A Solution to the Service Agent Transport Problem

Matthew J. Bays Thomas A. Wettergren

Abstract—We introduce a new problem in the area of scheduling and route planning operations called the service agent transport problem (SATP). Within the SATP, autonomous service agents must perform tasks at a number of locations. The agents are free to move between locations, however, the agents may also be transported throughout the region by a limited number of faster-moving transport agents. The goal of the SATP is to plan a schedule of service agent and transport agent actions such that all locations are serviced in the shortest amount of time. We pose the problem using a mixed-integer linear programming optimization framework, and compare several complexity reduction heuristics to the full optimization.

I. INTRODUCTION

The use of autonomous systems to perform increasingly complex and coordinated tasks has necessitated creating heterogeneous teams of agents, where different agent types specialize in different parts of an operation [1]–[3]. One such heterogeneous team operation is when one type of mobile agent is tasked with performing the direct servicing tasks for the operation, while another type of larger, faster-moving, or longer-range agent is responsible for transporting the service agents between jobs for faster completion. Within manned systems, there are numerous examples of the transport agent/service agent concept such as aircraft carriers and their respective aircraft, garbage trucks and accompanying garbage workers, or mail delivery and their respective postmen. While this form of close interaction between unmanned systems is still far from common, the underlying hardware and guidance infrastructure to allow autonomous docking and deployment between unmanned systems is actively researched [4]–[8].

Just as important as the fundamental infrastructure of docking and deploying unmanned systems autonomously is determining the most efficient schedule for when and where to perform these actions when multiple agents can be assigned to perform an operation. The main contribution of our work is a formal mathematical framework for planning a schedule of service agent and transport agent actions such that all locations are serviced in the shortest amount of time. We call the framework the service agent transport problem (SATP). In addition to defining the mathematical framework, we show that the problem is NP-hard, and propose several methods to limit the search space for use with mixed-integer linear programming (MILP) techniques.

The problem of scheduling and task allocation has been extensively studied with a variety of techniques in multiple different forms. Koes et al developed a coordination architecture for modeling multirobot coordination and task allocation in [9]. Our framework compliments this methodology by expanding the types of scenarios under which a constraint optimization framework can be used: namely to problems where there are strongly coupled constraints on agents such as transportation actions.

Korsah et al create a framework to explore optimal vehicle routing with cross-schedule constraints with applications to robotic assistance in [10]. However, there are several notable differences to our work. Firstly, the principal element of optimization is an agent route, not the collection of agent actions comprising the route. Additionally in Korsah’s problem, the individuals being transported along the routes are static; there is no mechanism within the optimization framework for the individuals themselves to move about the area. As such [10] is similar to the traditional vehicle routing problem (VRP) [11]. Similarly, Mathew et al addreses finding the shortest route for an unmanned ground vehicle (UGV) and unmanned aerial vehicle (UAV) pair to transit and make deliveries to locations only reachable by the UAV in [12]. However, their work focuses primarily on the overall path planning for the two vehicles, and not on fuel limitations and the higher-level scheduling aspects required when there are an arbitrary number of UAVs and UGVs. Our work takes a scheduling-centric approach, formally incorporates fuel constraints, and is generalized for an arbitrary number of service agents and transport agents.

Gombolay et al develop a centralized algorithm to efficiently schedule manufacturing processes using robotic teams in [13]. However, they primarily focus on the development of a task sequencing heuristic that allows the generalized problem, a Simple Temporal Problem [14], coupled with simple spacio-temporal constraints, to be tractable for large numbers of tasks and robots to be assigned to a fixed set of tasks. Our principal contribution is in the unique modeling and constraint-based logic required for the docking, transport, and deployment of service agents throughout the environment. The development of the non-trivial constraints within our framework allows for applications where a list of required tasks is not given, but instead a variety of different methods are available to the agents to collectively complete a set of higher-level objectives by the agents’ interaction.

This paper is outlined as follows: in Section II, we discuss the general setup of the service agent transport problem. In Section III, we outline the mathematical constraints that make up the SATP framework. We propose several heuristics in order to limit the computational complexity in IV. Finally, simulation results showing the performance of various im-
implementations of the SATP are discussed in Section V.

