Solving Team Coordination on Graphs with Risky Edges with Ant Colony Optimization

András Izsó and István Harmati

Budapest University of Technology and Economics, Department of Control Engineering and Information Technology, 2. Magyar tudósok Blvd., H-1117 Budapest, Hungary,
izso@iit.bme.hu, harmati@iit.bme.hu

Abstract: The Team Coordination on Graphs with Risky Edges (TCGRE) is a recently proposed problem by Limbu et al. in 2023. It is a framework built to promote coordination between agents, who are traversing a graph with risky edges in order to minimize the total cost of traverse. Both hard computing and reinforcement learning approaches have been used to try to solve this problem efficiently. We are introducing our soft computing method, utilizing the Ant Colony Optimization Metaheuristic which aims to reduce the runtime and memory requirements of the previously mentioned methodologies by taking advantage of the fact that to solve the TCGRE problem, effectively means to select the coordination pairs and their order. In this paper we present our approach, compare the results with previously published ones through rigorous testing and pinpoint further research directions.

Keywords: multiagent, optimization, ant colony, coordination, cooperation

1 Introduction

Multiagent path finding (MAPF) [1] is an emerging topic of robotics in recent years [2]. More use cases are defined where robots interact with each other and cooperate to efficiently solve a task at hand. Examples of this include but are not limited to delivery [3], warehouse and storage management [4] and public transport [5]. Furthermore, as much research revolves around soft computing optimization algorithms (SCO) [6], how they can be successfully taught [7] and their possible applications, including investments [8], risk assessment [9], and control systems [10] the combination of the two fields hold great potential.

In this study we focus on a problem that combines path planning on a graph with cooperation, by allowing agents to reduce the cost of certain edges through coordination. We review previous solution proposals, present our novel approach and point out the advantages over earlier methods.

The recently proposed Team Coordination on Graphs with Risky Edges (TCGRE) problem [11] introduces a framework to model cooperative multiagent planning tasks. In this setting, multiple agents have the task to traverse on a graph between their respective start and goal nodes. To represent the possibility of pairwise cooperation between agents, so-called *coordination pairs* are introduced. They consist of a node (*support node*) and an associated edge (*risky edge*) from the graph. An agent in the support node can choose to give support to an agent traveling through the risky edge. In this case, the moving agent can do so with a reduced cost.

Solutions proposals for the problem have already been made [12]. These guarantee optimal solutions with varying constraints. However, they do not scale well with increasing the number of agents and coordination pairs. In an effort to achieve better runtime for larger problems, reinforcement learning based methods have also been examined [13], with some extent of improvement.

This paper introduces a novel solution proposal of the TCGRE problem, which achieves the effectiveness of the reinforcement learning based approach, with lower complexity thus decreasing the time and memory required to solve the problem. The new approach uses Ant Colony Optimization (ACO) [14] to take advantage of the fact that a solution can be built step-by-step, and the evaluation of each possible component can be done based on the location and goal of the coordinating agents.

In Section 2 the recent work around the TCGRE problem and the ACO technique are shortly described. Following that, the TCGRE problem is reformulated, in a way that suits our approach better in Section 3. The proposed algorithm is presented in Section 4, and the achieved simulation results is discussed in Section 5. Finally, the paper is concluded by summarizing our work and pointing out future goals.

2 Related work

Since the introduction of the TCGRE problem in [11], multiple solution approach has been proposed. In this section these are briefly discussed, and the contribution of this paper is highlighted relative to them. Thereafter, the ACO metaheuristic is also presented, as it is the main background of the contribution.

2.1 Previous solution proposals of TCGRE

2.1.1 Joint State Graph

The Joint State Graph (JSG) method presented in [12] tackles the TCGRE problem by expanding the graph on which the agents are moving. In this much larger graph (the JSG), each node encodes a possible configuration of agent positions on the

original graph. An edge is present between two nodes of the JSG if each agent must traverse only one edge in order to move between the respective configurations.

This way the TCGRE problem can be decomposed into two subproblems. First, the optimal coordination assignments are computed for a single edge of the JSG, through an Integer Linear Program (ILP) [15]. Then any classical single agent path planning algorithm (e.g.: Dijkstra's [16]) is capable of providing the optimal solution.

Although the single agent path planning has well established solutions, the ILP is an NP-hard problem, causing long evaluation times. However, the exponential memory consumption is an even more significant drawback of this approach. Since a single node in the JSG encodes a possible configuration of agents on the original graph, the JSG will have $O(|V|^N)$ size, where N is the number of agents and $|V|$ is the number of nodes in the original graph. In order to mitigate this problem, the *Critical* Joint State Graph has been introduced in [11]. Each agent can travel between the start nodes, coordination pairs and goal nodes disregarding the rest of the agents. This way the path between such nodes can be replaced by single edges and all the others can be omitted, reducing the decision space of the optimization.

