-art methodologies and algorithms by leveraging the locality of agent interactions for planning and acting utilizing MCTS and Max-Plus algorithms.

Keywords: Monte Carlo Tree Search; MCTS; Multi-Agent System; real-time decision making; distributed coordination; planning algorithm

Citation: Alexander-Reindorf, N.-E.; Cotae, P. Collaborative Cost Multi-Agent Decision-Making Algorithm with Factored-Value Monte Carlo Tree Search and Max-Plus. *Games* **2023**, *14*, 75. https://doi.org/10.3390/g14060075

Academic Editor: Ulrich Berger

Received: 12 October 2023
Revised: 19 November 2023
Accepted: 11 December 2023
Published: 17 December 2023

1. Introduction

The relationship between decision-making algorithms and game theory is characterized by their shared focus on the process of selecting optimal decisions within contextual parameters. Decision-making algorithms, which are frequently employed in the fields of computer science and artificial intelligence, frequently incorporate principles derived from game theory. The proposed algorithms in this paper have been specifically developed to facilitate decision-making processes or problem-solving tasks, considering many elements and potential outcomes, akin to the principles observed in game theory. Individuals engage in the process of evaluating various scenarios by making predictions regarding the conduct of others and ultimately selecting the optimal course of action.

Here, we consider a group of cooperating agents who are trying to reach the same decision point [1–8]. In Multi-Agent Systems (MAS), deciding between several potential courses of action can be computationally expensive and difficult. Fortunately, in many contexts, an agent must coordinate its actions with a small number of agents, and the other agents can behave autonomously with respect to one another. The problem can be made more manageable by factorizing the action space into that of a team of agents who coordinate their judgments. In Cooperative Multi-Agent Systems (CMAS) with a common

goal, coordination is essential for making good decisions. Each agent acts independently to achieve a locally optimal goal while also helping the team achieve a global one.

Coordinating the independent decision-making processes of each agent is an essential component of putting a policy for CMAS into action. This challenge of coordination might be handled by establishing a centralized coordinator for the entire system. This coordinator could decide what action each agent should perform and then convey this decision to the agents. This technique is not reliable since there is a possibility that the centralized coordinator may fail to work properly or that agents would struggle to communicate with one another. As a result, we will investigate the possibility of employing a hybrid approach that integrates both centralized and decentralized (or distributed) coordination. In the absence of a central unit (also known as a coordinator), the agents need to find a way to coordinate their actions toward the achievement of the common objective by employing a distributed algorithm that considers the costs of those actions. This type of algorithm is being proposed for the first time in this work.

Here, we would like to provide an *anytime algorithm* that can return a result at any time, improving with time until the best result is obtained. The joint action with the highest payout should be reported if the algorithm ends due to budgetary restrictions (time, cost of activities, etc.).

The single Markov Decision Process (MDP) modeling that is the foundation of MAS is generalized into a Multi-agent Markov Decision Process (MMDP), which adds the concept of numerous cooperating agents who have their own action sets each (as specified in Definition 1, page 962 [9]). It is possible for an MMDP to either centralize or decentralize the decision-making process. In algorithms with centralized decision makers, a single decision maker decides what actions should be taken by all the agents, but in algorithms with decentralized decision makers, each agent acts as its own decision maker. Our contribution is part of the Constraint Multi-agent Markov Decision Process (CMMDP) field and focuses on online planning [9]. The CMMDP presents a greater challenge in terms of obtaining a solution compared to the MMDP due to the limitations placed on the available resources. These limitations on resources influence the decisions that are made by the CMAS.

In general, the framework of the Coordination Graphs (CG) and the Variable Elimination (VE) algorithms that can be employed in this respect are what are used to solve an MMDP [1–7,9,10]. These methods can be used to solve MMDPs. This VE technique works by removing agents one at a time using a local maximization step. Its complexity increases exponentially with the size of the induced tree width, which is determined by the size of the largest clique that is produced during the process of eliminating nodes. Unfortunately, the VE method is not an anytime algorithm, and the Max-Plus algorithm, which is offered in both a centralized and a decentralized version in [3–5], is a solution for the anytime MMDP. The Max-Plus algorithm is a payout propagation algorithm. Using this algorithm, agents will exchange suitable payoff messages over the CG. Then, the agents will be able to compute their individual actions based on the convergence of this approach. There is only a limited amount of information available regarding the hybrid version of the Max-Plus method with constraints, which is the primary topic of this work.

The goal of our work is to come up with a hybrid solution for a constrained scalable decision-making method (centralized and distributed coordination) in MAS. The online methods for making decisions offer an alternative approach in which the agents must plan and carry out their actions at different times.

Online planning frequently uses the anytime method known as Monte Carlo Tree Search (MCTS). In the context of MCTS, dealing with a sizable decision space is an open problem that is constantly attracting attention. The scalability and huge branching factor of the MCTS method are its drawbacks, which prevent it from scaling well beyond a certain point. For the scaled coordinated coordination of agents, there are currently two prominent methods, [1,2], which include three components: (a) online planning using MCTS, (b) factored representation with CG, and (c) centralized Max-Plus method for joint

action selection. Ref. [1] does not present the distributed Max-Plus method and ignores the cost of action. Ref. [2] lacks the cost of actions and anytime aspects. To solve the centralized multi-agent POMDP model, the authors use MCTS. A cutting-edge algorithm is used as the answer to the multi-agent POMDP problem. But as the agent's degree of connection rises, it becomes unsolvable. Factored Value MCTS with Max Plus (FV-MCTS-Max-Plus) and FV-POMDP, respectively, are the names of the suggested methods in [1,2].

This is, as far as we are aware, the first paper to deal with hybrid value factored representation using MCTS for planning. This is a novel approach, and the justifications for this two-step decision-making procedure are as follows:

(1) Contingent upon the circumstance (state), different actions have different levels of relevance. Based on the current scenario, local-level decision-making determines the order of activities, from high likelihood to low probability in terms of effectiveness. Thus, increasing the likelihood that the best solution would be discovered early is a highly helpful step for the anytime method.
(2) In a dynamic context, network segmentation is unavoidable since network connectivity is necessary for global-level optimization in multi-agent environments. In some circumstances (such as those involving cyberattacks or network problems), this communication may not always be assured. The local optimal solution in the first step may be the best option in such scenarios because the algorithm may not be able to reach the global optimal solution in such hostile situations.

We favor fully hybrid coordinating for multi-agent decision making for the following reasons:

(a) In centralized arrangements, the central controller observes all agents jointly and makes joint choices for all agents. Each agent acts in response to the central controller's decision. Failure or malfunction of the central controller is equal to the entire MAS failing.
(b) To exchange information, the central controller must communicate with each agent, increasing the communication overhead at the single controller. This may reduce the scalability and robustness of MAS.
(c) In a centralized setup (centralized controller), agents are not permitted to exchange information about state transitions or rewards with one another. A hybrid MAS coordination strategy could allow each agent to interact to make local and correlated decisions.

