Optimal Routing of Modular Agents on a Graph

Karan Jagdale and Melkior Ornik

***Abstract*— Motivated by an emerging framework of Autonomous Modular Vehicles, we consider the abstract problem of optimally routing two modules, i.e., vehicles that can attach to or detach from each other in motion on a graph. The modules' objective is to reach a preset set of nodes while incurring minimum resource costs. We assume that the resource cost incurred by an agent formed by joining two modules is the same as that of a single module. Such a cost formulation simplistically models the benefits of joining two modules, such as passenger redistribution between the modules, less traffic congestion, and higher fuel efficiency. To find an optimal plan, we propose a heuristic algorithm that uses the notion of graph centrality to determine when and where to join the modules. Additionally, we use the nearest neighbor approach to estimate the cost routing for joined or separated modules. Based on this estimated cost, the algorithm determines the subsequent nodes for both modules. The proposed algorithm is polynomial time: the worst-case number of calculations scale as the eighth power of the number of the total nodes in the graph. To validate its benefits, we simulate the proposed algorithm on a large number of pseudo-random graphs, motivated by real transportation scenario where it performs better than the most relevant benchmark, an adapted nearest neighbor algorithm for two separate agents, more than 85 percent of the time.**

I. INTRODUCTION

Modular systems — systems where agents can attach and detach from others mid-mission — are an emergent generalization of multi-agent systems. Their abstract formulation is motivated by the novel technology of Autonomous Modular Vehicles (AMV), which allows two vehicles to attach and detach from each other [13].

Applications of modular agents are primarily studied for public transportation to lower the operation cost of vehicles and improve the service quality to customers. For example, modular bus systems are proven to show significant benefits over the non-modular bus system [10]. Upon joining, these buses have an open area where the passengers can stand and travel from one bus to another. This ability to distribute the passengers between two buses (which can detach later) decreases the average passenger travel time. Vehicle modularity has been studied in specific transportation scenarios like oversaturated traffic — where the passenger demand is higher than the transportation network capacity [2], flex route transit service [12], shared-use corridors where different bus routes sharing a common bus stop [16] etc. Papers [9], [15] utilize the modularity to vary the vehicle capacity and propose an optimal vehicle dispatch schedule given passenger demand.

K. Jagdale is with the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA. karansj2@illinois.edu

M. Ornik is with the Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA. mornik@illinois.edu

In addition to the public transportation domain, modular vehicles can play a crucial role in improving vehicle platooning — a method of driving a group of vehicles together to increase the road capacity — for enhanced mobility and passenger comfort, as studied by paper [11]. Vehicular modularity has also been considered in military vehicles: by attaching two military vehicles of different types, a unit can obtain increased mobility on a variety of terrains [3].

A. Related work and contributions

Previous efforts in modular agents are primarily focused on the passenger transport system and are developed for specific problems where the agent routes are fixed. To the best of our knowledge, modular vehicles are not studied in an abstract setting. This paper considers the problem of routing modular agents on a graph to reach preset nodes (targets) by traversing the minimum possible distance. In the rest of the discussion, the module refers to the agent that cannot split further, and the agent is any vehicle possibly formed by joining multiple modules. We consider the scenario where the cost of traversing an edge for an agent is the same as that of a single module to simplistically model benefit of joining two modules like fuel efficiency [11], less traffic congestion.

We propose a novel algorithm that routes two modules to the preset targets. While the problem of doing so with minimal cost is NP-hard, we present a heuristic approach which requires polynomial time in the number of targets and graph size. The algorithm aims to determine when and where to join or split the modules by predicting the cost incurred to complete the mission by a single module vs. two modules starting at multiple nodes. Intuitively, the modules are joined if they are relatively close to each other and away from the remaining targets, and the joined agent splits once it is routed enough close to the remaining targets.