The method, presented in this paper sacrifices the exact optimum to reduce execution time, compared to the JSG approach.

2.1.2 Coordination Exhaustive Search

To find balance between optimality and runtime, the Coordination Exhaustive Search (CES) algorithm has also been introduced in [12] as a solution proposal with a restriction over the TCGRE problem. This approach assumes that each coordination pair is present in the optimal solution for a given, fixed number of times, thus opening the possibility of exhaustive search in this (constrained) solution space. The resulting algorithm's computational complexity increases exponentially with the number of possible coordination pairs.

With our previous work [17], we were able to enhance this to some extent, which made the calculation feasible for a few more agents and coordination pairs, but the applicability is still limited. The ACO based method cannot compete with CES in terms of computational costs, but it is much less restrictive, providing a different trade-off between optimality and runtime.

2.1.3 Reinforcement learning approach

Two reinforcement learning (RL) approaches are presented in [13]. One uses reinforcement learning as an optimization framework, to solve a single TCGRE problem, the other is to learn a single policy that can be later evaluated on any TCGRE problem instance. Here, the problem is converted into a Markov Decision Process formulation, with a state space of size $O(|V|^6 N)$ where N is the number of

agents and $|V|$ is the number of nodes in the original graph. The RL algorithms were able to handle around 3-4 agents and 15 nodes in the graph, with an optimality above 70% of the JSG. Although our novel approach aims to compete with such methods, direct comparison is not possible with these algorithms as we don't have the means to reproduce the results and exact data of evaluation time is not provided by the authors. However, the achieved effectiveness is of the same level, and the sheer size of the optimization space used in the RL approaches strongly suggests that their computational cost is much higher than the one presented in the current paper.

2.2 Ant Colony Optimization

Ant Colony Optimization is a group of heuristic approaches, widely used on combinatorial optimization problems [14]. ACO is applicable to such problems, where solutions can be constructed gradually from components. During execution, multiple solutions are built in parallel by virtual ants. Each solution component has a τ_i pheromone value assigned to it, indicating the desirability of that component. In each construction step, the next component is chosen stochastically, usually according to (1), where η is a heuristic value assigned to component i and α, β are parameters to adjust the importance of the pheromones and heuristic values.

$$p_i = \frac{\tau_i^\alpha \eta_i^\beta}{\sum_j \tau_j^\alpha \eta_j^\beta} \tag{1}$$

After a complete solution is constructed, its cost is calculated and the pheromones assigned to the components that are present in the solution are updated. Many update strategies exist, e.g.: to update all components, only update the components of the best solution of the current iteration or the best solution so far, etc. This is repeated until a maximum number of steps, there is no improvement on the objective function. The main steps of the procedure are shown in Algorithm 1.

Algorithm 1: ACO metaheuristic

```
1  Set parameters, initialize pheromone values
2  while termination criteria not met do
3      ConstructAntSolutions
4      ApplyLocalSearch (optional)
5      UpdatePheromones
6  end
7  return Best solution
```

3 Problem formulation

After the brief overview of related work, the problem is formally described in this section. For the sake of completeness, first, the definition of the TCGRE problem is given, based on [12]. Then, the formal optimization problem is presented in a novel form that suits our approach better.

Let a *risky graph* be defined as a $G = (V, E)$ graph, where $V = \{v_1, v_2, \dots, v_{|V|}\}$ is the set of nodes, $E = \{e_{i,j} = (v_i, v_j) | v_i, v_j \in V\}$ is the set of edges. Furthermore, the *cost of traversing* each edge is given by $e_{i,j} \mapsto c_{i,j} \in \mathbb{R}^+, c_{i,i} = 0$ where $e_{i,j} \in E$. A set of *risky edges* is selected from G as $E' \subset E$, and a *reduced traverse cost* is associated with them: $e_{i,j} \mapsto \tilde{c}_{i,j} \in \mathbb{R}^+ \; \forall e_{i,j} \in E'$. A set of *support nodes* is also chosen as $S \subset V$. The assignment between the support nodes and risky edges are given by $SP \subset S \times E'$ and are called *support pairs*.

Note that the cost of giving support can also be defined, however it can be incorporated in the reduced travel cost of risky edges according to [12].

