Annales Mathematicae et Informaticae
61 (2025) pp. 118–128
DOI: 10.33039/ami.2025.10.008
URL: https://ami.uni-eszterhazy.hu

Solving the Team Coordination on Graphs With Risky Edges problem for nonzero self-loop weights*

András Izsó, István Harmati

Budapest University of Technology and Economics
Department of Control Engineering and Information Technology
izso@iit.bme.hu
harmati@iit.bme.hu

Abstract. Multiagent system control is a well-researched area of recent years, since the cooperation of multiple agents opens up the possibility to tackle more complicated problems and create finer scaling systems. Team Coordination on Graphs with Risky Edges has been recently proposed and provides a framework to model such systems. In this problem, multiple agents traverse through a graph. Apart from the ordinary nodes and edges, the graph also contains support nodes, where an agent can choose to support another agent that is moving through a so-called risky edge, associated with the support node. Some solutions have already been proposed; however, all of them assume zero cost of waiting, which is restrictive in many real-world problems. In this paper, we generalize the problem, allowing non-zero cost of waiting, make a solution proposal, and present our comprehensive simulation results.

Keywords: multi-agent, control, cooperation, team, graph

AMS Subject Classification: 93A16 Multi-agent systems, 91A12 Cooperative games

1. Introduction

Multiagent system control is an increasingly popular research area nowadays. The reduction of price and size of hardware opened up the possibility of using multiple,

*This work was supported by the NSF grant DMS 2054735.
The project supported by the Doctoral Excellence Fellowship Programme (DCEP) is funded by the National Research Development and Innovation Fund of the Ministry of Culture and Innovation and the Budapest University of Technology and Economics.

Accepted: October 8, 2025
Published online: October 28, 2025

in some sense simpler agents instead of a single, complex one. This leads to better scalability and cost-effectiveness. However, the complexity is not avoided, only moved to a new level, which raises the need for algorithms that are capable of defining the behavior of multiple agents in order to reach the theoretical optimum in practice.

One of the core problems of such systems in robotics is the Multiagent Path Finding [9]. Its main application areas include public transport [1], package delivery [8] and general formation and swarm control [3]. Previously, most of these approaches looked at agents merely as obstacles that might make their own decisions, but mainly they just have to be avoided. The line of research to which we would like to contribute changes this by the introduction of coordination/cooperation, which promotes interaction between agents.

The Team Coordination on Graphs with Risky Edges (TCGRE) problem provides a framework to model multiagent scenarios, where the action of an agent can reduce the cost of the action of another agent. The agents operate on a weighted graph, representing points of physical or state space, and each agent's task is to traverse from their start nodes to their respective goal nodes, inducing the lowest possible cost. Some nodes of the graph, called *support nodes* are associated with certain edges, called *risky edges*. If an agent in one of the support nodes chooses, instead of moving, to support one of the associated risky edges of their current node, the other agent, that is traveling through that edge in the same time step can do so for a reduced cost.

An example graph is given in Figure 1. There are two agents, traveling respectively from S1 to G1 and S2 to G2. Support nodes are C1, C2, and C3; the risky edges are shown in red and the association between them as green dashed arrows. The cooperation enables the agents to diverge from their individual shortest paths (marked as orange and yellow) to achieve lower cost through cooperation (blue and green).

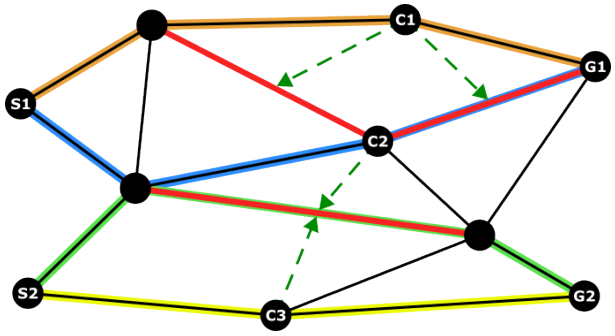


Figure 1. Example of a graph, containing support nodes (C1, C2, C3) and risky edges (red). While two agents are traveling from S1 and S2 to G1 and G2, they might stray from their individual shortest paths (orange and yellow), to cooperate and achieve lower total cost (blue and green).