Our contribution has three components:

a. We consider a budget constraint approach in which each action is assigned a cost. Different actions consume varying quantities of resources, which may be correlated to the team's global payoff [11]. In such a scenario, the goal of the local decision maker is to optimize his decision under cost and budget constraints at any given time. Consequently, the global team reward at each time step is calculated by subtracting the total cost incurred in analyzing the cost of the local actions. In this manner, we extend previous works [1–3] on centralized coordination where the only budgetary constraint was time.
b. We devise the Hybrid (i.e., centralized and distributed) coordination of the Max-Plus algorithm [3,4] where each agent computes and sends updated messages after receiving new and distinct messages from a neighbor. Messages are sent in parallel, which provides some computational advantages over the sequential execution of previous centralized coordination algorithms [1,2].
c. We developed a new FV-MCTS-Hybrid Cost-Max-Plus decision-making method with two stages. Our contribution is the development of a theoretical framework for integrating Monte Carlo Tree Search (MCTS) with the Cost Hybrid Max-Plus algorithm for decision making and execution. The proposed method is a suboptimal Dec-POMDP solution. Even for two agents, it is known that the exact solution to a Dec-POMDP is intractable and complete non-deterministic exponential (NEXP) [12].

Here is a summary of the paper's structure. Our presentation of the relevant literature is in Section 2. In Section 3, the mathematical foundation is laid. In Section 4, we examine and improve upon both cost-free variations of the Max-Plus algorithm, and in Section 5, we introduce the novel cost-based distributed Max-Plus algorithms. In Section 6, we looked at the Max Plus Hybrid Algorithm from a cost perspective. In Section 7, there is information about the FV-MCTS-Hybrid Cost-Max-Plus approach that was suggested. Section 8 contains our findings and suggestions for the future.

2. Related Work

The subject of multi-agent decision-making behavior coordination is a popular research topic that is tackled in various communities such as Game Theory, Reinforcement Learning, Decision Theory, Cybersecurity, Constraint Programming, Control, and Robotics, to mention a few [11,13–20]. The incorporation of game theory principles into algorithms facilitates the implementation of advanced and strategic decision-making processes inside intricate contexts. Although they function in distinct situations, the notions of agents in decision theory and players in game theory are comparable. According to decision theory, an agent is a person or thing that makes decisions, frequently considering their preferences, the information at hand, and possible results. The best solution for distributed agents sharing a Global Reward is of particular interest [1,6,8,10,21]. Figure 1 depicts a taxonomy of distributed and local optimization techniques for Multi-agent global behavior.

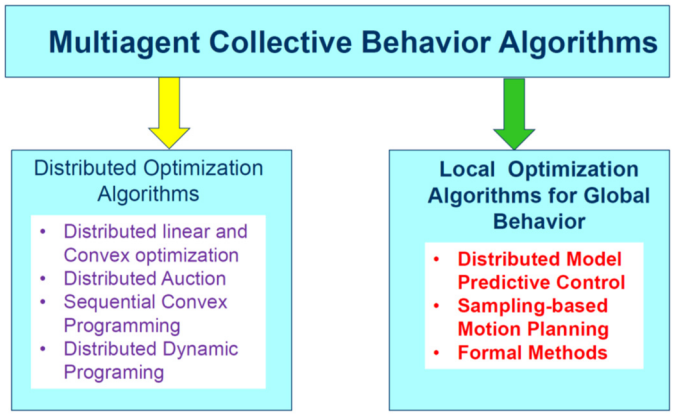


Figure 1. Distributed and local optimization algorithms.

This paper focuses on the coordination of centralized and distributed decision making in which agents interact while making decisions. In such CMAS, agents' prospective decisions are constrained by the cost of their actions. Little is known about the effect of action costs on decisions and the distributed coordination algorithms used to solve real-time distributed planning scenarios when agents share a common objective.

In the context of a multi-agent POMDP model, which is a centralized model, the concept of employing the MCTS algorithm to utilize the MAS structure was introduced for the very first time in [2]. The elegant technique that was proposed uses the precise method of solving the Coordination Graph (CG), which is known as Variable Elimination (VE) [3–5,9,10]. The Factored Value Partially Observable Monte Carlo Planning (FV-POMCP) method that is proposed in [2] is not *anytime*, which is unfortunate. This aspect was discussed in [1], in which the MCTS algorithm was utilized in conjunction with the *centralized* implementation of the Max-Plus method in order to pick joint actions. In [22], the strategy that was used was the same.

Our approach to the challenge of distributed coordination of multi-agent planning is distinct from that of [1,2,10], all of which take a centralized approach to solving the problem. In addition, the solutions that have been offered do not take into account the costs of the actions. Only in the first stage of our solution do we make use of the MCTS algorithm with the cost of [11], and this is to choose a limited, ordered collection of the promising actions

of agents that will be presented to the group in the subsequent stage. Our approach makes use of the Hybrid Cost Max-Plus method in the subsequent step. This algorithm will be presented later in this work.

The MCTS methodology utilized in [1,2] is utilized for two reasons:

(1) To advance agents to the following global state, where the CG structure does not change over time, by performing a global simulation from the current global state.
(2) To change each agent's joint action choice only in the last stage of the algorithm (after the centralized Max-Plus algorithm's convergence).

Additionally, the uniform distribution selections of activities as suggested in [22] will be improved by our hybrid method. The CG is used in other noteworthy works as suggested in [3–5,9,10] without the MCTS extension. Because of the naive implementation of the MCTS technique, the Dec-MCTS strategy utilized in [8] is unable to deal with the exponentially enormous action space it causes. In some circumstances, the proposed method in [8] is convergent.

3. Mathematical Background

We explore a cooperative multi-agent planning problem with a system that has N agents, a finite horizon called T, and an approach that takes place in discrete time. At every 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 team has decided on the joint action represented by the vector of team actions denoted by $\boldsymbol{a} = (a_1, a_2, \ldots, a_i, \ldots, a_N)$ and the reward function associated with the team is denoted by $Q(\boldsymbol{a})$. This function does not consider the cost of joint action. The goal of the classic coordination issue [6] is to determine 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.,

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}}{\operatorname{argmax}} Q(\boldsymbol{a}) \tag{1}$$

If one were to take a naive approach, they would list all the various joint actions that might be taken by all the agents and then choose the one that maximizes the global payoff function $Q(\boldsymbol{a})$. This is obviously unrealistic because the global team action space $\mathcal{A} = \times \mathcal{A}_i$ is an N fold cartesian product that grows exponentially as the number of agents N rises. Since the cost of actions is not negligible in some application domains, it is challenging to apply this model to some real-world situations because the payoff function Q(a) utilized in Equation (1) does not take this into account [11].

We then investigate 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 $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_N$. At time t, the MAS is in a global state $\boldsymbol{s} \in \mathcal{S}$, and in the next time step $t+1$, the system will transition to a new global state $\boldsymbol{s}'$. The probability of transitioning to the next state $\boldsymbol{s}'$ is $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$. We look at the most basic circumstances for factorizing the action space (FAS), where the system's state is entirely observed and available for determining an action for each component in the action space factorization. An MMDP can be used to model the aforementioned FAS [23].

The cost function c_i of doing 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. In the version that is distributed, we assume that only connected agents on an edge coordinate their local activities at a specific time t.

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

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}}{\operatorname{argmax}} R(\boldsymbol{a}) \tag{3}$$

The reward (payoff) function with the cost of 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 $C_{i,j}$.

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 the local state) and the cost of joint action $\mathcal{C}_{i,j}$, then a general message is sent from agent i to agent j (which is action a_j) is a scalar-valued function of the action space of the receiving agent j as given below:

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

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

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

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 is defined as $[\mu_{ij}(a_j)]^+ = \max[0,\ \mu_{ij}(a_j)]$. Finally, each agent computes its optimum action individually:

$$a_i^* = \underset{a_i}{\operatorname{argmax}} \{0, [Q_i(a_i) - c(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]\} \tag{6}$$

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

4. Both Variants of No Cost Max-Plus Algorithms

Max-Plus is a well-known algorithm for computing the maximal posterior configuration in an (unnormalized) undirected graph model. This algorithm is like the belief propagation (BP) algorithm or the sum-product algorithm in Bayesian Networks [5,7]. In this algorithm, two agents i and j send information regarding their locally optimized payoff values in an iterative manner. The earlier versions of this algorithm [1,2] did not incorporate the cost of actions and the mission benefit for a particular agent.