In the *TCGRE problem*, a set of N homogeneous agents travel through a risky graph, given as $G_r = (V, E, c_{i,j}, SP, \tilde{c}_{i,j})$. Their start and goal nodes are respectively defined as $V_0 = \{v_{0,1}, v_{0,2}, \dots, v_{0,N}\}, V_g = \{v_{g,1}, v_{g,2}, \dots, v_{g,N}\} \subset V$. The agents can take in each timestep either a *support action* or a *movement action* $a \in \{s_1, s_2, \dots, s_N\} \cup \{m_{i,j} | \forall e_{i,j} \in E\}$. A movement action $m_{i,j}$ simply results in the agent moving from v_i to v_j at $c_{i,j}$ cost if it is not supported and at $\tilde{c}_{i,j}$ cost if $e_{i,j} \in E'$ and the moving agent is supported by another one. During a support action s_i, an agent provides support to agent i, thus it can traverse a risky edge with reduced cost. The goal is to find a course of actions for each agent, which drives it from its start to its goal node, while minimizing the total traverse cost. A constrained optimization problem can be formulated that solves this task. It is given by (2)-(7).

The objective function to minimize is given by (2), which is the total cost over all agents and timesteps. The cost calculation (3) states that the support is costless (moved to the traverse cost of the risky edge). In case of a movement action, reduced traverse cost can be used if the traversed edge is risky and there is an agent supporting the traverse. Otherwise, the regular movement cost has to be included. As for the constraints (4) ensures that only valid movements can be performed, (5) restricts movements to be on continuous paths, (6) sets the start and end of the path as the given start and goal nodes, and (7) allows only one agent to support another one at a time step. This formulation does not restrict senseless support, but since both supporting and staying in place have 0 cost, it is indifferent.

$$\min_{\substack{a_{n,t} \\ 1\le n\le N \\ 1\le t\le T}} \sum_{n=1}^{N}\sum_{t=1}^{T} C(a_{n,t}) \tag{2}$$

$$C(a_{n,t}) = \begin{cases} 0 & \text{if } a_{n,t} = s_m \\ \tilde{c}_{i,j} & \text{if } a_{n,t} = m_{i,j} \wedge e_{i,j} \in E' \wedge \exists a_{m,t} = s_n \\ c_{i,j} & \text{otherwise} \end{cases} \tag{3}$$

w.r.t.

$$a_{n,t} = m_{i,j} \Leftrightarrow e_{i,j} \in E \quad \forall n \in \{1,2,\dots,N\}, \forall t \in \{1,2,\dots,T\} \tag{4}$$

$$a_{n,t} = m_{i,j} \Leftrightarrow a_{n,t+1} = m_{j,k} \quad \forall n \in \{1,2,\dots,N\}, \forall t \in \{1,2,\dots,T-1\} \tag{5}$$

$$\begin{aligned} a_{n,1} = m_{i,j} \Leftrightarrow v_i = v_{0,n}, \\ a_{n,T} = m_{i,j} \Leftrightarrow v_j = v_{g,n}, \end{aligned} \quad \forall n \in \{1,2,\dots,N\} \tag{6}$$

$$a_{n,t} = s_m \Leftrightarrow \nexists a_{o,t} = s_m, n \neq o \quad \forall n,m,o \in \{1,2,\dots,N\} \tag{7}$$

4 Proposed algorithm

Now that the problem is thoroughly defined, our solution proposal is described in this section. In order to utilize the ACO metaheuristic, a representation has to be found, which allows the solution to be built gradually, component after component.

As the CES approach has already pointed out, the TCGRE problem can be thought of similarly as a Sequential Ordering Problem (SOP). In the SOP, the order of some elements determines the cost, which should be minimized by rearranging them. In TCGRE the agents move individually between support pairs and waiting has zero cost, therefore, a solution can be built as an ordering of support pairs with agent associations. As ACO has already been effectively applied on the SOP problem [18], applying it to TCGRE is a well-founded objective. However, one important difference is that in case of TCGRE it is not required to incorporate every component, unlike the SOP. A direct conversion to the SOP problem representation could be performed like the case of the JSG, but it would not solve the problem of exponential memory consumption. In our approach we chose a different way to model the problem, allowing us to take advantage of symmetries, resulting in significantly reduced memory usage.

To be able to apply ACO, first the solution component and pheromone representation must be declared. Furthermore, a heuristic approximation of the utility of a solution component should be defined. Then, the component selection and pheromone update method should be selected. These steps will be detailed in the rest of this section. The optional local search step of the metaheuristic has not yet been incorporated.