II. PROBLEM DEFINITION

Let there exist a set of autonomous agents $A = \{1, \ldots, A\}$ that are tasked with servicing a number of service areas $S = \{1, \ldots, S\}$. The service areas are connected by a directed graph $(D, V)$. The nodes $D = \{1, \ldots, D\}$ represent locations at which an agent $a \in A$ may enter a service area $s \in S$ in order to perform operations. The agents may move from node $i$ to node $j$ along an edge of the graph $v = (i, j) \in V$. The subset $D_a \subseteq D$ represents nodes from which agents may service the service area $s$.

The agents are separated into two mutually exclusive and collectively exhaustive subsets $M$ and $N$ consisting of service agents and transport agents, respectively. A service agent $m \in M$ is capable of directly servicing the service areas, while a transport agent $n \in N$ may collect a fixed number of the service agents and transport them between nodes. The service agents may move between nodes with a time cost $c_{ij}^{\text{move}}$. However, they are significantly slower than transport agents with transportation cost of $c_{ij}^{\text{trans}} \leq c_{ij}^{\text{move}}$. The transport agents also require a fixed amount of time to collect and deploy the service agents. The collection and deployment costs are denoted $c_{\text{dock}}$ and $c_{\text{deploy}}$, respectively.

The agents initially start at an origin node $o$, with each transport agent $n$ containing $A_n$ service agents. Figure 1 shows a notional illustration of the service agent transport problem.

![Notional illustration of service agent transport problem](image)

We propose a phase-based model for the problem, where every agent may perform a single task during a phase of the mission, with phases labeled $P = \{1, \ldots, P\}$. Each agent-phase pair can have a varying time length that is dependent on the particular agent and action scheduled to be performed during the phase. During a phase, an agent may perform at most one task from the set of tasks $K = \{\text{move, service, dock, deploy}\}$. When service agents perform a move action, they only move themselves, while transport agents may simultaneously move up to a specified number of service agents currently docked with the transport agent.

Given the above terminology, the phase-based optimization problem can be formulated as follows:

**Problem II.1. SATP - Find a sequence of service agent and transport agent tasks $\bar{I}$ and start times $\bar{T}_{\text{start}}$ during phases $1, \ldots, P$ such that all areas $S$ are serviced, fuel limitations are observed during each phase, and all service agents are docked with a transport agent at the end of phase $P$ such that $\max_{a \in A} \bar{T}_{\text{end}} - \bar{T}_{\text{start}}$ is minimized.**

The primary optimization variables for our specific solution to the SATP, which we call decision variables, fall into two groups: scheduling indicator variables and time variables. The scheduling indicator variable $\bar{T}_{a,p,k} \in \{0,1\}$ denotes that task type $k$ is scheduled for completion by agent $a$ during phase $p$ at location set $L \in L_k$, where $L_k$ is the set of all feasible locations required for execution of task type $k$ and $\bigcup_{k \in K} L_k = L$. The timing variables $\bar{T}_{a,p} \in \mathbb{R}$ and $\bar{T}_{a,p}^\text{end} \in \mathbb{R}$ denote the times at which the task of agent $a$ during phase $p$ is scheduled to start and end, respectively. In contrast to the decision variables, additional variables are needed for the optimization problem in order to develop the necessary constraints to formulate the optimization problem. While the additional variables, which we denote as helper variables are technically variables used in the optimization, they are not of primary concern for the optimization problem. We will discuss the helper variables as the details of the optimization problem are formulated.

III. CONSTRAINT FORMULATION

We now turn to defining the constraints required for the mathematical formulation of the service agent transport problem.

**A. Docking and Deployment Constraints**

The first constraint within the SATP is that a service agent must only complete certain tasks when they are docked or deployed, as appropriate. For enforcing this constraint, we create the helper variables to track whether a service agent $m$ is docked or deployed from a transport agent $n$. We denote this variable

$$\bar{D}_{m,n,p} = \begin{cases} 1 & \text{Agent } m \text{ is docked with agent } n \text{ in phase } p \\ 0 & \text{Otherwise} \end{cases}$$  \hspace{1cm} (1)