A. Centralized Max-Plus Algorithm with no Cost

The iterative sequential operation of iterations is utilized by the centralized Max-Plus algorithm. 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 are convergent 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.

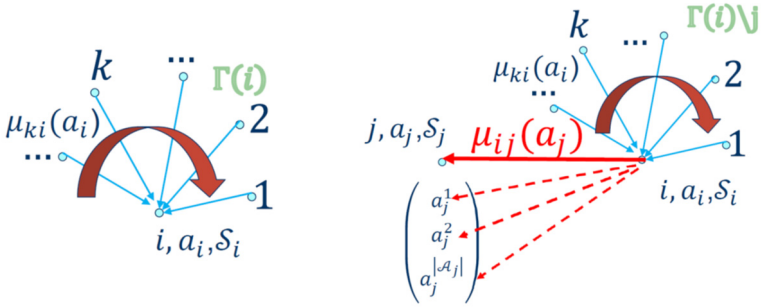
A coordination graph (CG) is a graph $G = (V, E)$ where each node V represents an agent and each edge E defines the dependency between two agents: $N = |V|$ and $\mathcal{N} = |E|$. The Max-Plus algorithm is scalable in terms of the number of agents N and their number

of local connections $\mathcal{N}$ in the CG representation [3]. In general, for any CG, we can *factor* the global payoff function *value* as follows:

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

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 an input, a pair of actions (a_i, a_j), and provides a real number. In the Max-Plus algorithm with no 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 Figure 2a. This agent sends a local message μ_{ij} that maps an action a_j of an agent j to a real number as shown in Figure 2b.



(**a**) (**b**)

Figure 2. Interaction on edge between two agents in centralized Max-Plus algorithm. (**a**): The agent i is a receiver from all its neighbors' payoffs. (**b**): The agent i is sending the message $\mu_{i,j}$ to agent j.

In Figure 2b, the transmitted message $\mu_{ij} : \mathcal{A}_j \to \mathbb{R}$ 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 the local payoff $Q_i(a_i)$, joint payoff $Q_{ij}(a_i, a_j)$, and the cumulated payoff from all its neighbors $\Gamma(i) \backslash j$ of node i except node j is as given below:

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

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

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

The agent i can send payoff messages for any action of agent j as in Figure 2b (dotted arrows). The agents keep exchanging messages at each iteration until they converge. It is proved 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 six agents is shown in Figure 3. From Agent 2's perspective, 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 roots (parents) are different from messages sent "down" to their children (leaves). 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 it their payoffs.

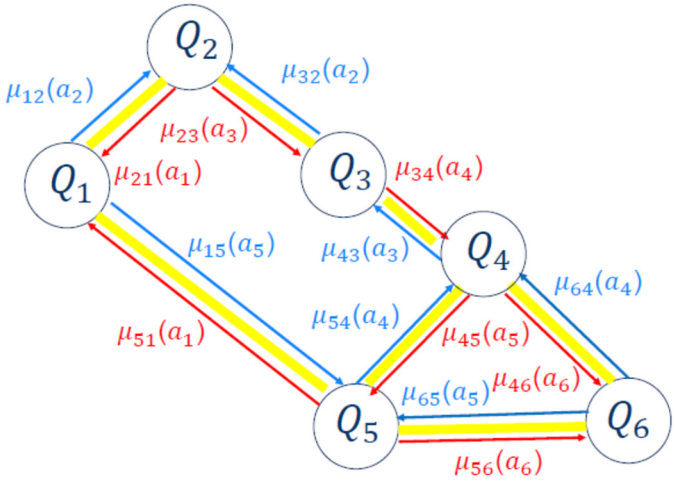


Figure 3. Coordination Graph with 6 agents. Agent 3 is sending messages $\mu_{32}(a_2)$, $\mu_{34}(a_4)$ to its neighbors 2 and 4.

As mentioned before, the Max-Plus centralized coordination problem with no cost is to find the optimal joint action $\boldsymbol{a}^*$ that maximizes $Q(\boldsymbol{a})$, defined in (9) as follows:

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}}{\text{argmax}} Q(\boldsymbol{a}) \tag{9}$$

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

B. Distributed Iterative Max-Plus Algorithm with no Cost

The distributed version of the Max-Plus algorithm with no cost is much more complex in terms of implementation and increased number of iterations since the agents have no 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 to compare those participating in the centralized algorithm. We need to distinguish between the evaluated Global Reward G in distributed MAS, the instantaneous Global reward at iteration m as G_m and the initial desired Global Reward as G_0 that is presented to all agents at the initial time step of the horizon $t = 0$. We assume a distributed synchronous coordination.

The global payoff function $Q(\boldsymbol{a}^*)$ after calculating using (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 a solution that improves the results on all previous solutions is obtained. However, this method has drawbacks both in terms of the increase in the number of messages and the violation of the agent's 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 the coordinated action of the other two agents i and j.

An elegant solution to overcome the above issue is to use a Spanning Tree (ST) $G_t = (V, E_t)$, where at each time step t, the number of agents is fixed as 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 ST associated with the CG given in Figure 3 is represented in Figure 4. In general, an 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.

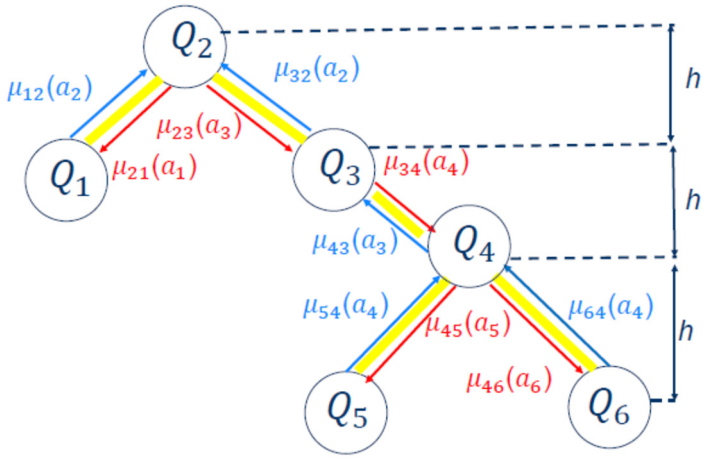


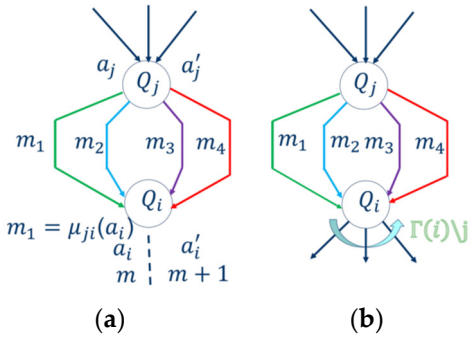
Figure 4. Example of spanning tree graph associated with CG from Figure 3.

In an ST, every agent receives information from his children, calculates the resulting payoff including their own contribution, and passes it to their parents. For example, in Figure 4, Agent 2 is the *root* that 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, in its calculations, it considers the payoff from its child, Agent 4. 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 communicates this value to other agents, i.e., 1, 3, 4, 5, and 6, by propagating down in the ST. Using this mechanism, all the agents in the distributed MAS are aware of the global payoff value at every time step t.