4.1 Solution component

As mentioned earlier, the agents move independently between the coordination pairs, so optimal paths between them can be computed individually. Because of this, the optimization problem's decision space can be greatly reduced if the solution components for the ACO algorithm are defined as more complex manoeuvres, rather than individual actions. This way two major types of behaviour have been recognized. An agent either collaborates with one (or later more) agent or individually moves to its goal node. That is why the following two types of *solution component* (SC) were defined:

- A *support solution component* (SSC) consists of a *supporting* robot, *receiving* robot and a *coordination point*, represented as the $(r_s, r_r, cp) \in N \times N \times SP$ triplet, and means that robot r_s is going to provide support to robot r_r at the coordination point cp
- A *goal solution component* (GSC) consists of a single robot r that will travel directly to its goal node

A solution for the TCGRE problem can be represented as a sequence of such components, where a goal solution component must be the last for each robot, after which that robot cannot be used in other solution components. This way, the multiagent behaviour can be formed by selecting one solution component at a time.

4.2 Pheromone representation

The pheromones encode 'how desirable' it is to select a solution component, which can be decided based on the cost of the solution component. However, the cost is not a constant value, but dependent on the current state of the robots, which is given by their location on the G_r graph:

$$\boldsymbol{v}_{agents} \in V^N \tag{8}$$

Now the cost of a goal solution component can be defined as the cost to traverse to the robot's goal node, as given by

$$C_{GSC}(r) = Path(v_r, v_{g,r}) \tag{9}$$

where r defines the robot being considered, v_r is the location of the robot stored in $\boldsymbol{v}_{agents}$ and $v_{g,r}$ is the goal of r. The $Path(v_1, v_2)$ function is considered to calculate the cost of the path with the lowest cost between v_1 and v_2 on G_r. Since we defined the cost to be positive, Dijkstra's algorithm [16] can be used for this purpose.

In case of a support solution component, the cost must include both agents' traversal cost to the coordination pair's respective nodes and the cost of the coordination (cost of traversing the risky edge

$$C_{SSC}(r_s, r_r, cp) = Path(v_{r_s}, v_{cp,s}) + Path(v_{r_r}, v_{cp,r}) + \tilde{c}_{cp} \tag{10}$$

where r_s is the supporting robot, r_r is the receiving robot, v_{r_s} and v_{r_r} are their respective locations from $\boldsymbol{v}_{agents}$, cp is the coordination point with $v_{cp,s}$ as support node and $v_{cp,r}$ is the endpoint of the risky edge that is reachable by r_r with the lower cost. $\tilde{c}_{cp}$ is the cost of traverse over the risky edge.

Since cost fundamentally affects the desirability of a choice, the pheromone representation should depend on everything the cost does. Furthermore, a coordination pair might have different utility if the agent is going to different goals. Based on these observations, we define the pheromone structure to consist of two parts: one for SSC and one for GSC as

$$\mathbf{T}^g \in \mathbb{R}^{(2|CP|+|V_0|)\times|V_g|}$$
$$\mathbf{T}^s \in \mathbb{R}^{(2|CP|+|V_0|)\times(2|CP|+|V_0|)\times|CP|\times|V_g|\times|V_g|} \tag{11}$$

$\mathbf{T}^g$ contains the pheromones for goal components. The value depends on which goal the agent is going to ($|V_g|$ number) and the location of the agent ($2|CP| + |V_0|$ number since only start points and coordination pairs have to be considered). The pheromone values for the possible support actions are stored in $\mathbf{T}^s$ which for each coordination pair ($|CP|$ number) depends on both the current and goal locations of both agents. The following function is defined as a unified notation of a pheromone value associated with a given solution component SC, depending on the current state of the agents given by $\boldsymbol{v}_{agents}$

$$\tau_{SC}(\boldsymbol{v}_{agents}) = \begin{cases} \tau^s_{v_{agents}[r_s], \boldsymbol{v}_{agents}[r_r], cp, v_{g,r_s}, v_{g,r_r}} & \text{if } SC = (r_s, r_r, cp) \\ \tau^g_{v_{agents}[r], r} & \text{if } SC = r \end{cases} \tag{12}$$

where τ marks the elements of **T** with the indices defined by the subscript.

4.3 Heuristic values

In addition to the pheromone values, which meant to represent the utility of a solution component based on experience, heuristic values should be defined which give an initial approximation of the desirability of each solution component. Two of such values have been examined.