II. PROBLEM STATEMENT

This paper considers the problem of optimal planning with modular agents traversing on an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The modules aim to visit a set of preset nodes while incurring the least cost of travel. These preset nodes are referred to as *targets* in the paper, and the set of the targets is given by $\mathcal{T} \subset \mathcal{V}$. The cost incurred by a module on its path is equal to the sum of the weights of the traversed edges, and the agent formed by joining two modules incurs the same cost as the individual modules. Exploiting their modular capabilities, modules can join and split at any node in the graph. Moreover, a module can also choose not to move.

Let us mathematically define the above problem of optimal planning for modular agents. The sequence of nodes tra-

versed by a module defines its *path*. For every edge $e \in \mathcal{E}$, let its weight, i.e., the cost incurred by traversing it, be denoted by $w_e > 0$. Let the set of modules moving at time instant t is given by $\mathcal{K}(t)$ and let the edge traversed by Module i at that time be instant denoted by $e_i(t)$, then the total cost incurred by modules during the mission is given by

$$\sum_{t=1}^{T} \sum_{e \in \cup_{i \in \mathcal{K}(t)} \{e_i(t)\}} w_e \tag{1}$$

where T is the time step after which the modules' mission is complete. In equation 1, we assume that all modules that traverse the same edge at the same time are joined. Given the proposed cost model, there is no benefit in modules not joining when traversing the same edge concurrently. Thus the actions of joining and splitting are automatically encoded by keeping a record of timed paths of individual modules.

Problem 1: Let n modules operate on a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a target set $\mathcal{T} \subset \mathcal{V}$. Denote the path of Module i by $P_i = (v_i(0), ..., v_i(T))$ with $v_i(t) \in \mathcal{V}$ and $(v_i(t), v_i(t+1)) \in \mathcal{E}$ for each $0 \leq t < T$. Determine paths $P_1, ..., P_n$ which minimize 1 such that $\mathcal{T} \subset \cup_{i=1}^{n} \cup_{t=0}^{T} \{v_i(t)\}$

Problem 1 is in general NP-hard, as the case of n=1 reduces to the classical NP-complete problem of finding a minimum Hamiltonian path [5]. As a first step in finding a computationally feasible approximate solution, this paper considers the case of $n = 2$. Such a scenario already exhibits the fundamental characteristics of modularity by allowing the two modules to join into a combined agent and subsequently split. To demonstrate the potential benefits of modularity in such a case, we provide a short example illustrated in Fig. 1. The two modules start from nodes A and B, and need to visit target set $\mathcal{T} = \{E, F, I, J\}$. The graph structure and weights are given on Fig. 1. The optimal paths for two modular agents are given by (A, C, D, E, G, I, J, K) and (B, C, D, F, H, I, J, L). The modules join at node C, split at D, join again at G, and finally split at H. The total cost incurred is 40. If the agents were not modular, i.e., lacked the capability to join and split, by inspection we can verify that the optimal routing would be that one of the modules does not move at all, whereas the other visits all the targets, e.g., $(A, C, D, F, D, E, G, I, H, I, J, K, J, L)$. The cost incurred in this case is 45.

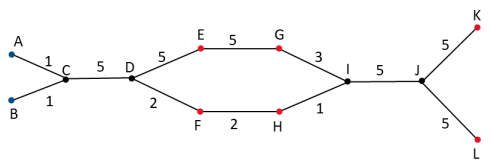

Fig. 1. Example graph with modules' initial locations shown with blue dots and the targets indicated with the red dots. The numbers next to the edges indicate the costs of traversing the edges.

We now move to propose an approximately optimal solution to Problem 1 in the case when $n = 2$.

III. Routing algorithm

As described earlier, the analytical solution to Problem 1 is NP-hard; we opt for heuristic approach. Our solution consists of three interacting elements which are considered anew at every time step:

1) If the modules are separated, deciding if and where to join them: We do that by comparing distance between the modules with the distance of the remaining targets from the modules. Intuitively, we join the modules if they are close to each other and away from the targets.
2) If the modules are joined, deciding if and where to split them: We make this decision by estimating the costs of paths obtained by splitting the joined agent at different nodes, including not splitting at all, and visiting the remaining targets using the adapted nearest neighbor algorithm for two agents. We find the optimal splitting node that corresponds to the minimum cost.
3) Optimal routing of the modules towards the targets: We decide whether to route one of the modules or both of them at the current time step by predicting the cost of doing so using nearest neighbor strategy.