Other important issues are (1) the status of every agent in MAS and (2) the time when the agents start their local algorithm. Every agent is in *waiting* mode and any agent will initiate the process of calculating the Global Reward in MAS when it *believes* it will make a significant contribution to it. This happens when an agent has an increased change in its local payoff or/and *edge* payoff with its neighbor j.

In the distributed Max-Plus algorithm with no cost, the agent i at any iteration m may receive four types of messages $\Updownarrow_l$, $1 \leq l \leq 4$ sent by agent j, and therefore its response to agent j should match the incoming messages. As mentioned before, any agent can initiate exchange messages at any time. In addition, we need to distinguish if the agent i is a leaf or root, as shown in Figure 5a,b.



(**a**) (**b**)

Figure 5. (**a**) Leaf (terminal); (**b**) root (parent).

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

The incoming messages received from agent j are explained next.

(1) Message $\Updownarrow_1 = \mu_{ji}(a_i)$ (green line in Figure 5a) is a Max-Plus typical message received from agent j when agent i is taking action a_i. Agent i will respond with the message $m_1' = \mu_{ij}(a_j)$ and will start exchanging messages until the local convergency is achieved: $m_1 \approx m_1'$ (no improvement of the local reward or no significant difference in the messages exchanged). The final action of agent i is the action a_i' as in (9). Agent j initiates the process since it observes 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 observes a significant change in the reported payoff of agent j. When local convergency is reached, the action presented by agent i to its neighbors is a_i' (which is not necessarily the optimal one a_i^*). Since the local information of agent i is modified and possibly improved at convergency, agent i also believes that the team Global Reward has been improved. In this case, after local convergency, agent i will ask for a payoff evaluation in the distributed MAS system and agent j will respond in this regard by sending the message $\Updownarrow_2$. Before asking for this payoff evaluation, the action of agent i is a_i' and its reward is

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

since agent i must share the edge reward with agent j which also contributes to r_i reward via its action a_j'. In (7), the payoff function $Q_{ij}(a_i, a_j)$ contains the cumulated factor $\frac{1}{2}$ from each agent on the edge.

The distributed Max-Plus Algorithm runs more iterations than those of the Centralized Max-Plus Algorithm. The centralized version runs at most M iterations which are necessary for the convergency to reach the fixed point. Usually, this number is small $M \leq 10$ and depends on the applications. If the convergency cannot be reached within M iterations, the Centralized Max-Plus algorithm will stop. For the distributed Max-Plus version, there are additional hops h as illustrated in Figure 4. To send the evaluated Global Reward in the particular case illustrated in Figure 4, where there are three hops, there are necessary six additional steps (three upward and three downward) to propagate the evaluated Global Reward upward to Agent 2 from Agents 1, 3, 4, 5, and 6. There are three other hops to send the evaluated Global Reward from Agent 2 to Agents 1, 3, 4, 5, and 6. For N agents, the worst scenario is the $2(N-1)$ step, so the distributed Max-Plus algorithm runs at most $1 \leq m \leq M + 2(N-1)$ iterations.

Let us assume that *anytime* Global Reward value in the distributed coordination Max Plus algorithm at iteration $m \in \{1, 2, \ldots, M + 2(N-1)\}$ is G_m. Since we assume a distributed synchronous coordination, each agent i is synchronized at iteration m. Each agent will be able to go to iteration $m+1$ without a coordinator (controller). There is no possibility that any other agent j will start iteration $m+1$. The rest of the messages (m_2, m_3, andm_4) described below are necessary to calculate the time Global Reward G_m and the *evaluated* Global Reward in the 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 which is shared by all neighbors.

(2) Message $\Updownarrow_2 = evaluate\ (j)$ (blue line in Figure 5a) is a request for payoff evaluation sent to agent i from agent j. Agent i will respond with the message $evaluate\ (i)$. This process is illustrated in Figure 6a,b depending on whether agent i is a leaf or root. When agent i receives this request and it is a leaf (Figure 6a), it will lock the best action a_i' found so far and respond to it. In its response to agent j, agent i will communicate its best action found so far:

$a_i' = \arg\max_{a_i}\{Q_i(a_i) + \mu_{ji}(a_i)\}$ which equals the contribution of the individual function of agent i and different subtrees with the neighbors of agent i as root and its local payoff

$$r_i = Q_i(a_i') + \frac{1}{2}Q_{i,j}(a_i', a_i) \tag{11}$$

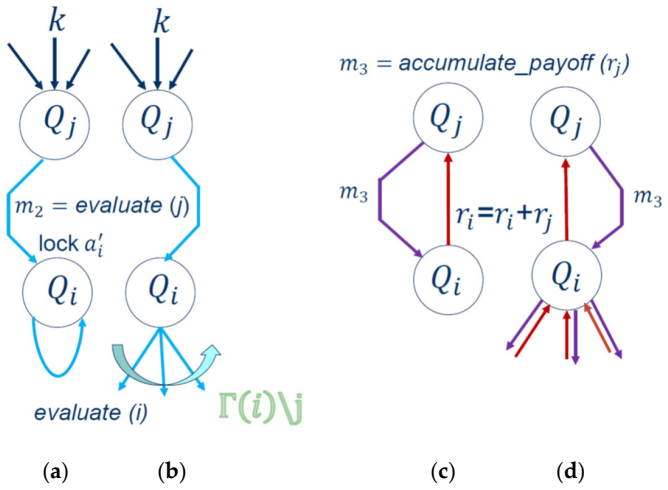


Figure 6. (**a**) Leaf, (**b**) root, (**c**) leaf, and (**d**) root.

No further action is required from agent i.

If agent i is a root (Figure 6b), it will not lock its action a_i'. It will send the message *evaluate* (i) to all its children. As a root, agent i is asking all its children to report their payoffs to it and it will initiate the calculation of the payoff accumulation, which is explained next. Agent i will fix its individual action after the evaluation.

(3) Message $\Updownarrow_3$ (purple line in Figure 6c,d) is a request from agent j to calculate the accumulated payoff in the MAS system. Agent j will send its local reward r_j accumulated so far to agent i as illustrated in Figure 6c (leaf) and Figure 6d (root), respectively. If agent i is a leaf, then it will modify its accumulated local reward to $r_i = r_i + r_j$ and it will stay in this state until it receives the last message for calculating the Global Reward for the team as shown in Figure 7. If the agent is a root (parent), it will send a request message to calculate the global payoff to all its children (purple arrows) by sending its value $r_i + r_j$ as shown in Figure 6d. As a parent, it will receive the payoff for all its children (red line) and send it to its parent, agent j (red line), as shown in Figure 6d. In other words, after receiving this message of *calculating the accumulated payoff*, agent i will send the information regarding its local payoff either back to the neighbor j or to all its children.

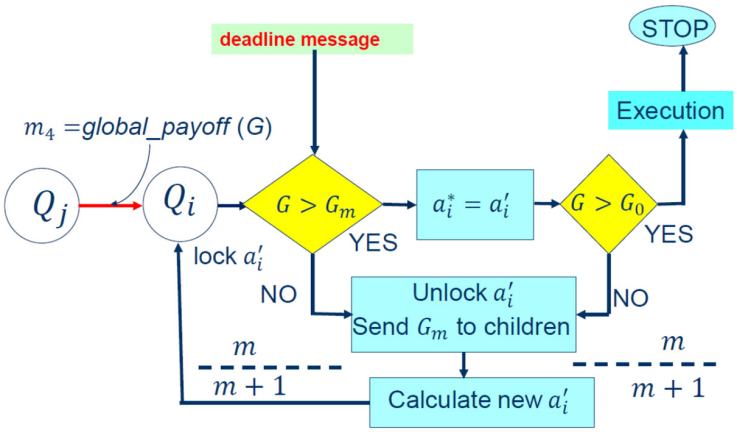


Figure 7. Evaluated Global Reward and the instantaneous Global Reward by the root agent of ST.