The first one considers only the cost that is directly added by selecting a solution component, i.e. cost-to-come

$$\eta_{SC}^{c2c}(\boldsymbol{v}_{ag.}) = \begin{cases} \dfrac{1}{C_{SSC}(r_s, r_r, cp, \boldsymbol{v}_{ag.})} & \text{if } SC \text{ is } SSC \\ \dfrac{1}{C_{GSC}(r)} & \text{if } SC \text{ is } GSC \end{cases} \tag{13}$$

The second one incorporates a simple predictive element, also adding the cost of going directly to the agent's goal after the coordination, i.e. the cost-to-go

$$\eta_{SC}^{c2g}(\boldsymbol{v}_{ag.}) = \begin{cases} \dfrac{1}{C_{SSC}(r_s, r_r, cp, \boldsymbol{v}_{ag.}) + C_{GSC}(r_s) + C_{GSC}(r_r)} & \text{if } SC \text{ is } SSC \\ \dfrac{1}{C_{GSC}(r)} & \text{if } SC \text{ is } GSC \end{cases} \tag{14}$$

4.4 Solution component selection

While building the solution, each component is selected probabilistically based on the pheromone and heuristic values, as given by (1). However, not all solution components are feasible in each iteration because some agents might be already in their goal locations. Let

$$\boldsymbol{b}_{active} \in \{True, False\}^N \tag{15}$$

store for each agent whether it had reached its goal or is still active. Based on this the set of feasible solution components can be defined as

$$\mathcal{F}(\boldsymbol{b}_{act.}) = \{SSC \mid \boldsymbol{b}_{act.}[r_s] \wedge \boldsymbol{b}_{act.}[r_r] = True\} \cup \{GSC \mid \boldsymbol{b}_{act.}[r] = True\} \tag{16}$$

which is the union of support solution components where both participating agents are still active and goal solution components for every active agent. Using this notation, the general equation of (1) can be written as

$$p_{SC}(\boldsymbol{v}_{ag.}, \boldsymbol{b}_{act.}) = \begin{cases} \dfrac{\tau_{SC}^{\alpha}(\boldsymbol{v}_{ag.}) \cdot \eta_{SC}^{\beta}(\boldsymbol{v}_{ag.})}{\sum_{SC' \in \mathcal{F}(\boldsymbol{b}_{act.})} \tau_{SC'}^{\alpha}(\boldsymbol{v}_{ag.}) \cdot \eta_{SC'}^{\beta}(\boldsymbol{v}_{ag.})} & \text{if } SC \in \mathcal{F}(\boldsymbol{b}_{act.}) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

for this case. Where α and β are parameters, setting the weight of the pheromones (τ_{SC}) and heuristic values (η_{SC}).

4.5 Pheromone update

As described earlier, the pheromone values are updated after each complete solution. There are multiple approaches in the literature, however, based on our study, the $\mathcal{M}ax\text{-}\mathcal{M}in$ Ant System ($\mathcal{MM}$AS) [19] seemed the most appealing. This algorithm updates only the pheromones of the solution components that are contained in the best solution. The pheromones' value is also restricted to a lower and upper bound.

$$\tau \leftarrow [(1-\rho) \cdot \tau + \Delta\tau^{best}]_{\tau_{min}}^{\tau_{max}}, \qquad \forall \tau \in \mathbf{T}^s, \mathbf{T}^g \tag{18}$$

τ_{min}, τ_{max} are respectively the lower and upper bound of the pheromone values, ρ is the evaporation rate of the pheromones and the operator $[x]_b^a$ defined as

$$[x]_b^a = \begin{cases} a & \text{if } x > a \\ b & \text{if } x < b \\ x & \text{otherwise} \end{cases} \tag{19}$$

and

$$\Delta\tau^{best} = \begin{cases} \dfrac{1}{C_{best}} & \text{if } it\ is\ the\ best\ solution\ so\ far \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

4.6 Algorithm outline

To tie it altogether, the pseudocode of the resulting algorithm can be found in Algorithm 2.

The algorithm takes a risky graph G_r, the start nodes V_0, the goal nodes V_g as inputs and provides the solution to the TCGRE problem in $Sol_{best-so-far}$ as an array of solution components, and its cost $Cost_{best-so-far}$ on its output.

First, the variables that store the pheromones and the result are initialized in line 1-3. Then, in line 4 every shortest path between the start nodes, goal nodes and nodes that are part of a coordination pair are calculated to provide caching. The main loop starts by initializing the solution candidates in line 6-7 and runs until a stopping criteria is reached. The stopping criteria can be a maximum number of iterations, as well as a condition for when the solution does not change significantly.

In each iteration a solution is built by every ant. The selection of the next solution component is performed in line 11, based on (17). Then, through lines 12-14 the solution candidate is extended, the states of the agents are updated, and the cost is increased accordingly. After each ant completed its solution candidate, the best of them is selected, and compared to the current best-so-far solution, which is updated if needed in lines 17-20. At the end of each iteration, the pheromone values are updated based on (18) in line 21. At last, the best-found solution and its cost is returned in line 23.