This approach opens up the possibility to develop such robotics solutions, where the agents are required to operate in a coordinated manner to increase efficiency. There are, however, crucial limitations of the current TCGRE framework that limit its practical applicability.

In our opinion, one of the most crucial discrepancies is the fact that the current formulation cannot assign cost to delay. This shortcoming is evident in the search and rescue area, but even delivery systems can benefit if it is possible to incorporate such a metric in the model. Our goal with this paper is to close this gap. We present an extended formulation of the TCGRE problem that allows the use of non-zero cost self-loops, which can model the cost of staying still or delay. Previous methodologies all took advantage of the costless delay, thus we propose a novel extension to our previously published algorithm [4], and provide proof that it is able to handle the extended problem.

First, we provide a brief overview of the recent research results around the TCGRE problem in Section 2. Then we move on to present our extension proposal in Section 3, including a formal description and a solution algorithm in Section 4. We close our paper by presenting the simulation results in Section 5 and summarizing the completed work in Section 6.

2. Related work

The TCGRE problem has been introduced recently by Limbu et al. [6], along with a solution outline for a two-agent scenario. Later, Zhou et al. have carried out a mathematical analysis, proving the NP-hardness of the problem [10], thus showing that a polynomial time algorithm is not to be expected. However, multiple solution proposals have already been made, with various tradeoffs between runtime and optimality. In this chapter, we provide a brief overview of them.

In addition to the NP-hardness proof, three algorithms were introduced in [10].

The *Joint state graph* (JSG) approach decomposes the problem into a coordination assignment and route finding part. First, a joint state graph is built, where each node represents a configuration in the original problem. During this, the optimal coordination assignment for the movement between any possible neighbor configurations can be calculated by an integer linear program. After the construction of the JSG, the optimal path can be found by any single-agent path finding method.

The *Coordination exhaustive Search* (CES) method limits the maximum number of occurrences for a coordination pair in the solution. This makes an exhaustive search possible over the robot pairs, coordination pairs, and their ordering, resulting in an optimal solution for the limited case. In our previous work, we were able to improve the runtime of this search by precomputing paths between coordination pairs [5].

The final algorithm of [10] was the *Receeding Horizon Optimistic Coordination - A** (RHOC-A*). This solution builds on top of the JSG, but it is constructed only up to a limited horizon. This way, the planning is optimal up until the horizon, after

which the agents plan their path individually, with the assumption that support will be available on each risky edge.

To scale up the solution for more agents, Limbu et al. also introduced a reinforcement learning model [7]. They created a general representation that is able to take any graph as input, with any support structure, but the cost is an explosion of model size. They were able to achieve around 80% of the improvement gained by the optimal solution relative to the agents working independently.

In an effort to compete with the reinforcement learning approach, we developed a method [4], based on the Ant Colony Optimization (ACO) metaheuristic [2]. We assigned pheromone values to each possible next action (robot pair, coordination point assignment) for each relevant actual and goal location pair. Here relevant locations include initial, goal and coordination nodes of the graph, since between these each agent follows individually their optimal path. With this representation we tuned the pheromone values based on the $\mathcal{MIN} - \mathcal{MAX}$ ACO. Our approach showed similar results as [7], but with significantly less resource usage.

3. Extension proposal

3.1. Limitation of current model

The TCGRE problem is a new and unique construction that enhances coordination in multiagent systems. However, the underlying assumptions limit its power to model practical scenarios.

The most critical one is that self-loops in the graph are always considered with $c_{i,i} = 0$ costs. This essentially means there is no cost to delay, which is often not the case in practice. Few such examples, which are often brought up as applications of multiagent systems, are search and rescue operations, deliveries, and transportation.

This, and the fact that collisions between agents are not considered, are utilized in each previous solution approach, as this way the problem reduces to selecting robot pairs, assigning coordination pairs to them, and choosing the order of execution. It doesn't have to be considered if multiple agents use the same edge or node simultaneously, or if one of the cooperating robots has to wait for their mate. Because of this, a new approach or the significant modification of previous ones is required to solve the non-zero self-loop TCGRE problem.

3.2. Problem formulation

In this section we give the formal description of the problem to solve. It is similar to the one proposed in [4], except the self-loops have a constant, non-zero cost. We give the full formal description, to ease comprehension.