Now, we present a detailed analysis of the above decisions and the tools used within one by one; we start with the first element in our solution: an adapted nearest neighbor algorithm for two concurrently moving agents.

A. Adapted nearest neighbor for two agents

A nearest neighbor approach [14], in which an agent always visits the closest unvisited target, has been widely used to approximate a solution to minimum Hamiltonian path problem for a single agent. We now adapt this approach for two agents, and use it in the formulation of the proposed algorithm and later as a benchmark of optimal policies of *non-modular agents* to compare with the proposed algorithm.

The adapted nearest neighbor algorithm for a target set $\mathcal{T}$ and module nodes a_1 and a_2 is as follows: Let $d(n_1, n_2)$ be the minimal possible cost of a path between nodes n_1, n_2. First, we find $(\tau_1, \tau_2) \in \mathcal{T}^2$ with $\tau_1 \neq \tau_2$ such that $d(a_1, \tau_1) + d(a_2, \tau_2) = \min_{\tau_1' \neq \tau_2'} d(a_1, \tau_1') + d(a_2, \tau_2')$. Then we move Agent 1 and Agent 2 by one node on the shortest path to τ_1 and τ_2 respectively. While we consider $n = 2$ in keeping with the remainder of the paper, this algorithm can be directly generalized for any $n > 1$.

B. Joining decision

Intuitively, joining the modules is beneficial if the modules are relatively close to each other and far from the remaining targets and there are sufficiently many targets remaining to be visited. This intuition is illustrated by Fig. 2. In case (a), joining the two modules is beneficial, and the cost of the optimal path, as indicated in Fig. 2, is 40. If the modules did not join, only one of the modules would visit both targets, or both modules would visit one of the targets, and the cost incurred in these cases would be 45 and 50, respectively. In case (b), joining the modules is not beneficial as only one target is remaining and there is no need for both modules to

move. In case (c), the module locations are not sufficiently away from the target nodes.

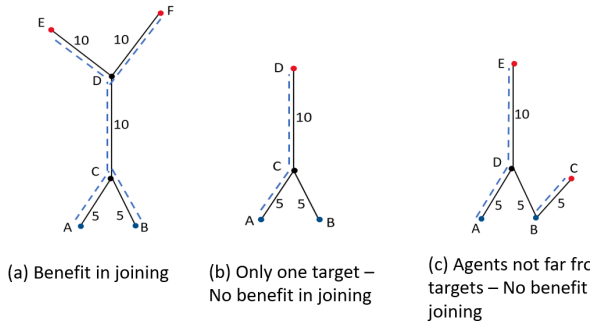

Fig. 2. Different scenarios to explain the reasoning behind the benefits of joining. Blue dots show the modules' initial locations, and the red dots show the target locations. Blue dashed lines show the optimal module path.

To formalize the notion of the modules being far from the remaining targets, we will define a set of *central nodes* $\mathcal{C}$ with respect to the module locations and is given by

$$\mathcal{C}(t) = \arg\min_{v \in \mathcal{V}} \frac{d(v, m_1(t)) + d(v, m_2(t))}{2},$$

where $m_1(t)$ and $m_2(t)$ denotes Module 1 and Module 2 nodes at time t. Let $d_c(t)$ be the distance between central node $c \in \mathcal{C}(t)$ and the closest remaining target, i.e., $d_c(t) = \min_{\tau \in \overline{\tau}(t)} d(c, \tau)$, where $\overline{\tau}(t)$ denotes the set of targets remaining at time t. We formulate the following joining condition: if $d(m_1(t), m_2(t)) < d_c(t)$, then the modules join at node $n_j = \arg\min_{c \in \mathcal{C}} d_c(t)$. In the case n_j is not uniquely defined, we choose any node in the set n_j. The above formulation ensures that we join the modules at the node which is both central with respect to the modules' positions and "in the direction of" the remaining targets. Thus, the modules move closer to the remaining targets while moving towards the joining node.