5 Results

Algorithm 2 has been implemented in Python[1], taking advantage of NetworkX [20] and NumPy [21] libraries. The JSG algorithm has also been implemented to serve as a comparison baseline using SciPy [22] ILP tools. Our CES implementation [17] was also included in the comparison. JSG has no parameters, CES assumed that each coordination pair is present at max once in the solution and the parameters of the ACO based solution is presented in Table 1. The cost-to-come cost function (13) did not yield sufficient results so the cost-to-go version (14) was used during testing.

Table 1. Parameters of the ACO based algorithm

n_{ants}	α	β	τ_{max}	τ_{min}	ρ
100	1	1	5	0	10^{-6}

The algorithm was tested on a wide range of risky graphs. To compare against different graph sizes, node numbers of $|V| \in \{10, 15, 20, 25, 30\}$ has been used. The variation of graph sparseness has been characterized by two different methods. In the first case the number of edges has been set according to $|E| = c|V|, \forall c \in \{2, 3, 4\}$, resulting in sparse graphs. The generation of dense graphs has been

[1] The source code is available at https://github.com/izsoandras/tcgre-aco

Algorithm 2: ACO to solve the TCGRE problem

Input: $G_r = (V, E, c_{i,j}, SP, \tilde{c}_{i,j}), V_0 = \{v_{s,1}, v_{s,2}, \dots, v_{s,N}\}$, $V_g = \{v_{g,1}, v_{g,2}, \dots, v_{g,N}\}$ k

Parameters: $n_{ants}, \alpha, \beta, \tau_{max}, \tau_{min}, \rho$

Output: $Sol_{best-so-far}, Cost_{best-so-far}$

$\mathbf{T}^s \leftarrow \mathbf{1} \in \mathbb{R}^{(2|CP|+|V_0|)\times(2|CP|+|V_0|)\times|CP|\times|V_g|\times|V_g|}$

$\mathbf{T}^g \leftarrow \mathbf{1} \in \mathbb{R}^{(2|CP|+|V_0|)\times|V_g|}$

$Sol_{best-so-far} \leftarrow \{\ \}, Cost_{best-so-far} \leftarrow \infty$

Calculate all shortest path between $V_0 \cup V_g \cup S \cup \{v_i, v_j | \forall e_{i,j} \in E'\}$

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**

    $\boldsymbol{v}_{agents} \leftarrow V_0$, $\boldsymbol{b}_{active} \leftarrow \boldsymbol{True} \in \{True, False\}^N$

    while $any(\boldsymbol{b}_{active})$ **do**

      $SC \leftarrow$ Select solution component (17)

      Add SC to $SolCandidates[i_{ant}]$

      Update $\boldsymbol{v}_{agents}$ and $\boldsymbol{b}_{active}$ according to SC

      $CostCandidates \leftarrow CostCandidates + C(SC)$ (9), (10)

    end

  end

  $Sol_{best-now}, Cost_{best-now} = \min(SolCandidates, CostCandidates)$

  if $Cost_{best-now} < Cost_{best-so-far}$ **then**

    $Cost_{best-so-far} \leftarrow Cost_{best-now}, Sol_{best-so-far} \leftarrow Sol_{best-now}$

  endif

  for $\tau \in \mathbf{T}^s \cup \mathbf{T}^g$ **do** Update τ (18)

end

return $Sol_{best-so-far}, Cost_{best-so-far}$

achieved by setting the number of edges relative to the complete graph, namely $|E| = c\frac{|V|(|V|-1)}{2}, \forall c \in \{0.75, 0.8, 0.99\}$. The number of robots has been varied for $N \in \{3, 4, 5, 6\}$ and the number of coordination pairs for $|CP| \in \{0, 1, 2, 3, 4\}$. The JSG, CES and ACO algorithms have been evaluated for each combination of parameters 5 times yielding 3000 distinct evaluations for 600 different input size. If an algorithm did not conclude after 1000s, its execution has been stopped to enforce feasible runtime.

The graphs have been generated randomly, according to the Erdős-Rényi random graph model [23]. Coordination pairs, start and goal nodes have been selected randomly from the graph. In order to ensure that each problem is in fact solvable, the connectivity of the graph has been assured by resampling.

The most prominent indicator of improvement is the fact that JSG ran out of the 8GB available memory after the $|V| = 25, |E| = 250, |CP| = 4, N = 3$ case, while ACO was able to provide a result for all cases. The runtime for the JSG, CES and ACO algorithms for the cases where JSG were able to run are shown in Figure 1. Dots represent cases where an algorithm did not conclude in the given time limit, which occurred 89 times for JSG and 4 times for ACO out of the 1073 cases where JSG was able to run.