Let $G = (\mathbb{V}, \mathbb{E})$ be the graph, on which N homogeneous agents move where $v_i \in \mathbb{V}$ are the nodes, $e_{i,j} = (v_i, v_j) \in \mathbb{E}$, $(v_i, v_i) \in \mathbb{E}\ \forall v_i \in \mathbb{V}$ are the edges and $\exists c_{i,j} \in \mathbb{R}^+ \forall e_{i,j} \in \mathbb{E}, \forall \mathbf{e_{i,i}} = \mathbf{c_{delay}}$ traverse cost. Furthermore let $\mathbb{E}' \subset \mathbb{E}$, $e_{i,i} \notin \mathbb{E}'$

the set of risky edges, $\tilde{c}_{i,j} \in \mathbb{R}^+ \quad \forall e_{i,j} \in \mathbb{E}'$ the reduced cost, while support is available, $\mathbb{S} \subset \mathbb{V}$ the support nodes and $\mathbb{SP} \subset \mathbb{S} \times \mathbb{E}'$ the association between support nodes and risky edges, called support pairs. The location of agent n in timestep t is denoted as $l_{n,t}$.

The goal is to find a series of actions that leads each agent from their respective $v_{0,n}$ initial node to their $v_{g,n}$ goal node. These series consist of $a \in (\{s_n \mid q \leq n \leq n\} \cup \{m_{i,j} \mid e_{i,j} \in \mathbb{E}\})$, where s_n denotes supporting the nth agent and $m_{i,j}$ the movement from v_i to v_j, and has to minimize

$$C_{total} = \sum_{n=1}^{N} \sum_{t=1}^{T} C(a_{n,t}) \tag{3.1}$$

such that

$$C(a_{n,t}) = \begin{cases} 0 & \text{if } a_{n,t} = s_m \vee l_{n,t} = v_{g,n} \\ \tilde{c}_{i,j} & \text{if } a_{n,t} = m_{i,j} \wedge e_{i,j} \in \mathbb{E}' \wedge \exists a_{m,t} = s_n \\ c_{i,j} & \text{otherwise} \end{cases} \tag{3.2}$$

$$l_{n,1} = v_{0,n} \qquad \forall n \in \{1, 2, \ldots, N\} \tag{3.3}$$

$$l_{n,T} = v_{g,n} \qquad \forall n \in \{1, 2, \ldots, N\} \tag{3.4}$$

$$a_{n,t} = \begin{cases} m_{i,i} & \text{if } l_{n,t} = v_i = v_{g,n} \\ m_{i,j} & \Leftrightarrow l_{n,t} = v_i \wedge e_{i,j} \in \mathbb{E} \\ s_m & \Leftrightarrow (l_{n,t}, (l_{m,t}, l_{m,t+1})) \in \mathbb{SP} \\ & \nexists a_{o,t} = s_m, n \neq o \end{cases} \quad \begin{array}{l} \forall n, m, o \in \{1, 2, \ldots, N\}, \\ \forall t \in \{1, 2, \ldots, T\} \end{array} \tag{3.5}$$

$$l_{n,t+1} = \begin{cases} l_{n,t} & \text{if } a_{n,t} = s_m \\ v_j & \text{if } a_{n,t} = m_{i,j} \end{cases} \quad \begin{array}{l} \forall n, m \in \{1, 2, \ldots, N\}, \\ \forall t \in \{1, 2, \ldots, T-1\} \end{array} \tag{3.6}$$

The cost function (3.1) states that the goal is to minimize the total cost, accumulated over each agent and timestep, by selecting the right actions. The cost of each action is given by (3.2), being 0 if the agent reached its goal or the action is to support. This is possible since ours is a global planner, so the cost of support can be moved to the reduced cost of traverse. If the agent is able to and does receive support, its movement cost is the reduced travel cost; otherwise, the original.

Equations (3.3)–(3.6) summarize the constraints on selecting the actions. (3.3) and (3.4) restrict the start and final locations of the agent, where $v_{0,n}$ are given initial, and $v_{g,n}$ are given goal locations for each agent. The possible actions are given by (3.5). If the agent has reached its goal location, it must stay there and cannot take other actions. When this does not apply, the action can select a movement over an edge from its current location, or support another agent, if that agent is moving through one of the risky edges, associated with the current location and is not yet supported. Lastly, the agent's location is updated according to (3.6). If a support action is taken, the agent stays put; otherwise, its location is updated according to the selected movement.