Within an integer programming framework, we create (1) with the formal constraint

$$\forall m \in M, \forall n \in N, \forall p \in P \quad \bar{D}_{m,n,p} = \sum_{p' \leq p, d \in D} \left[ \bar{T}_{\text{dock}}_{mn,p',d}^{m,n,p',d} - \bar{T}_{\text{deploy}}_{mn,p',d}^{m,n,p',d} \right],$$  \hspace{1cm} (2)

where $\bar{T}_{\text{dock}}_{mn,p',d}^{m,n,p',d}$ and $\bar{T}_{\text{deploy}}_{mn,p',d}^{m,n,p',d}$ are indicator variable docking and deploying, respectively, at particular dock/deploy nodes $d$. We bound $\bar{D}_{m,n,p}$ with the constraint

$$\forall m \in M, \forall n \in N, \forall p \in P \quad 0 \leq \bar{D}_{m,n,p} \leq 1.$$  \hspace{1cm} (3)
With constraints (2) and (3), $\bar{D}_{mn,p}$ equals 1 if agent $m$ is currently docked with agent $n$ during phase $p$, and 0 otherwise. Constraint (2) and the lower bound in (3) satisfies the additional requirement that in order for a service agent to be deployed, it must first have been docked. It is also necessary to track if an agent is docked or deployed in general. This is performed using the helper variable $D_{mn,p}$ and constraints
\[ \forall m \in M, \forall p \in P \quad D_{m,p} = \sum_{n \in N} D_{mn,p} \tag{4} \]
and
\[ \forall m \in M, \forall p \in P \quad 0 \leq D_{m,p} \leq 1 \tag{5} \]
Additionally, we limit the total number of service agents that can be docked at a given time by a transport agent to $D_{max}$ using the constraint
\[ \forall n \in N, \forall p \in P \quad \sum_{m \in M} D_{mn,p} \leq D_{max} \tag{7} \]
With constraints (4)-(7), we can now limit the feasible actions of the individual agents when deployed or docked as appropriate.

B. Fuel Limitation and Charging Constraints

Fuel consumption is an additional variable that must be tracked for each agent during all phases of the mission. For tracking fuel, we define the variable $F_{m,p} \in \mathbb{R}$ as the amount of fuel service agent $m$ has available during the start of phase $P$. The variable $F_{m,p}$ is assigned a value using the constraints
\[ \forall m \in M, \forall p \in P \quad F_{m,p} = \sum_{p' \leq p} F_{m,p'} - \sum_{s \in S} f_{m,s} \bar{I}_{m,p,s} - \sum_{d \in D} \left( f_{m,d}^{\text{dock}} \bar{D}_{m,p,d} + f_{m,d}^{\text{deploy}} \bar{I}_{m,p} \right) - \sum_{v \in V} f_{m,p,v}^{\text{move}} \bar{I}_{m,p,v} \tag{8} \]
where $f_{m,s}$, $f_{m,d}^{\text{dock}}$, $f_{m,d}^{\text{deploy}}$, and $f_{m,p,v}^{\text{move}}$ are fixed costs for executing their respective goals, and $F_{m,p}$ is a floating variable indicating the amount an agent is charged during a phase. The variable $F_{m,p}$ is fixed to its value using the linear service agent charge rate $c_{m}^{\text{charge}}$ and the constraints
\[ \forall m \in M, \forall p \in P \]
\[ D_{m,p} = 1 \Rightarrow F_{m,p} \leq c_{m}^{\text{charge}} (T_{\text{end},m,p} - T_{\text{start},m,p}) \tag{9} \]
\[ D_{m,p} = 0 \Rightarrow F_{m,p} = 0. \]
Additionally, we limit the fuel capacity with the constraints
\[ \forall m \in M, \forall p \in P \quad F_{m,p} \leq F_{max} \tag{10} \]
\[ \forall m \in M, \forall p \in P \quad 0 \leq F_{m,p} \tag{11} \]

C. Goal and Capability Constraints

In our formulation, each agent can only prosecute one goal during a phase of the mission. As such, we have the constraint
\[ \forall a \in A, \forall p \in P \quad \sum_{k \in K} \sum_{l \in L_k} \bar{I}_{a,p,l} \leq 1. \tag{12} \]
As not every agent is capable of completing every task, there requires additional constraints to force only agents capable of completing a task to be scheduled for the task. As such, we have the constraint
\[ \forall a \notin A_k, \forall p \in P, \forall l \in L_k \quad \bar{I}_{a,p,l} = 0 \tag{13} \]
We note that this constraint may also be implemented for tractability by limiting both the scheduling variables $\bar{I}_{a,p,l}$ and constraints specific to a certain capability that are created in the optimization problem to those allowed by the individual vehicle’s capabilities.