Note that when agent i is receiving the accumulated rewards from all its children, it might choose a different value for its 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.

The last message $\Updownarrow_4$ (red line in Figures 5a and 7) is the message sent by agent j to agent i, and it contains the information about the evaluated Global Reward G of the team calculated so far. Agent i is not aware of this value when it notices an increase in its local payoff, and it believes that it can make a significant contribution to the MAS evaluated Global Reward. Agent i will check if 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 the instantaneous Global Reward G_m and finally compare it against the imposed Global Reward G_0. This mechanism is illustrated in Figure 7 and it is called "*anytime*".

(4) After receiving the message $\Updownarrow_4$, agent i will lock the action and calculate its contribution to the team by calculating the instantaneous Global Reward G_m. If its contribution to team (instantaneous Global Reward) G_m is not increasing and it is less than the evaluated Global Reward so far (NO in Figure 7), i.e., the best action found so far a_i' is not increasing the team's Global Reward G_m, then it will ask for help from its children. Agent i will unlock a_i' and repeat its optimization process. If its contribution is increasing the team's Global Reward G_m (YES) at iteration, m, then its optimal action a_i' will be reported as an optional one a_i^* and the evaluated Global Reward G will become G_m. This instantaneous value will be communicated to its neighbors (team). In this way, its belief that it can contribute to the instantaneous Global Reward G_m is certain. Now, agent i needs to know if the team must move forward or stop. The anytime algorithm will STOP if the resources of agent i or a deadline message arrives, or if the desired Global Reward is met. This is a very important task. 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 the initial time step of the horizon $t=0$.

If the instantaneous Global Reward G_m (Figure 7) and the evaluated Global Reward G are both greater than the desired Global Reward G_0 (YES), then the root Agent 2 (see Figure 4) will stop working and communicate its decision to the rest of the team. The actions can be executed if it is required. If not (NO) then the agent will continue its optimization process. Therefore, the team will work together in a distributed coordination until the time T. This is a major difference in contrast with the Max-Plus centralized coordination version. 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 capabilities of the agent are enhanced in contrast with the centralized version.

5. Distributed Max-Plus Algorithm with Cost

The iterative Max-Plus algorithm with limited budget and coordination graphs is a scalable and anytime algorithm that provides 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:

$$R(\boldsymbol{a}) = \sum_{i \in V} R(a_i) + \sum_{(i,j) \in E} R\left(a_i, a_j\right) \tag{12}$$

The Cost-Distributed Max-Plus algorithm runs individually for every agent i and works with Spanning Tree (ST) coordinated graphs. This feature confers a major advantage against CG used in previous works [1,2] because some local interactions could be disconnected due to harsh communication channels. The only requirement for our algorithm is that the domain is "connected" regardless of the number of edges. The algorithm is provided in [24] and it is part of the Hybrid Max-Plus Algorithm with Cost as described next.

6. 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 in Figure 8) indicating the communication status to the centralized coordinator for all agents in the system. Initially, at moment t, all the agents are working together in a centralized scenario on a CG under the supervision of the centralized controller. Later, at $t = t + 1$, the agents will regroup together keeping the same complexity of the message under the same centralized controller. If $\boldsymbol{s} = 1$ for all agents at time step t, then no segmentation occurs in the system for all N agents. All agents will continue in their centralized setting by using CG until a segmentation message occurs. However, if $\boldsymbol{s} = 0$ for some $N_2 < N$ agents at time step t, then a segmentation process occurs. 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 coordinators (left side) on a CG, and N_2 agents (right side) will execute the distributed Max-Plus algorithm on a spanning tree (ST). Later, the two aforementioned groups may reorganize under centralized supervision.

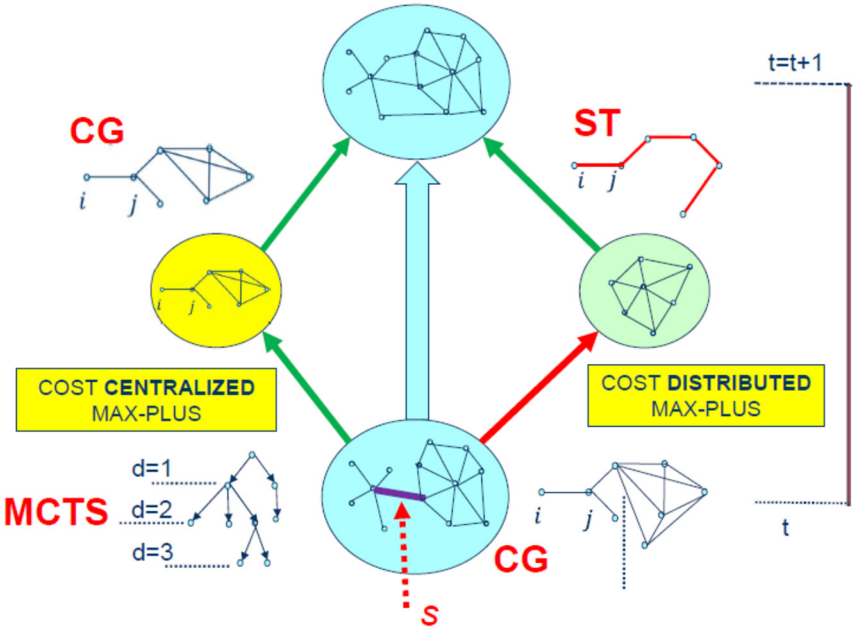


Figure 8. Illustration of the segmentation process setting for Hybrid Factored Value Cost Max-Plus method [24] (reproduced with permission from P. Cotae, N. E. A. Reindorf, M. Kang, and A. Velazquez, "Work-in-Progress: A Hybrid Collaborative Multi Agent Decision Making Algorithm with Factored-Value Max-Plus", 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkiye, 2023, pp. 438–443, doi: 10.1109/BlackSeaCom58138.2023.10299698).

For $N = 6$ agents, from our example in Figure 3, the system will split into two groups by cutting the link between the agents Q_3, Q_4 and Q_1, Q_4 respectively, as soon as the segmentation message $\boldsymbol{s} = 0$ is received. The grouping N_2 is composed of the agents Q_1, Q_2, Q_3 chosen to take part in a Spanning Tree formation (decentralized) [4]. The group N_1 consists of agents Q_4, Q_5, Q_6 chosen to participate in the centralized CG setting [3]. For the sake of illustration and numerical experimentation, we have divided them into equal groups. The group N_1 that can still connect with the centralized coordinator will remain in their centralized setting (CG) (left green arrow in Figure 8). The secondary group N_2 that is unable to maintain communication with the coordinator will continue in a distributed Max-Plus algorithm (right red arrow in Figure 8) in a Spanning Tree (ST) setting [4]. After a while, at time step $t = t + 1$, if $\mathbf{s} = 1$, every agent will reassemble into a group.