4. Solution proposal

Now that the task is formally described, we would like to present our solution proposal. The outline of the algorithm is given in Algorithm 1.

This is a direct extension of the one we published earlier in [4] by including c_{delay} in the cost calculation step. This is achieved by the introduction of the function $L(\cdot)$, that maps the number of edges in a path of a graph to the path itself. This way, the additional cost can be calculated as

$$c_{waiting} = c_{delay} \cdot |L(p_1) - L(p_2)|,$$

where p_1 and p_2 are the paths, given as a sequence of edges.

The input of the algorithm, using the notation presented in Section 3.2, are the graph on which the agents move (G), the number of agents (N), the set of risky edges ($\mathbb{E}'$), the support pairs ($\mathbb{SP}$) and the respective initial and goal nodes ($v_{0,i}$, $v_{g,i}$) for each agent. Additionally, the algorithm can be adjusted by the following parameters: number of ants (n_{ants}), selection probability coefficients (α, β) as in (4.1), pheromone extreme values (τ_{min}, τ_{max}) and the evaporation rate of the pheromones (ρ).

The algorithm operates by storing the state of the agents and stochastically selecting the next action for a single or a pair of agents jointly. We differentiate between two kinds of actions, the *support solution component (SSC)* where an agent pair is selected to cooperate next and the *goal solution component (GSC)* where a single agent is selected that goes to its goal directly, without further cooperation. The pheromones for these actions are stored in $\mathbf{T}^s$ and T^g respectively and are initialized to 1 in lines 1-2. The best solution is set to an empty list and the cost to infinity in line 3. To take advantage of the findings in [5], the shortest paths between nodes that are relevant in action decision are precalculated in line 4.

The main loop constructs n_{ants} number of independent action lists in each cycle. First, a possible next action is selected with probability

$$p_{SC} = \frac{\tau_{SC}^{\alpha} \eta_{SC}^{\beta}}{\sum_{f \in \mathcal{F}} \tau_f^{\alpha} \eta_f^{\beta}}, \tag{4.1}$$

where $\tau_{SC} \in \mathbf{T}^s \cup T^g$ is the pheromone associated with the solution component SC, η is a heuristic value, inversely proportional to the cost of the solution component, and $\mathcal{F}$ is the set of feasible solution components based on the state of the agents (line 11).

If the selected solution component, $SolComp$ is a support solution component, there is a selected supporter agent r_s, receiver agent r_r, support node v_s and risky edge $e_r = (v_{from}, v_{to})$ where v_{from} is the end of the edge where r_r would arrive, and v_2 is where it would be after the cooperation (line 13). In this case, the shortest paths from the current location of the agents to their coordination pair can be extracted from SP (lines 14–15) and the cost of the action list is extended with the cost of traverse, coordination and delay (line 16). At last, the state of the agents is updated in line 17.

If the selected solution component is a goal solution component, then only the agent going to the goal is selected. The cost only has to be updated by the traverse cost, and the state update includes deactivating the agent (lines 19–21).

Each independent action list is extended until all agents reach their goals, indicated by having all $False$ values in $\mathbf{b}_{active}$. At the end of a cycle, the best solution and cost are updated if appropriate, as well as the pheromone values, based on them (lines 23–30).

The stopping criteria of the iteration may include a limit on the maximum number of steps or the settling of the best cost value. When this is reached, the best found solution and its cost are returned in line 31.

5. Results

In this section we present proof that our solution proposal is capable of solving the novel extension of the TCGRE problem.

Algorithm 1 has been tested on the graph, presented in Figure 2. The edges of the graph are shown by solid lines, with the costs written over them (including self-loops). For ease of analysis, all original costs are set to $c_{i,j} = 1,\ i \neq j$ and the cost of delay $c_{i,i} = 0.1$. Risky edges are presented in red, and each of them has $\tilde{c}_{i,j} = 0.5$ as reduced costs. The coordination pair assignments are marked by green dashed lines.