D. Time Dependency and Wait Constraints

We now turn to the constraints required for determining the specific times at which phases are scheduled for starting and ending. Without loss of generality, we assume that the start time of the initial phase is at $t = 0$. As such we have the constraint
\[ \forall a \in A \quad T_{a,p=0} = 0. \tag{14} \]
In order to adhere to the boundary conditions between the end of the last phase and the start of the next phase, we have the constraint
\[ \forall a \in A, \forall p \in P \mid p \neq 0 \quad T_{a,p+1} - T_{a,p} \geq 0 \tag{15} \]
With constraints (14) and (15), we have defined all variables with the exception of the end time of each phase, $T_{\text{end},a,p}$. The end time may be defined by taking advantage of (12) with only one task being scheduled for an agent during a phase
\[ \forall a \in A, \forall p \in P \quad T_{a,p} = T_{\text{start},a,p} + \sum_{k \in K} \sum_{l \in L_k} c_{k,a,l} \bar{I}_{a,p,l} \tag{16} \]
where $c_{k,a,l}$ is the duration of task $k$ if performed by agent $a$ on location set $L$. Additionally, the dual relationship between a service agent and transport agent performing docking and deployment constraints necessitates that we must ensure that those particular operations occur at the same time for both vehicles. Thus, we have the time dependency constraints
\[ \forall m \in M, \forall n \in N, \forall p \in P \quad \sum_{d \in D} \bar{I}_{mn,p,d} = 1 \Rightarrow T_{m,p} = T_{\text{start},m,p} \tag{17} \]
\[ \forall m \in M, \forall n \in N, \forall p \in P \quad \sum_{d \in D} \bar{I}_{mn,p,d} = 1 \Rightarrow T_{m,p} = T_{\text{end},m,p} \tag{18} \]
Some scheduling situations necessitate an agent loitering or otherwise performing no meaningful actions during a given phase so that the agent may properly coordinate with
the actions of other agents. For example, a transport agent may need to wait at a docking point during a phase in which a service agent is servicing an area with which it will dock in the following phase. In this case, we require an additional variable indicating that an agent is idle during that phase at a given location. We define the \( \bar{I}_{a,p,d} \) indicator variable by the constraints

\[
\forall a \in A, \forall p \in P, \forall d \in D \\
\sum_{k \in \mathcal{K}} \left[ \sum_{L \in \mathcal{L}_k} \bar{I}_{a,p,l} \right] + \sum_{d' \in D | d' \neq d} \bar{I}_{a,p,d'} = 1 \\
\Rightarrow \bar{I}_{a,p,d} = 0
\]

\[
\forall a \in A, \forall k \in \mathcal{K}, \forall p \in P, \forall d \in D \\
\sum_{L \in \mathcal{L}_k} \bar{I}_{a,p-1,L} + \sum_{d' \in D | d' \neq d} \bar{I}_{a,p-1,d'} = 1 \\
\Rightarrow \bar{I}_{a,p,d} = 0
\]

Constraint (19) forces \( \bar{I}_{a,p,d} \) to equal zero if the agent is otherwise tasked or waiting at another location. Constraint (20) forces \( \bar{I}_{a,p,d} \) to equal zero if the agent performed an action at a different location \( d' \) during the previous time step, thereby prohibiting the agent from waiting at location \( d \). Constraint (21) forces \( \bar{I}_{a,p,d} \) to one if the agent is not otherwise tasked.

\[E. \ Transit \ and \ Transport \ Constraints\]

We now turn to the constraints required for transiting vehicles and transporting them. In order for a vehicle to transit from one location to the next, it must have performed an action at the previous location

\[
\forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall p \in P, \forall d_0 \in D \\
\sum_{d \in D} \bar{I}_{m,p,(d_0,d)} = 1 \\
\Rightarrow \sum_{d \in D} \bar{I}_{m,p-1,(d,d_0)} + \sum_{n \in \mathcal{N}} \bar{I}_{m,n,p-1,d_0} + \sum_{s \in S \cap (d_0,d) \in \mathcal{D}_s} \bar{I}_{m,p-1,s} + \bar{I}_{a,p,d} = 1.
\]

where \((d_0,d),(d',d_0)\) \in \mathcal{V}\. Additionally, we have the constraint that for a vehicle to either move from one area to the next or to service an area, the vehicle must be deployed

\[
\forall m \in \mathcal{M}, \forall p \in P \\
\sum_{n \in \mathcal{N}} \bar{I}_{m,n,p} = 1 \Rightarrow \bar{D}_{m,p} = 0
\]