Another message received only by agents in the group N_2 that loses connection with the central coordinator is $\boldsymbol{s} = 0$. Message $\boldsymbol{s} = 0$ is used by the distributed Max-Plus algorithm and it typically tells each agent what part of distributed Max-Plus algorithm

should be executed. For example, it can be $\Updownarrow_1 = \mu_{ji}(a_i)$, 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 information about the evaluated Global Reward of the team calculated so far.

We present the pseudocode of the proposed cost hybrid algorithm as Algorithm 1 below. We provided at the input of our proposed algorithm the number of agents N, CG configuration, and the following information about every agent i:

- Actions for any agent $A_i, a_i \in A_i$.
- Initialization by the centralized coordinator 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 $R(\boldsymbol{a})$.
- The costs of actions c_i for any agent i and the costs $\mathcal{C}_{i,j}$ for any pair of actions (a_i, a_j).

Algorithm 1 Wait for segmentation message $\boldsymbol{s}$ as shown in Figure 8

1. IF an agent receives the segmentation message $\boldsymbol{s} = 1$ Go to the Cost Centralized Max Plus algorithm given below://All agents that receive $s = 1$
2. WHILE the fixed point is not reached, time and cost budget are not reached//the centralized coordinator is evaluating this condition
3. DO for any iteration m
4. FOR any agent i
5. FOR all neighbors $j \in \Gamma(i)$
6. a. compute $\mu_{ij}(a_j) = Q_i(a_i) - c_i + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) - \mathcal{C}_{i,j}$
7. b. normalize the message $\mu_{ij}(a_j)$
8. c. send the message $\mu_{ij}(a_j)$ to the agent j
9. d. check if $\mu_{ij}(a_j)$ is closed to the previous message (equivalent to reaching the convergence)
10. END FOR all neighbors
11. Calculate by centralized coordinator
$$a_i^* = \arg\max_{a_i} \Big\{0,\ [Q_i(a_i) - c(a_i) + \textstyle\sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]\Big\}$$
12. Determine $\boldsymbol{a}^*$, the optimal global action so far including all previous a_i'
13. //Use anytime:
14. IF $R(\boldsymbol{a}) \geq r$ THEN $a_i^* = a_i'; r = R(a_i')$
15. ELSE $a_i^* = a_i'$
16. END IF
17. END FOR every agent i
18. END DO for any iteration m
19. END WHILE
20. Return the global reward $R(\boldsymbol{a})$
21. ELSE IF//All agents that receive $s = 0$
22. Provide the Spanning Tree with algorithm in [15] 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
23. IF m_1 = (regular Max Plus typical message given by (4))
24. FOR any iteration l, $1 \leq l \leq M + 2(N-1)$
25. FOR all neighbors $j \in \Gamma(i)$
a. compute $\mu_{ij}(a_j)$
$\mu_{ij}(a_j) = Q_i(a_i) - c_i + Q_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) - \mathcal{C}_{i,j}$
b. normalize the message $\mu_{ij}(a_j)$ for convergence
c. send the message $\Updownarrow_1 = \mu_{ij}(a_j)$ to the agent j if different from previous
d. check if $\mu_{ij}(a_j)$ is closed the previous message value
26. END FOR//for all neighbors.
27. Calculate a_i' with (6) the optimal individual action $a_i' = \arg\max_{a_i}\{0,\ [Q_i(a_i) - c(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]\}$
28. Determine $\boldsymbol{a}'$ the optimal global action so far including all previous a_i'

Algorithm 1 *Cont.*

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 sends the global payoff to all children
ELSE sends 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)

The anytime extension of the distributed Max-Plus method is more complex. Therefore, in this system, an agent only initiates the evaluation of distributed joint action when it is deemed worthwhile. The fact that messages are transmitted in parallel gives this method a computational edge over centralized Max-Plus algorithms that execute sequentially. A distributed Max-Plus method, on the other hand, requires more work than a centralized one because each agent has to decide whether to report the action individually or for the system to converge.

In a distributed Max-Plus algorithm, each agent computes its local contributions to the global payout by initiating the propagation of an evaluation message via a spanning tree and then forwards it to the parent node. A parent node then adds its own reward to the total of all the payoffs of its children and sends the result to all of its parent nodes before combining the payoffs for all nodes.

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

The primary innovation we offer is the Cost Hybrid Factored Value MCTS Max-Plus method (Figure 9), a new approach to planning and acting. Decision making is performed using this two-phase method:

(I) For Phase I (Figure 9, green), each agent individually runs the MCTS algorithm (MCTS depth steps are marked by d), resulting in a condensed, ordered list of its best possible actions (we will not discuss the MCTS findings [11] here). We limit our attention to the Cost MP algorithm. For each and every state agent s, at time step t, the initial set of actions $\mathcal{A}_i$ is now reduced to the new set $\mathcal{A}_i^k = \left\{a_i^1, \ldots, a_i^k\right\}$ where $k < |\mathcal{A}_i|$ as in Figure 9 where an example is illustrated for three actions: $k = 3$. After Phase I is over, each agent will give the team this sorted, selected set. 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 the state s' as shown in Figures 8 and 9. In Figure 9, this segmentation process is depicted by a red feedback connection.

Given by decreasing the branching factor $|\mathcal{A}_i|$ for every agent, the number of actions per state $\mathcal{A}_i$ will be significantly diminished, as the potential action space for each agent exhibits an exponential cardinality in relation to the time horizon. The proposed methodology

for decreasing the branching factor $|\mathcal{A}_i|$ for each agent is superior to randomly contracting the sample space as described in reference [8], or selecting actions based on a uniform probability distribution as mentioned in reference [10]. This is due to the fact that our method gives the most likely activities for preparation and implementation the highest probability.

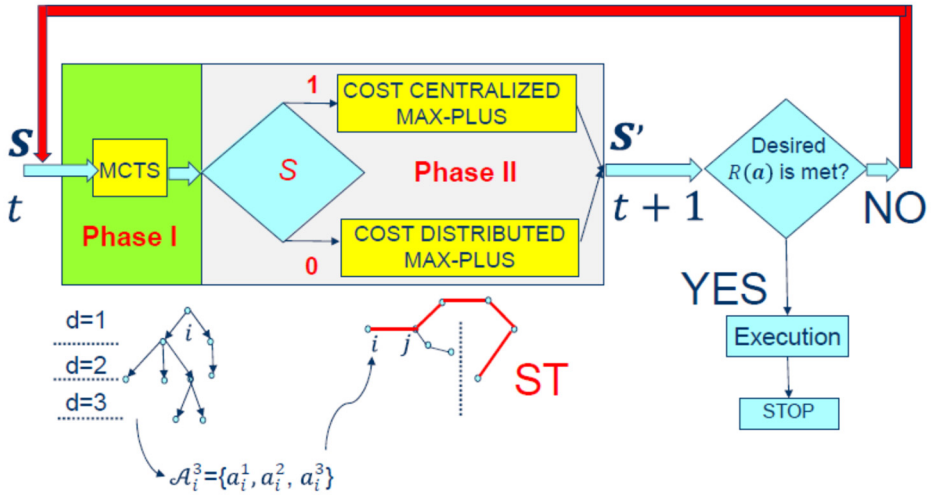


Figure 9. Cost Hybrid Factored Value MCTS Max-Plus Method [24] (reproduced with permission from P. Cotae, N. E. A. Reindorf, M. Kang, and A. Velazquez, "Work-in-Progress: A Hybrid Collaborative Multi Agent Decision Making Algorithm with Factored-Value Max-Plus", 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkiye, 2023, pp. 438–443, doi: 10.1109/BlackSeaCom58138.2023.10299698).