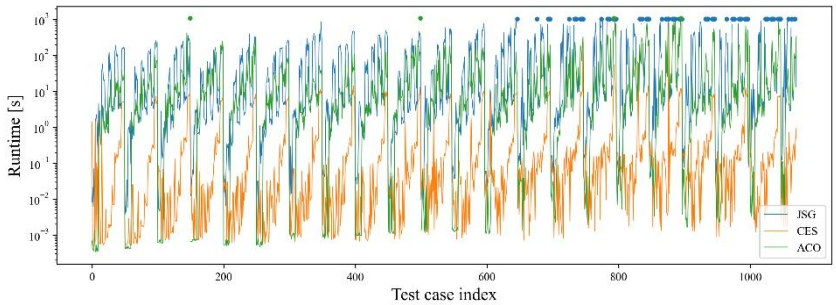

Figure 1

Runtime of the JSG, CES and ACO algorithms, for the cases where JSG was able to run. Dots mark where the algorithm did not conclude in the given time limit. JSG have run out of memory after these cases. The runtime of ACO is visibly lower, often by 1 magnitude, and is still concluding, while JSG runs out of the allowed time.

This already indicates an improvement in terms of runtime, which is further reinforced by analyzing the ratio between ACO and JSG runtimes as $\frac{t_{ACO}}{t_{JSG}}$ in cases where both are concluded. The histogram of this ratio is plotted in Figure 2, separately for less than and greater than 1 case. ACO were able to provide a solution faster than JSG in 684 cases. Furthermore, it is visible on the histograms that the ratio is concentrated below 0.2, which indicates significant improvement.

Since CES is a much simpler but restricted algorithm compared to both JSG and ACO, it is expected to have a lower runtime. However, the fact that its runtime is exponential of the number of coordination pairs, $O\left((2N^2)^{|CP|}\right)$ [17] is already showing and is expected to rise above the much slower increasing runtime of ACO.

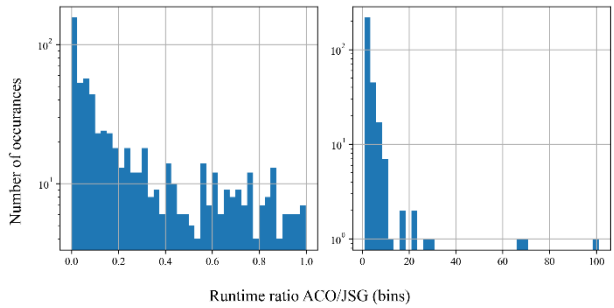

Figure 2

Ratio of ACO and JSG runtimes as $\frac{t_{ACO}}{t_{JSG}}$. In 684 cases ACO provides a solution significantly faster than JSG, and even if it doesn't, apart from a few outliers, its runtime is concentrated near that of JSG.

In terms of cost, the ACO is on par with the RL approach. As in [13], we use the ratio $\frac{c_{JSG}}{c_{ACO}}$ as a measure of optimality. The histogram of this value is presented in Figure 3, with the 70% and 80% margins marked as orange and green, which are also used in [13]. In 68.6% of the cases, ACO was able to match with the optimal solution of JSG. 94% of the solutions lie above 80% optimality margin and 96.5% above the 70%, which shows that this algorithm is a capable alternative to the RL based presented in [13], with lower resource requirements.

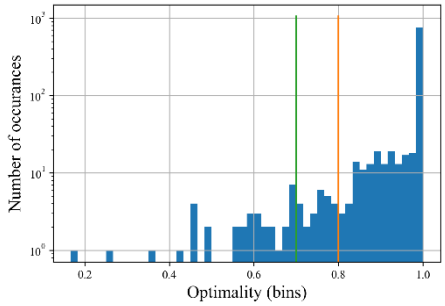

Figure 3

Distribution of the optimality ratio $\frac{c_{JSG}}{c_{ACO}}$ in 68.6% of the cases ACO was able to match the cost with JSG. In 94% the optimality is above 0.8 (orange) and in 96.5% is above 0.7 (green), indicating similar performance as [13]

Conclusions

TCGRE is a recently emerged problem that makes the modelling of certain cooperative multiagent behaviour possible. Previous solution proposals show the lack of ability to scale well with multiple robots or coordination pairs or do it with a significant amount of processing power. In this paper, we have proposed a novel solution to the problem and showed that it is capable of similar performance as the recently proposed alternatives, with significantly less computational needs. In our future work, we are going to further analyse the capabilities of the new approach, by developing a local search method that would potentially increase the performance. Our goals also include the application of similar approaches to generalized versions of the TCGRE problems.

References

[1] Ma, H.: Graph-Based Multi-Robot Path Finding and Planning. Current Robotics Reports, 3 (3), 2022, p. 77–84.