C. Splitting decision

To formulate a splitting condition, we use a similar intuition to that of joining. Routing the joined agent close to the remaining targets and splitting there will produce a lower cost than splitting first and routing the modules separately. To exemplify this, we refer to Fig. 3. In strategy (a), modules split at node A, and both modules visit the targets. The cost of routing obtained in this case is 40. In strategy (b), the agent splits at node A, and only one of the modules visits the remaining targets. The cost of routing is again 40. However, in strategy (c), the joined agent first moves to node B, i.e., closer to the remaining targets and splits there, and each module visits a target with the total routing cost of 30.

With the above intuition, we employ the following method to determine the optimal splitting node.

For each $r \geq 0$ We calculate the cost of the agent traveling for r nodes on its nearest neighbor path — path obtained using the nearest neighbor approach for the agent — followed by splitting the agent into two modules, with the separated modules proceeding using the adapted nearest

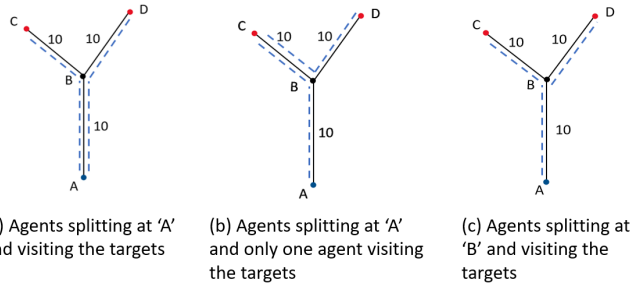

Fig. 3. Different scenarios depicting intuitive reasoning behind the optimal split location. Blue dots show the modules' initial locations, and the red dots show the target locations. Blue dashed lines show the modules'/agent's path.

neighbor algorithm for two agents. We choose the splitting node corresponding to the r, which produces smallest cost.

In a case where the targets are naturally clustered, the above strategy might not produce optimal results because the modules will split while in the last cluster to be visited, and not at any earlier point which is not optimal as demonstrated in Section II. To address such cases, we include an additional clustering filter to the splitting decision. We divide the targets into two clusters using the standard technique of k-means clustering [7]; if the *DB-index* [4] of the clusters is lover than a threshold value, i.e., the clusters are fairly separated, we only consider the targets in the cluster closest to the agent location to obtain optimal splitting node.

While the intuitions for the joining condition and for splitting are the same, the technical approach differs. Using analogy of joining, one can get optimal splitting node as the one which is central with respect to the targets and closest from the agent location. Although, for the case when the number of targets is greater than two, this method will not produce optimal splitting node. Consider a case with four target nodes arranged to look like vertices of a quadrilateral, the most central node for the target nodes will be at the node closest to the centroid of the quadrilateral. Now, routing the agent all the way to the most central node and splitting there in the above case is clearly not optimal. To avoid the possible infinite loop of performing repeated joining and splitting, we do not let the agent join immediately after it is split. Instead, in the next step, we route the agent as given in Section III-D.

The splitting strategy is summarized in **Algorithm** 1. Note that, in the pseudocodes described below, $p(n_1, n_2)$ gives the sequence of nodes corresponding to the shortest path (least cost) between n_1, n_2 starting with n_1 and $d(n_1, n_2)$ gives the shortest distance(least cost) between nodes n_1, n_2. Now we analyze the decision of routing one Vs. both the modules.

D. Target assignment

To understand the significance of this section, consider the scenario in Fig. 1 with targets at only F and J nodes. In this case, it is not optimal to join the modules as there is no benefit in splitting later as the shortest path to the farther target passes through the closer target. The optimal solution would be only one of the modules going to both targets.