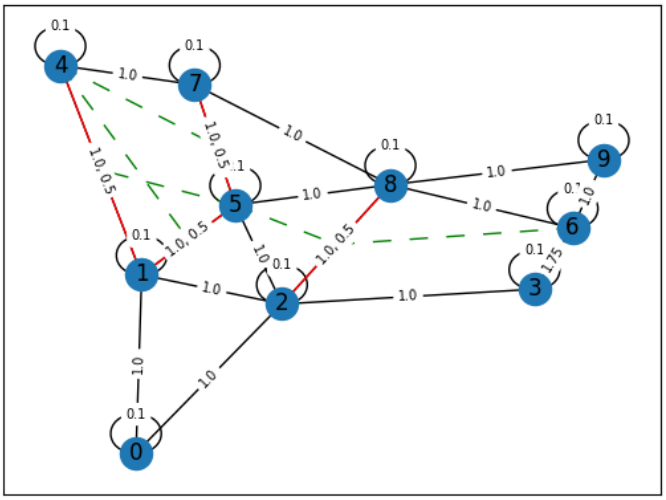


Figure 2. The test graph that we used.

The agents' initial and goal locations are shown in Table 1. They have been selected, so multiple cooperations are required for an efficient solution. Since an optimal algorithm is not yet available, the result has been compared to the indi-

Algorithm 1: TCGRE-ACO for non-zero self-loops

Input : $G = (\mathbb{V}, \mathbb{E})$, N, $\mathbb{E}'$, $\mathbb{SP}$, $v_{0,i}$, $v_{g,i} \forall i \in \{1, 2, \ldots, N\}$
Parameter: n_{ants}, α, β, τ_{max}, τ_{min}, ρ
Output : $Sol_{best-so-far}$, $Cost_{best-so-far}$

$\mathbf{T}^s \leftarrow \mathbf{1} \in \mathbb{R}^{(2|\mathbb{SP}|+|\mathbb{V}_0|)\times(2|\mathbb{SP}|+|\mathbb{V}_0|)\times|\mathbb{V}_0|\times|\mathbb{V}_g|\times|\mathbb{V}_g|}$
$\mathrm{T}^g \leftarrow \mathbf{1} \in \mathbb{R}^{(2|\mathbb{SP}|+|\mathbb{V}_0|)\times|\mathbb{V}_0|}$
$Sol_{best-so-far} \leftarrow \{\}, Cost_{best-so-far} \leftarrow \infty$
$SP \leftarrow ShortestPathFor(G, \mathbb{V}_0 \cup \mathbb{V}_g \cup \mathbb{S} \cup \{v_i, v_j | \forall e_{i,j} \in \mathbb{E}'\})$
while *stopping criteria is not reached* **do**
  $SolCandidates \leftarrow \{\{\} \times n_{ants}\}$
  $CostCandidates \leftarrow \mathbf{0} \in \mathbb{R}^{n_{ants}}$
  for $i_{ant} = 1$ *to* n_{ants} **do**
    $\mathbf{v}_{agents} \leftarrow \begin{bmatrix} v_{0,1} & v_{0,2} & \ldots & v_{0,N} \end{bmatrix}, \mathbf{b}_{active} \leftarrow \{True\}^N$
    while $any(\mathbf{b}_{active})$ **do**
      $SolComp \leftarrow SelectWeighted(G, \mathbf{v}_{agents}, \mathbf{b}_{active}, \mathbf{T}^s, \mathrm{T}_g)$
      if *SolComp is Suppoort Solution Component* **then**
        $(r_s, r_r, v_s, e_r = (v_{from}, v_{to})) \leftarrow SolComp$
        $p_1 \leftarrow SP$ from $\mathbf{v}_{agents}[r_s]$ to v_s
        $p_2 \leftarrow SP$ from $\mathbf{v}_{agents}[r_r]$ to v_{from}
        $CostCandidates[n_{ants}] \leftarrow CostCandidates[n_{ants}]$ + Cost of p_1+ Cost of p_2 + $\tilde{c}_r$ of $e_r + c_{delay} \cdot |L(p_1) - L(p_2)|$
        $\mathbf{v}_{agents}[r_s] \leftarrow v_s, \mathbf{v}_{agents}[r_r] \leftarrow v_{to}$
      else
        $r \leftarrow SolComp$
        $CostCandidates[n_{ants}] \leftarrow CostCandidates[n_{ants}] + SP$ from $\mathbf{v}_{agents}[r]$ to $v_{g,r}$
        $\mathbf{v}_{agents}[SC.r] \leftarrow v_{g,r}, \mathbf{b}_{active}[SC.r] \leftarrow False$
      $append(SolCandidates[i_{ant}], SC)$
  $Cost_{best-now} \leftarrow \min(SolCandidates, CostCandidates)$
  $Sol_{best-now} \leftarrow \arg\min(SolCandidates, CostCandidates)$
  if $Cost_{best-now} < Cost_{best-so-far}$ **then**
    $Cost_{best-so-far} \leftarrow Cost_{best-now}$
    $Sol_{best-so-far} \leftarrow Sol_{best-now}$
  for $\tau \in T^s \cup T^g$ **do**
    if $\tau \in Sol_{best-so-far}$ **then**
      $\tau \leftarrow \left[(1-\rho) \cdot \tau + \frac{1}{Cost_{best-so-far}} \right]_{\tau_{min}}^{\tau_{max}}$
return $Sol_{best-so-far}$, $Cost_{best-so-far}$