[2] Abujabal, N.A., Rabie, T., Kamel, I.: Path Planning Techniques for Multi-robot Systems: A Systematic Review. In: 2023 15th International Conference on Innovations in Information Technology, IIT 2023. Institute of Electrical and Electronics Engineers Inc., 2023, p. 1–6.

[3] Kou, N., Peng, C., Ma, H., Kumar, T., Koenig, S.: Idle time optimization for target assignment and path finding in sortation centers. In: AAAI conference on artificial intelligence. 2020, p. 9925–9932.

[4] Salvado, J., Krug, R., Mansouri, M., Pecora, F.: Motion planning and goal assignment for robot fleets using trajectory optimization. In: IEEE/RSJ international conference on intelligent robots and systems. 2018, p. 7939–7946.

[5] Precup, R.E., Bojan-Dragos, C.A., Gao, K., Cui, S.: Problem Setting for Trajectory Planning and Cruise Control of a Connected Autonomous Electric Bus in Intersection Scenarios with Human-Driven Vehicles to Optimize Energy, Comfort and Tracking. Romanian Journal of Information Science and Technology, 28 (3), 2025, p. 299–312.

[6] Tompa, T., Kovács, S.: Knowledge Base Optimization of the HFRIQ-Learning. Acta Polytechnica Hungarica, 21 (10), 2024, p. 93–110.

[7] Precup, R.E., Hedrea, E.L., Roman, R.C., Petriu, E.M., Szedlak-Stinean, A.I., Bojan-Dragos, C.A.: Experiment-Based Approach to Teach Optimization Techniques. IEEE Transactions on Education, 64 (2), 2021, p. 88–94.

[8] Zavadskas, E.K., Dinçer, H., Yüksel, S., Eti, S.: Intelligent Expert Systems Using Molecular Fuzzy Genetic Algorithms and Multi-Objective Particle

Swarm Optimization for Circular-Oriented Project Investments. Romanian Journal of Information Science and Technology, 28 (3), 2025, p. 260–273.

[9] Laufer, E., Takács, M., Rudas, I.J.: Optimization of Mamdani-type Fuzzy Risk Assessment Models. Acta Polytechnica Hungarica, 21 (10), 2024, p. 167–183.

[10] Zamfirache, I.A., Precup, R.E., Petriu, E.M.: Q-learning, Policy Iteration and Actor-Critic Reinforcement Learning Combined with Metaheuristic Algorithms in Servo System Control. Facta Universitatis, Series: Mechanical Engineering, 21 (4), 2023, p. 615–630.

[11] Limbu, M., Hu, Z., Oughourli, S., Wang, X., Xiao, X., Shishika, D.: 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, p. 679–684.

[12] Zhou, Y., Limbu, M., Stein, G.J., Wang, X., Shishika, D., Xiao, X.: Team Coordination on Graphs: Problem, Analysis, and Algorithms. In: IEEE International Conference on Intelligent Robots and Systems. Institute of Electrical and Electronics Engineers Inc., 2024, p. 5748–5755.

[13] Limbu, M., Hu, Z., Wang, X., Shishika, D., Xiao, X.: 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, p. 16538–16544.

[14] Dorigo, M., Stützle, T.: Ant Colony Optimization: Overview and Recent Advances. In: Handbook of Metaheuristics (Editors: M. Gendreau and J.-Y. Potvin). Springer Cham, 2019, p. 311–352.

[15] Clautiaux, F., Ljubić, I.: Last fifty years of integer linear programming: A focus on recent practical advances. Elsevier B.V., 2024, 324, p. 707–731.

[16] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. (Heidelb)., 1, 1959, p. 269–271.

[17] Izsó, A., Harmati, I.: Improved search algorithm for team coordination on graph. In: Proceedings of the Workshop on the Advances of Information Technology 2025 (Editors: B. Kiss and L. Szirmay-Kalos). Budapest, 2025, p. 103–109.

[18] Gambardella, L.M., Dorigo, M.: An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem. INFORMS J. Comput., 12 (3), 2000.

[19] Stützle, T., Hoos, H.H.: MAX-MIN Ant System. Future Generation Computer Systems, 16 (8), 2000, p. 889–914.

[20] Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring Network Structure, Dynamics, and Function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy2008) (Editors: G. Varoquaux et al.). Pasadena, 2008, p. 11–15.

[21] Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., et al.: Array programming with NumPy. Nature Research, 2020, 585, p. 357–362.

[22] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., et al.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods, 17 (3), 2020, p. 261–272.

[23] Erdős, P., Rényi, A.: On random graphs I. Publicationes Mathematicae Debrecen, 6, 1959, p. 290–297.