(II) During Phase II, as soon as the segmentation message $\boldsymbol{s}$ occurs, there will be two subgroups created from the original agent group. If $\boldsymbol{s} = 1$ (Line 2 of Section V's algorithm), the Cost Centralized Max-Plus algorithm will then be used by a small group of agents to carry out the decision-making process as shown in Figure 2. The other subgroup, which loses contact with the centralized coordinator ($\boldsymbol{s} = 0$), will employ the Cost-Distributed MP algorithm, as outlined in lines 21 through 49 of the pseudocode, during that same time period. It should be noted that, based on our experiments, there may be a difference in the number of iterations required to run the cost MP algorithm in a decentralized and centralized setting. The results will be gathered at time step $t + 1$ in any case.

During this time step t, the MAS is in the global state $\boldsymbol{s} \in \mathcal{S}$, and in the following time step $t + 1$ the system will undergo a global transition. $\boldsymbol{s}'$. The possibility of transitioning to the following state $\boldsymbol{s}'$ is $p(\boldsymbol{s}', \boldsymbol{s}, \boldsymbol{a}) = 1$ (is certain) if the intended Global Reward $R(\boldsymbol{a})$ is not achieved by the team as in Figure 9 (NO option). If $p(\boldsymbol{s}', \boldsymbol{s}, \boldsymbol{a}) = 0$, the MAS is going to maintain its current state. $\boldsymbol{s}$ or $\boldsymbol{s}'$ with $p(\boldsymbol{s}', \boldsymbol{s}, \boldsymbol{a}) = 0$ if the intended Global Reward is met (YES option). In the latter scenario, MAS agents execute the best joint action $\boldsymbol{a}^* = (a_1^*, a_2^*, \ldots, a_i^*, \ldots, a_N^*)$ discovered thus far if it is needed by the team, and the algorithm will terminate.

Our approach is capable of surmounting the centralized MCTS solution that was suggested in reference [1]. The final coordinated actions of the team are determined within that system via simulation of the global state $\boldsymbol{n}$. Nevertheless, progress towards the subsequent global state cannot be assured. Our proposed approach is capable of surmounting this challenge.

In order to comprehend the hybrid algorithm's performance characteristics, we only provide the numerical results of the Payoff Value and Convergence of reaching the fixed point in this paper. Any online strategy must consider convergence as a primary concern, and the suggested algorithm's efficiency in achieving the set payoff value is a determining factor. The Cost-Distributed Max-Plus algorithm (Figure 10) and the Cost Centralized MP algorithm (Figure 11) are graphical plots that represent our results. The four primary scenarios of decentralized deterministic, decentralized randomized, centralized deterministic,

and centralized randomized are generated by the combination of the reached Payoff value and the convergence of the algorithms.

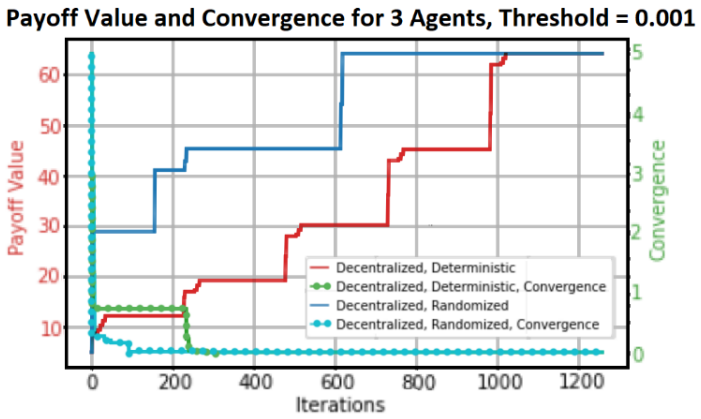


Figure 10. Performance of Cost-Distributed Max-Plus Algorithm [24] (Reproduced with permission from P. Cotae, N. E. A. Reindorf, M. Kang, and A. Velazquez, "Work-in-Progress: A Hybrid Collaborative Multi Agent Decision Making Algorithm with Factored-Value Max-Plus", 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkiye, 2023, pp. 438–443, doi: 10.1109/BlackSeaCom58138.2023.10299698).

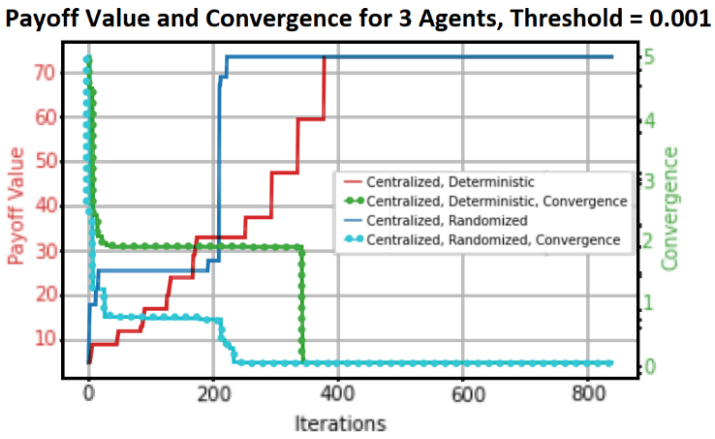


Figure 11. Performance of Cost Centralized Max-Plus Algorithm [24] (Reproduced with permission from P. Cotae, N. E. A. Reindorf, M. Kang, and A. Velazquez, "Work-in-Progress: A Hybrid Collaborative Multi Agent Decision Making Algorithm with Factored-Value Max-Plus", 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkiye, 2023, pp. 438–443, doi: 10.1109/BlackSeaCom58138.2023.10299698).

The blue plots show random actions, and the red plots show deterministic payoff, or performing actions in a deterministic order. In order to compare with sequential decision-making systems, randomized action (uniform distribution) is taken into consideration. For every plot, the red scale on the left represents the payoff value range, and the green scale on the right represents the convergence scale. The splitting segmentation process in a three-node MAS configured in both distributed and centralized algorithms depends on the actual case at time t. This is explained in the above section.

We examine the CG scenario with six agents, akin to Figure 3. We will split into two groups if a segmentation message s is received by either of them (red arrow). In this scenario, there will be no communication between the two groups of six agents. The results are plotted in Figure 10 for the first tree agents, $Q-1.$, $Q-2.$, $Q-3.$, which will be configured in a Spanning Tree configuration (distributed or decentralized). The payoff value and the convergency performance for the second group (centralized in CG settings) of agents, $Q-4.$, $Q-5.$, and $Q-6.$, will be plotted, respectively, as shown in Figure 11. We maintained the same action costs.

Every plot has a convergence threshold and four graphical structures. The maximum payout for the centralized algorithm is reached after approximately 374 iterations for a convergence threshold of 0.001, whereas the maximum payout for the decentralized algorithm is reached after approximately 1023 iterations. But the centralized algorithm reaches convergence after roughly 344 iterations, and the distributed algorithm does so after about 215 iterations. Take note of the various maximum payoff amounts as well. In the decentralized case, the maximum payoff value is approximately 67, whereas in the centralized scenario, it is approximately 74.

The payoff value is greater in the centralized scenario because the centralized controller has "global view" information of all the agents in the MAS and can therefore control the Global Reward (GR) or payoff. Local agents in the scenario involving distributed control have a "partial view" of the environment in which they function. It has been observed that the distributed algorithm converges more quickly than the centralized algorithm for the same number of agents; nevertheless, it requires a longer duration to reach GR.