For an agent \( m \) to be docked with another agent \( n \) capable of capturing agents, both agents must have been located previously at the location of the capture. Thus, we have

\[
\forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall p \in P, \forall d \in D \\
\bar{I}_{\text{dock}_{m,n,p,d}} = 1 \Rightarrow \sum_{d_0 \in D} \bar{I}_{\text{move}_{m,p-1,d_0},d} + \bar{I}_{\text{wait}_{m,p-1,d}} \\
+ \sum_{s \in \mathcal{S} | (d_0,d) \in \mathcal{D}_s} \bar{I}_{\text{service}_{m,p-1,d}} = 1 \\
\bar{I}_{\text{move}_{m,n,p-1,d}} = 1 \\
+ \bar{I}_{\text{dock}_{m,n,p-1,d}} + \bar{I}_{\text{deploy}_{m,n,p-1,d}} \geq 1 \\
m' \neq m
\]

where \((d_0,d),(d_0',d)\) \in \mathcal{V}\. Likewise, we have the constraint for deploying vehicles as

\[
\forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall p \in P, \forall d \in D \\
\bar{I}_{\text{deploy}_{m,n,p,d}} = 1 \Rightarrow \bar{D}_{m,n,p,d} = 1 \\
\bar{I}_{\text{move}_{m,n,p-1,d}} = 1 \\
+ \bar{I}_{\text{dock}_{m,n,p-1,d}} + \bar{I}_{\text{deploy}_{m,n,p-1,d}} \geq 1 \\
m' \neq m
\]

\[F. \ Servicing \ Constraints\]

The first constraint for a servicing action is that a service area can only be serviced if it is scheduled after a deployment action at a node \( d \in \mathcal{D}_s \) or move action. Thus, we have the constraint

\[
\forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall p \in P, \forall d \in D, \forall s \in \mathcal{S} \\
\bar{I}_{\text{service}_{m,n,p,s}} = 1 \Rightarrow \sum_{d_0 \in \mathcal{D}_s} \bar{I}_{\text{deploy}_{m,n,p-1,d}} + \sum_{d \in D} \bar{I}_{\text{move}_{m,n,p-1,d}} = 1.
\]

where \((d',d)\) \in \mathcal{V}\. Next, we have the constraint that a service action can only be performed if it is scheduled while the agent is deployed

\[
\forall m \in \mathcal{M}, \forall p \in P \\
\sum_{n \in \mathcal{N}} \bar{I}_{\text{service}_{m,p,n}} = 1 \Rightarrow \bar{D}_{m,p} = 0
\]

Additionally, every service goal should be completed exactly once by the group of agents. Thus,

\[
\forall s \in \mathcal{S} \\
\sum_{m \in \mathcal{M}} \sum_{p \in \mathcal{P}} \bar{I}_{\text{service}_{m,p,s}} = 1.
\]

\[G. \ Cost \ Function\]

The cost function to be minimized in our initial formulation is to minimize the end time of the last phase over all agents. That is, our objective is

\[
\text{minimize } \max_a \bar{T}_{a,p}.
\]
IV. COMPLEXITY AND COMPLEXITY REDUCTION

Now that we have fully modeled the service agent transport problem, we now discuss initial methods to limit the search space required by MILP solvers. The full optimization is NP hard, which can be shown with the following proofs.

**Proposition IV.1.** The SATP belongs to the class NP.

*Proof.* We first pose the corresponding decision problem to the optimization problem constructed from constraints (2)-(30) and cost function (31) as a formal language.

\[
SATP = \{(D, V), c, S, M, N, L, f(\cdot), P, T'\}
\]

\[
f : D \to S
\]

\[T' \in \mathbb{R}
\]

\[M \text{ service agents and } N \text{ transport agents are assigned tasks during phases } P = \{1, \ldots, P\}
\]

such that constraints (2)-(30) hold with

\[\max_{p \in P} T_{a, p}'\]