To decide the number of modules to be routed, we

Algorithm 1 Node to split the agent

Input: Agent node : a, target set: $\mathcal{T}$
Threshold *DB-index*: D_t
Output: Node to split the agent n_{sp}

Divide $\mathcal{T}$ into two clusters $\mathcal{C}_1, \mathcal{C}_2$ with $\mathcal{C}_1$ closer to agent.
D: *DB-index* of $\mathcal{C}_1, \mathcal{C}_2$
if $D < D_t$ **then**
    $\overline{\tau} = \mathcal{C}_m$
else
    $\overline{\tau} = \mathcal{T}$
end if
P : Agent's nearest neighbor path to visit targets in $\overline{\tau}$
for r=0:length(P)-1 **do**
    $n_r = P(r)$
    Compute $cost_r$, i.e., cost of visiting remaining
    nodes using adapted nearest neighbor with modules
    starting from n_r
    $c_r = d(a, n_r) + cost_r$
end for
$p = \arg\min_r(c_r)$
For any $\overline{p} \in p, n_{sp} = P(\overline{p})$

compare the cost incurred by each module if it visits all the targets using the nearest neighbor approach with the cost of visiting all the targets using both modules with the adapted nearest neighbor for two agents. Based on the computed costs, we move the modules by one node using the strategy that produces the least cost and compute the optimal route. Combining the joining, splitting, and target assignment mechanisms, we obtain the complete routing algorithm, presented in **Algorithm** 2.

IV. Complexity analysis

We will show that **Algorithm** 2 for modular planning both generally outperforms the benchmark of planning for two non-modular agents — provably so for a particular class of graphs — and does it in a computationally feasible manner. We begin by analyzing its computational complexity. We separately investigate the cases of modules being separated and joined and later state the overall algorithm complexity.

Let m, n denote the total number of nodes and the targets, respectively. We use Dijkstra's algorithm to find the shortest path between two nodes, which has the time complexity of $O(m^2)$ [1]. In the nearest neighbor algorithm for a single module, at each step, we find the closest target by computing the shortest distance from the module to n targets using Dijkstra's algorithm with $O(nm^2)$ computations. These computations are performed for n times to visit all the targets. Thus, the nearest neighbor algorithm has the complexity of $O(n^2m^2)$. Similarly, the complexity of the adapted nearest neighbor algorithm is also $O(n^2m^2)$, as it finds the target nearest to the modules 4 times performing $O(nm^2)$ computations, and this process is repeated for $n/2$ times to visit all the targets.

At every time step, if the modules are separated, they are either joined or moved by one node. The complexity of

Algorithm 2 Optimal Routing

Input: Graph with the module nodes : m_1, m_2 and the target nodes: $\mathcal{T}$
Output: Modules' timed paths : P_1, P_2

Initialize $P_1(0) = m_1, P_2(0) = m_2, t = 0$
lastSplit = False
while All targets are not visited **do**
    if $P_1(t) = P_2(t)$ **then**
        Obtain splitting node n_{sp} using **Algorithm** 1
        $m_1 = m_2 = n_{sp}$
        $P_1 = [P_1, p(m_1, n_{sp})], P_2 = [P_2, p(m_2, n_{sp})]$
        $t = length(P_1)$
        lastSplit = True
    else
        if $d(m_1, m_2) \leq d_c$ and lastSplit = False **then**
            $P_1 = [P_1, p(m_1, n_{jn})]$,
            $P_2 = [P_2, p(m_2, n_{jn})]$
            $n_1 = length(P_1), n_2 = lenght(P_2)$
            if $n_1 < n_2$ **then**
                $P_1 = [P_1, n_{sp} \times ones(1, n_2 - n_1)]$
            else
                $P_2 = [P_2, n_{sp} \times ones(1, n_1 - n_2)]$
            end if
            $t = length(P_1)$
        else
            Compute:
            p_1 : Cost of visiting all the targets with
            Module 1 using nearest neighbor algorithm
            p_1 : Cost of visiting all the targets with
            Module 2 using nearest neighbor algorithm
            p_3 : Cost of visiting all the targets with
            both modules using adapted nearest neighbor.
            $i_m = \arg\min_i(p_i)$
            if $i_m = 1$ or $i_m = 2$ **then**
                $\tau^c_{i_m} = \arg\min_{\tau \in \mathcal{T}} d(m_{i_m}, \tau)$
                $P = p(m_{i_m}, \tau^c_{i_m})$
                $m_{i_m} = P(2)$
                $P_1 = [P_1, m_1], P_2 = [P_2, m_2]$
            else
                Obtain module paths N_1, N_2 using
                adapted nearest neighbor.
                $m_1 = N_1(2), m_2 = N_2(2)$
                $P_1 = [P_1, m_1], P_2 = [P_2, m_2]$
            end if
            $t = t + 1$
        end if
        lastSplit = False
    end if
end while