vidual shortest paths of the agents in order to observe the improvement achieved through coordinated behavior.

Table 1. Initial and goal nodes of each agent during the tests.

n	Initial node	Goal node
1	0	7
2	0	8
3	0	9
4	4	3

The algorithm has been run on cases $N \in \{2, 3, 4\}$, the obtained results are shown in Table 2 and Figure 3. Parameters $n_{ants} = 100$, $\tau_{max} = 5$, $\tau_{min} = 0$, $\rho = 10^{-6}$, $\alpha = \beta = 1$ have been used through a maximum of 1000 iterations. Our results show that for $N = 2$, the method is capable of finding the coordination improved path every time. Resulting paths are shown in Figure 3a (Agent 1: orange, Agent 2: purple). However, for $N = 3$, the algorithm is much less successful, achieving only the same value as the non-cooperative case. Increasing the number of ants to $n_{ants} = 1000$ yielded somewhat better results, but the algorithm wasn't able to take full advantage of the cooperation, utilizing a coordination pair only once, while the 3rd agent moves directly to the goal as seen in Figure 3b (Agent 3: olive). In case of 4 agents, we weren't able to summon cooperation by change of parameters. This is also visible on Figure 3c (Agent 4: cyan).

Table 2. Results of the repeated evaluation of Algorithm 1. Cooperationless movement steps are merged together for brevity.

N	Actions	$C_{\emptyset coop}$	C_{coop}
2	$(m_{0,1}, m_{0,2}), (m_{1,5}, m_{2,2}), (s_2, m_{2,8}), (m_{5,7}, \varnothing)$	5	4.6
3	$(m_{0,1}, m_{0,2}, m_{0,9}), (m_{1,5}, m_{2,2}, \varnothing), (s_2, m_{2,8}, \varnothing), (m_{5,7}, \varnothing, \varnothing)$	8	7.6
4	$(m_{0,7}, m_{0,8}, m_{0,9}, m_{4,3})$	11	11

A possible reason for the decline of the performance, while increasing the number of agents, is due to the low selection probability of supporting components compared to the goal components. This might be addressed with a non-linear probability density function w.r.t. cost decrease, or adaptive α and β selection. However, these results show that our proposal is applicable to the TCGRE problem with non-zero self-loops. Further refinement of the algorithm and parameter recommendations will be the main goal of our future research work.

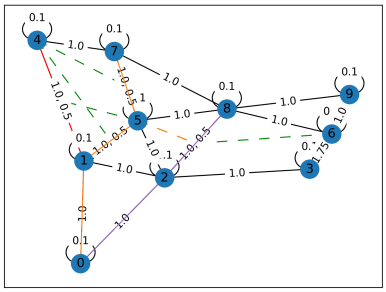
6. Conclusion

In this paper, we have presented a novel extension of the TCGRE problem, which touches on important practical details. We have also proposed a solution algo-

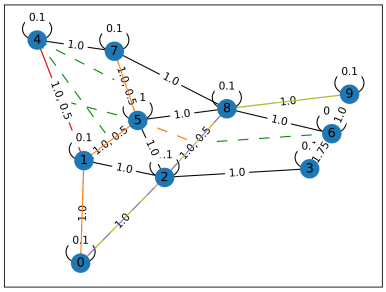
rithm, developed from our previous ACO-based solution [4], further showing the importance of this approach.