Next, consider a two-input, polynomial time algorithm \(A(\cdot)\) that can verify \(SATP\). One of the inputs to \(A(\cdot)\) is an implementation of constraints (2)-(30). The other input is a binary assignment of the decision variables \(\mathbf{T}_{a, p}, \mathbf{T}_{\text{start}}, \mathbf{D}_{m, p}, \mathbf{E}_{m, p}, \text{ and } T_{a, p}'\). We construct algorithm \(A(\cdot)\) as follows: for each instantiation of the constraints, we check to see if the constraints hold in polynomial time. For implementation of constraints (9), (17)-(30), we first convert them to a standard MILP implementation using techniques such as Big-M [15]. The conversion can also be done in polynomial time. The algorithm then finds the maximum value of \(T_{a, p}'\) and checks to see if it is at most \(T'\). This process certainly can be done in polynomial time. □

**Proposition IV.2.** The SATP is NP-complete.

*Proof.* To prove the SAT is NP-Complete, we must show that it 1) belongs to NP, and 2) is polynomial-time reducible to another problem that is NP-Complete [16]. Proposition IV.1 satisfies the first criterion. For the second criterion, we will use the Traveling Salesman Problem (TSP), which has been shown to be NP-Complete [16]. We will now show \(SATP \leq_p TSP\).

\[
TSP = \{(D_t, V_t), c_t, T'\}
\]

\[c_t : V \times V \to \mathbb{R}
\]

\[T' \in \mathbb{R}
\]

\[\langle D_t, V_t \rangle \text{ has a tour of at most } T_t.
\]

We first construct an algorithm \(f_t : TSP \to SATP\) as follows. Since we make no restriction on node \(d \in D_S\) being exclusive to service area \(s \in S\) within SATP we assign the edges of \(D_t\) to map one-to-one to individual service areas, all service areas reachable by service agents and transport agents by all edges, and one of the service areas to serve as the origin node \(o\). Thus, there are \(S = |D_t|\) service areas in the SATP equivalent. The standard TSP problem assumes a route for only one agent. We therefore restrict all mappings of TSP to the case where \(N = M = 1\) within the new SATP problem. Likewise, we may consider the 'phases' within the TSP as equal to twice the number of vertices (one each for transit and service), plus two for docking and deployment of the single service agent by the single transport agent. Thus, we have \(P = |2D_t| + 2\). For clarity, we will remove the now superfluous subscripts \(m\) and \(n\). Let \(c_{\text{dock}} = c_{\text{deploy}} = c_{\text{service}} = 0\) in the new SATP. Let \(f_{\text{service}} = f_{\text{dock}} = f_{\text{deploy}} = f_{\text{move}} = c_{\text{charge}} = 0\). Let \(c_{\text{move}} = c_{\text{trans}}\) for all \(i, j \geq v \in V\). These assignments can be done in polynomial time.

We now show that TSP has a solution of at most \(T'\) if and only if the mapped SATP has a solution of at most \(T'\) as well. Suppose TSP has a solution of at most \(T'\). The assignment of fuel costs to zero causes constraints (8)-(10) to become trivial, and the combination of \(c_{\text{dock}} = c_{\text{deploy}} = 0\) and \(c_{\text{move}} = c_{\text{trans}}\) cause there to be no difference in servicing time between a service agent moving or being transported and deployed between service areas via (16). The assignment \(P = |2D_t| + 2\) cause the only feasible schedules to be those involving an initial deployment of the single service agent, followed by a TSP tour of alternating service agent transits and servicing actions followed by a single docking, due to constraints (2)-(7) and (30). However, the equal transport/move cost and zero dock and deploy cost assignment causes (31) to degenerate into the simple tour cost of TSP. Thus, the resulting SATP has a minimum time of at most \(T'\) as well. Next, suppose that SATP has a completion time of at most \(T'\). Then, since each service node must have been visited exactly once due to the choice of \(P\), and all costs are zero with the exception of transit costs, the corresponding TSP problem has a tour cost of at most \(T'\) as well. □

We now present two strategies for increasing the tractability of the SATP by limiting the required number of variables needed in a particular SATP implementation. The first strategy reduces the number of candidate dock, deploy, and transit nodes to the closest \(S\) nodes that provide one node per service area. The second strategy maintains the number of candidate nodes, but eliminates all transition edges between nodes within the same service area. Since the embedded routing problem significantly increases the complexity of scheduling problem as a whole, the general idea of both reduction strategies is to decrease the routing problem in intelligent ways.