deciding whether to join is $O(nm^2+m^3) = O(m^3)$. Namely, we first compute the central node with respect to the module locations, which requires computing the shortest distance from each node to the module locations with $O(2m \times m^2) = O(m^3)$ calculations. Then, we obtain the closest target from the central node by performing $O(n \times m^2) = O(nm^2)$ calculations. To decide the next nodes to move the modules, we compute the nearest neighbor path individually for both modules and the path using the adapted nearest neighbor. The complexity of these operations is $O(n^2m^2)$. Thus $O(n^2m^2+m^3)$ computations are required to decide the next steps with separated modules.

To determine the splitting node, we first compute the nearest neighbor path of the joined agent with $O(n^2m^2)$ computations. Then, we compute the adapted nearest neighbor path from the r^{th} node in that path as described in Section III-C with the number of operations upper-bounded by $O(n^2m^2)$. The total number of times adapted nearest neighbor path is computed is upper bounded by m. Thus, the computations required to find the optimal splitting node are upper-bounded by $O(m^2n^2 + n^2m^3) = O(n^2m^3)$.

Thus, at each time step, we perform $O(n^2m^3)$ computations if the modules are joined, whereas we perform $O(n^2m^2 + m^3)$ computations if the modules are separated. The number of time steps to visit all the targets is upper-bounded by $O(m^2n)$: the same pair of nodes cannot be visited more than twice without visiting a target in between. Thus the worst case complexity of the proposed algorithm is $m^2n \times O(n^2m^3) = O(n^3m^5)$, i.e., given $n \leq m, O(m^8)$.

V. Performance on Clustered Graphs

As a first step for a future theoretical discussion of our proposed algorithm's performance, we present the class of graphs where the proposed algorithm with modular agents *provably* produces lower cost than the non-modular agents.

We consider a class of graphs where the targets can be categorized into clusters. First, we define the notion of cluster; $\mathcal{C} \subset \mathcal{V}$ is a cluster is it satisfies following condition:

$$\max_{c,c' \in C} d(c,c') < \min_{c \in C, b \notin C} d(c,b).$$

Fig. 4 illustrates the considered class of graphs. The clusters are shown by C_1 and C_2. While this class is obviously simplistic — e.g., it has only two clusters, only two nodes that are not in the clusters, and the distances of the clusters to those nodes is equal — the proof below can be directly extended to graphs with slightly more complicated graphs. Formally defining such graphs would require burdensome notation, so we omit such a discussion. While noting that the same proof intuition continues to hold. On the other hand, extending this proof for *substantially more complicated graphs* is one of the central objectives for future work.

Modules are initially joined and present at **A**. Nodes **A** and **B** are α distance apart. Clusters C_1 and C_2 are at λ distance from node **B**, i.e., $\lambda = \min_{\tau \in C_1} d(B,\tau) = \min_{\tau \in C_2} d(B,\tau)$. Let β_1 and β_2 be the cost of visiting all targets in C_1 and C_2 starting from the node closest to **B** in C_1 and C_2, respectively, using the nearest neighbor approach.