The capability of the algorithm has been proven by simulation results. We can state that the method is able to find an action list that reduces the cost of traversal, compared to the individual solution in the extended case. The number of agents that is effectively handled is, however, relatively low. We attribute this to the greediness of the base algorithm.

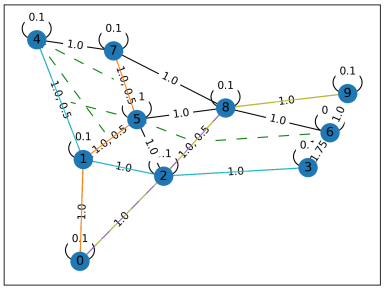
Based on this, we conclude that this extension of the TCGRE problem can be handled by the algorithm, and focus our future work on improving the base algorithm.



(a) Only Agent 1 and 2 are present. Appropriate solution is found.



(b) Agent 1,2 and 3 are present. Agent 3 doesn't take part in cooperation.



(c) All agents are present. Non-cooperative solution is returned.

Figure 3. Resulting paths, marked on the graph. Agent 1: orange, Agent 2: purple, Agent 3: olive, Agent 4: cyan.

References

[1] J. L. Adler, V. J. Blue: *A cooperative multi-agent transportation management and route guidance system*, in: Transportation Research Part C: Emerging Technologies, vol. 10, 5-6, 2002, pp. 433–454.

[2] M. Dorigo, T. Stützle: *Ant Colony Optimization: Overview and recent Advances*, in: Handbook of Metaheuristics, ed. by M. Gendreau, J.-Y. Potvin, Springer, 2019, chap. 10, pp. 311–352.

[3] Y. Hu, M. Chen, W. Saad, H. V. Poor, S. Cui: *Distributed multiagent meta learning for trajectory design in wireless drone networks*, in: IEEE Journal on Selected Areas in Communications, vol. 39, 10, 2021, pp. 3177–3192.

[4] A. Izsó, I. Harmati: *Solving Team Coordination on Graphs with Risky Edges with Ant Colony Optimization*, Acta Polytechnica Hungarica (2025), (submitted).

[5] A. Izsó, I. Harmati: *Improved search algorithm for team coordination on graph*, in: Proceedings of the Workshop on the Advances of Information Technology, ed. by L. K. Bálint; Szirmay-Kalos, 2025, pp. 103–109.

[6] M. Limbu, Z. Hu, S. Oughourli, X. Wang, X. Xiao, D. Shishika: *Team Coordination on Graphs with State-Dependent Edge Costs*, in: IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 679–684, isbn: 9781665491907, doi: 10.1109/IROS55552.2023.10341820.

[7] M. Limbu, Z. Hu, X. Wang, D. Shishika, X. Xiao: *Scaling Team Coordination on Graphs with Reinforcement Learning*, in: Proceedings - IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 16538–16544, isbn: 9798350384574, doi: 10.1109/ICRA57147.2024.10610619, url: https://cs.gmu.edu/~xiao/papers/team_coordination_rl.pdf.

[8] M. Liu, H. Ma, J. Li, S. Koenig: *Task and path planning for multiagent pickup and delivery*, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2019.

[9] P. Surynek: *An optimization variant of multi-robot path planning is intractable*, in: Proceedings of the AAAI conference on artificial intelligence, vol. 24, 1, 2010, pp. 1261–1263.

[10] Y. Zhou, M. Limbu, G. J. Stein, X. Wang, D. Shishika, X. Xiao: *Team Coordination on Graphs: Problem, Analysis, and Algorithms*, in: IEEE International Conference on Intelligent Robots and Systems, Institute of Electrical and Electronics Engineers Inc., Oct. 2024, pp. 5748–5755, isbn: 9798350377705, doi: 10.1109/IROS58592.2024.10802095, url: https://cs.gmu.edu/~xiao/papers/tcgre.pdf.