Within a centralized environment managed by a central coordinator, the agents are prohibited from sharing any information. A completely decentralized coordination strategy for MAS might enable correlated and localized decisions for every agent capable of communication. When local rewards are considered, the decentralized approach is unable to attain the maximum network-wide reward that the centralized case is capable of accomplishing.

As a result, we can conclude that (a) the centralized algorithm reaches a maximum payoff before the distributed algorithm, (b) the distributed algorithm converges more quickly than the centralized algorithm, (c) the centralized algorithm reaches a maximum payoff earlier than the distributed algorithm, and (d) the centralized algorithm reaches a higher maximum payoff than the distributed algorithm during the convergence process.

8. Conclusions and Future Work

The focus of our research was on real-time multi-agent decision making, with a particular focus on the incorporation of a cost factor. To tackle this challenge, we employed the Coordination Graphs and Spanning Tree setups. For the first time, the Cost Hybrid Max-Plus algorithm was presented. The proposed technique, called Cost Hybrid Factored Value MCTS Max-Plus, has properties that make it suitable for practical uses, like cybersecurity [11]. These characteristics include online functionality, anytime capability, distribution over multiple agents, and scalability in terms of the number of agents and local interactions.

Future work on enhancing the Global Reward with the MAS cost based on the agent interactions' locality offers numerous avenues.

Using the most recent developments in Deep Reinforcement Learning, which have shown the enormous potential of neural networks for function approximation in handling a large state space, is one potential future direction.

"Regret techniques" from Game Theory serve as inspiration for another path. One or more agents may feel "regret" for their prior actions when the team selects the global optimal course of action. Maximizing the Global Reward is the same as minimizing the counterfactual regret. The essential concept is that the data changing at a specific node is qualitatively represented by the information state.

The information statistical approach is an additional approach to maximize the information from a specific state of MAS.

Author Contributions: Conceptualization and methodology, P.C.; Software and validation N.-E.A.-R.; Supervision P.C.; writing N.-E.A.-R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by ARLIS- Pilot Projects for the Intelligence & Security University Research Enterprise (INSURE) Academic Consortia Award No. #95111-Z9634203 "Machine Learning Experimentation" Period: 21 August 2021–22 July 2022.

Data Availability Statement: The data is provide upon request. It can be found the PhD Thesis of the first author.

Acknowledgments: We thank the Naval Research Laboratory, Washington, DC, for providing funding for this study under contract N00173-16-D-2009/D05 and for Myong Kang and Alexander Velasquez for their previous comments on this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Choudhury, S.; Gupta, J.K.; Morales, P.; Kochenderfer, M.J. Scalable Anytime Planning for Multi-Agent MDPs. *arXiv* **2021**, arXiv:2101.04788.
2. Amato, C.; Oliehoek, F. Scalable planning and learning for multi agent POMDPs. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29.
3. Cotae, P.; Kang, M.; Velazquez, A. A Scalable Real-Time Multiagent Decision Making Algorithm with Cost. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021; pp. 1–6.
4. Cotae, P.; Kang, M.; Velazquez, A. A Scalable Real-Time Distributed Multiagent Decision Making Algorithm with Cost. In Proceedings of the IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2022; pp. 745–746. [CrossRef]
5. Kok, J.R.; Vlassis, N. Collaborative multi agent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.* **2006**, *7*, 1789–1828.
6. Kok, J.R.; Vlassis, N. Using the max-plus algorithm for multi agent decision making in coordination graphs. In *Robot Soccer World Cup*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 1–12.
7. Vlassis, N.; Elhorst, R.; Kok, J.R. Anytime algorithms for multi agent decision making using coordination graphs. In Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583), The Hague, The Netherlands, 10–13 October 2004; IEEE: Piscataway, NJ, USA, 2004; Volume 1, pp. 953–957.
8. Best, G.; Cliff, O.M.; Patten, T.; Mettu, R.R.; Fitch, R. Dec-MCTS: Decentralized planning for multi-robot active perception. *Int. J. Robot. Res.* **2019**, *38*, 316–337. [CrossRef]
9. de Nijs, F.; Walraven, E.; De Weerdt, M.; Spaan, M. Constrained multi agent Markov decision processes: A taxonomy of problems and algorithms. *J. Artif. Intell. Res.* **2021**, *70*, 955–1001. [CrossRef]
10. Gupta, J.K. Modularity and Coordination for Planning and Reinforcement Learning. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 2020.
11. Patra, S.; Velazquez, A.; Kang, M.; Nau, D. Using online planning and acting to recover from cyberattacks on software-defined networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 15377–15384.
12. Guestrin, C.; Lagoudakis, M.; Parr, R. Coordinated reinforcement learning. *ICML* **2002**, *2*, 227–234.
13. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840. [CrossRef]
14. Revach, G.; Greshler, N.; Shimkin, N. Planning for Cooperative Multiple Agents with Sparse Interaction Constraints. In Proceedings of the 6th Workshop on Distributed and Multi-Agent Planning (DMAP) at ICAPS 202, Haifa, Israel, 2020. Available online: https://icaps20subpages.icaps-conference.org/wp-content/uploads/2020/11/The-online-Proceedings-of-the-6th-Workshop-on-Distributed-and-Multi-Agent-Planning-DMAP-at-ICAPS-2020.pdf (accessed on 15 November 2023).
15. Pettie, S.; Ramachandran, V. An optimal minimum spanning tree algorithm. *J. ACM* **2002**, *49*, 16–34. [CrossRef]
16. Czech, J. Distributed Methods for Reinforcement Learning Survey. In *Reinforcement Learning Algorithms: Analysis and Applications*; Springer: Cham, Switzerland, 2021; pp. 151–161.
17. Li, R.; Patra, S.; Nau, D.S. Decentralized Refinement Planning and Acting. In Proceedings of the International Conference on Automated Planning and Scheduling, Guangzhou, China, 2–13 August 2021; Volume 31, pp. 225–233.
18. Hayes, C.F.; Reymond, M.; Roijers, D.M.; Howley, E.; Mannion, P. Risk Aware and Multi-Objective Decision Making with Distributional Monte Carlo Tree Search. *arXiv* **2021**, arXiv:2102.00966.
19. Rossi, F.; Bandyopadhyay, S.; Wolf, M.T.; Pavone, M. Multi-Agent Algorithms for Collective Behavior: A structural and application-focused atlas. *arXiv* **2021**, arXiv:2103.11067.
20. Grover, D.; Christos, D. Adaptive Belief Discretization for POMDP Planning. *arXiv* **2021**, arXiv:2104.07276.
21. Guestrin, C.; Koller, D.; Parr, R. Multi agent Planning with Factored MDPs. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 1523–1530.
22. Landgren, P.; Srivastava, V.; Leonard, N.E. Distributed cooperative decision making in multi-agent multi-armed bandits. *Automatica* **2021**, *125*, 109445. [CrossRef]
23. Mahajan, A.; Samvelyan, M.; Mao, L.; Makoviychuk, V.; Garg, A.; Kossaifi, J.; Whiteson, S.; Zhu, Y.; Anandkumar, A. Reinforcement Learning in Factored Action Spaces using Tensor Decompositions. *arXiv* **2021**, arXiv:2110.14538.
24. Cotae, P.; Reindorf, N.E.A.; Kang, M.; Velazquez, A. Work-in-Progress: A Hybrid Collaborative Multi Agent Decision Making Algorithm with Factored-Value Max-Plus. In Proceedings of the 2023 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Istanbul, Turkey, 4–7 July 2023; pp. 438–443. [CrossRef]