Let us compare the performance of modular agents to a non-modular benchmark. First, consider the case of non-modular agents: Depending on the relationship between α and λ, one of the two following strategies is optimal:

1) Either Agent 1 or Agent 2 alone visits all the targets, and the cost incurred is given by
$$\alpha + 3\lambda + \beta_1 + \beta_2. \tag{2}$$
2) Each agent visits the targets in one of the clusters, and the cost incurred is given by,
$$2\alpha + 2\lambda + \beta_1 + \beta_2. \tag{3}$$

Now, let us consider the proposed algorithm for modular agents. The algorithm will predict the cost of visiting all the targets by splitting at different nodes on its nearest neighbor path i.e., it will compute the cost of splitting at

1) **A**: The cost will be equal to expression 3 as the modules use the adaptive nearest neighbor after splitting.
2) **B**: $\alpha+2\lambda+\beta_1+\beta_2$, i.e, each module visiting the targets in one of the clusters, as the clusters are equidistant from **B** and consequently the adaptive nearest neighbor assigns one target in each cluster to both modules when the agent is at node B.
3) any node in C_1 or C_2: The cost will be roughly equal to $\alpha+3\lambda+\beta_1+\beta_2$, as both modules first visit targets in the clusters they entered and then visit targets in the other cluster. The cost of routing two modules with the adaptive nearest neighbor is slightly different than the cost of routing using the nearest neighbor using a single module in the clusters.

As the predicted cost of splitting at **B** is lowest, the agent will be routed to node **B**, incurring α cost, and split there. Then the cost of visiting all the targets is evaluated with

1) only Module 1 or Module 2, i.e., $3\lambda + \beta_1 + \beta_2$,
2) using both modules with the adaptive nearest neighbor approach, i.e., $2\lambda + \beta_1 + \beta_2$.

Once the modules are inside the clusters, they will be routed using adaptive nearest neighbor, which basically is the nearest neighbor on each module separately as the modules are in different clusters . Thus, the total cost using the proposed algorithm is given by

$$\alpha + 2\lambda + \beta_1 + \beta_2. \tag{4}$$

Comparing 2, 3 with 4 makes it clear that the cost obtained with the proposed algorithm using the modular agents is lesser than the cost obtained with the non-modular agents.

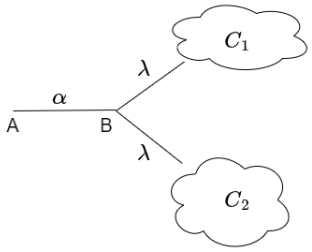

Fig. 4. Figure describing the class of graphs where the proposed algorithm with modular agents produces lower cost than non-modular agents.

VI. Numerical Results

To describe the performance of the proposed algorithm, we consider an illustrative example motivated by real-life scenario in in retail logistics industry and simulation on a batch of randomly generated graphs.

A. Illustrative Example

In retail logistics, companies use warehouses to store manufactured goods before distributing them to shops. Usually, big companies have a network of warehouses spread near big cities. We consider the problem of distributing shipments in a set of Amazon warehouses located in/near Los Angeles, California, USA. We create a graph using the data from google maps [6] from several warehouse locations and use the proposed algorithm to obtain the optimal path using two modular trucks. The trucks start from two warehouses denoted by nodes **n1** and **n2** in Fig. 5 and want to visit another 6 warehouses to deliver the shipments. Solid lines in Fig. 5 shows the truck routes obtained with the proposed policy. The trucks join at **n3** and split at node **n5** and the total cost incurred is 33.9. On the other hand, dashed lines in Fig. 5 shows the optimal routes of the non-modular trucks, i.e., trucks without the capability to join or split. The cost incurred in this case is 37.1.

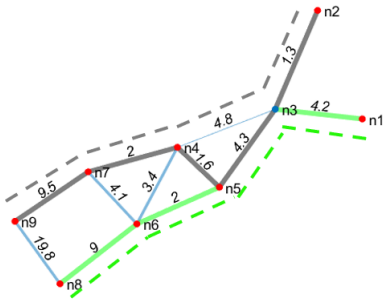

Fig. 5. Graph representing the road network connecting 8 warehouses. The number next to the edges is the distance in miles between the locations represented by the nodes. Green and grey highlights indicate the path of Truck 1 and Truck 2, respectively. Red dots highlight the warehouse locations. Solid and dashed lines indicate the cases when the agents are modular and non-modular, respectively.

B. Batch simulation on random graphs

Moving to show the comparative performance of the proposed algorithm on a large set of graphs, we run 100 simulations on graphs generated in a pseudo-random fashion having a total of 18 nodes and 8 target nodes. These pseudo-random graphs are produced in a way that ensures that target nodes are located in three clusters, with a structure shown in Fig. 6. Such a structure is motivated by the public transportation scenario , where multiple buses partially go through the same area [8]. We use 0.4 as the threshold value of *DB-Index* (D_t) in the simulations. Note that the considered class of graphs is more general than those in Section V, for which we provided a theoretical proof of performance.

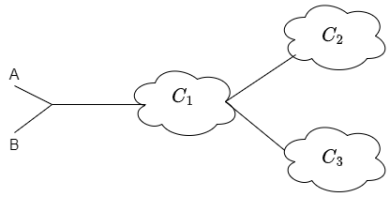

Fig. 6. Figure indicating the of graphs where the target nodes are present in three clusters. The modules' initial locations are marked by A, B.

Intuitively, having a cluster structure ensures that the split and join actions can reduce the total routing cost. The average cost of routing with the proposed algorithm was 92.22 with a standard deviation of 6.79, whereas, the average routing cost with the non-modular agents was 96.58 with the standard deviation of 7.93. Thus, the proposed algorithm performs better in 85% of the cases, yields a lower average cost when considering all cases, and has a lower standard deviation.

References

[1] Michael Barbehenn and Seth Hutchinson. Efficient search and hierarchical motion planning by dynamically maintaining single-source shortest paths trees. *IEEE Transactions on Robotics and Automation*, 11(2):198–214, 1995.

[2] Zhiwei Chen, Xiaopeng Li, and Xuesong Zhou. Operational design for shuttle systems with modular vehicles under oversaturated traffic: Continuous modeling method. *Transportation Research Part B: Methodological*, 132:76–100, 2020.

[3] Jean M Dasch and David J Gorsich. Survey of modular military vehicles: Benefits and burdens. Technical report, Army Tank Automotive Research, Development and Engineering Center (TARDEC), 2016.

[4] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):224–227, 1979.

[5] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *6th Annual ACM Symposium on Theory of Computing*, pages 47–63, 1974.

[6] Google maps. https://www.google.com/maps/search/amazon+warehouses+near+los+angeles.

[7] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[8] Karan Jagdale and Melkior Ornik. Optimal routing of modular agents on a graph. *arXiv preprint arXiv:2302.04933*, 2023.

[9] Yuxiong Ji, Bing Liu, Yu Shen, and Yuchuan Du. Scheduling strategy for transit routes with modular autonomous vehicles. *International Journal of Transportation Science and Technology*, 10(2):121–135, 2021.

[10] Zaid S Khan, Weili He, and Mónica Menéndez. Application of modular vehicle technology to mitigate bus bunching. *Transportation Research Part C: Emerging Technologies*, 146, 2023.

[11] Qianwen Li and Xiaopeng Li. Trajectory planning for autonomous modular vehicle docking and autonomous vehicle platooning operations. *Transportation Research Part E: Logistics and Transportation Review*, 166, 2022.

[12] Xiaohan Liu, Xiaobo Qu, and Xiaolei Ma. Improving flex-route transit services with modular autonomous vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 149, 2021.

[13] Next Future Transportation. https://www.next-future-mobility.com/.

[14] Christian Nilsson. Heuristics for the traveling salesman problem. *Linkoping University*, 38:00085–9, 2003.

[15] Mingyang Pei, Peiqun Lin, Jun Du, Xiaopeng Li, and Zhiwei Chen. Vehicle dispatching in modular transit networks: A mixed-integer nonlinear programming model. *Transportation Research Part E: Logistics and Transportation Review*, 147, 2021.

[16] Xiaowei Shi, Zhiwei Chen, Mingyang Pei, and Xiaopeng Li. Variable-capacity operations with modular transits for shared-use corridors. *Transportation Research Record*, 2674(9):230–244, 2020.