**A. Node Reduction Strategy**

We first attempt to reduce the number of variables by selecting only a subset of dock and deploy nodes for consideration in the overall optimization framework, specifically the \(S\) nodes in a cluster that minimize the distance traveled to visit all nodes in the subset while providing a dock and deploy point to every service area \(S\). Formally, we can write
this node subset as
\[ D' = \left\{ \text{argmin}_{D_s} \sum_{i,j \in D_s} c_{ij} |i,j \in D, D \subset D, \forall s \in S, \right\} \]
\[ |D_s| = 1 \}
\tag{34} \]
where \( D_s \) is the number of nodes connected to service areas within the node subset \( D^* \). Equation (34) represents a form of the generalized traveling salesman problem (GTSP) [17], [18]. While the GTSP is another NP-hard problem, its use increases the tractability of the SATP in two ways. First, because the GTSP is partly encapsulated within the SATP when determining routes for the individual vehicles, solving the GTSP before the main schedule optimization will eliminate orders of magnitude more permutations required in the overall optimization than required to solve the GTSP. Second, there are several heuristics that can quickly provide efficient but sub-optimal solutions to the GTSP [17]–[19].

\section*{B. Edge Reduction Strategy}

A second candidate strategy for reducing the scale of the SATP is to reduce the number of decision variables by reducing the total number of candidate edges within the problem. Like the node reduction strategy, this is a balancing act between reducing the scale of the problem and eliminating options for the vehicles to travel in order to transit in an efficient manner. An appropriate compromise between the two conflicting needs is in the elimination of intra-service area edges. Specifically, we create a new edge map \( V' \) as
\[ V' = \{(i,j) \in V | i \in D_{s_i}, j \in D_{s_j}, s_i \neq s_j \}. \tag{35} \]
The intuition behind using (35) is the assumption that there is little need for either a service agent or transport agent to transit between two nodes in a single service area, as once an area is serviced, the agents can simply choose the best route to the next service area.

Transit paths between four service areas consisting of four dock and deploy nodes per area are shown in Figure 2. The full graph \( \langle D, V \rangle \) is shown on the left-hand side of the figure. The graph \( \langle D, V' \rangle \) resulting from edge reduction strategy is shown in the center. The graph \( \langle D', V' \rangle \) depicting the node reduction strategy is shown on the right. The edge reduction strategy eliminates approximately 18% of the edges in the search space for agent movement. The node reduction dramatically decreases the number of edges and nodes, reducing the number of edges to 8% of the full graph.

\section*{V. Simulation Results}

We now present simulation results for the proposed scheduling framework. All simulations were performed using the IBM’s ILOG CPLEX optimization software [20]. In the following simulations, we assume that a transport agent has a speed of 8 m/s, while service agents have a speed of 1.5 m/s. We assume each of the \( A = 4 \) service areas requires \( c_{\text{service}} = 100 \) minutes. Furthermore, we assume that each docking action requires \( c_{\text{dock}} = 20 \) minutes, and \( c_{\text{deploy}} = 10 \) minutes. All simulations were performed on a 2.8 GHz quad-core CPU equipped with 8 GB RAM.

Figures 3 and 4 illustrate a simulation of \( N = 1 \) transport agent, \( M = 3 \) service agents, and \( P = 10 \) phases using the node reduction strategy. Figure 3 represents the spatial map of the various agents moving throughout the field. The MILP optimization software found a feasible solution within 18% of the optimal solution within one second, and the globally optimal solution in 963.01 seconds, with an objective value of 247.0 minutes. As seen in the figure, the agent schedule for the transport agent deploy service agent 1 and 2 to a deployment node near service areas 2 and 3. Service agent 1 immediately starts servicing the two service areas, while service agent 2 travels to the further service area 1. Simultaneously, the transport agent transits to the furthest service area to deploy service agent 3. The transport agent then returns to a deployment node between service areas 2 and 3 to dock with the three service agents, which transit to the deployment node as they complete their service tasks.

Figure 5 shows results from a Monte Carlo simulation comparing the full graph to the two complexity reduction strategies discussed in Section IV for an SATP optimization problem involving \( N = 1 \) transport agent, \( M = 3 \) service agents, and \( P = 15 \) phases. Sixty simulations were performed with three service areas randomly generated over a 10,000m by 10,000m field. For each simulation, the cutoff time for providing the best solution from CPLEX for the optimization problem was 120 seconds. As seen in the figure, the node reduction strategy resulted in the fastest sortie time, averaging an objective value (31) of 281.4 minutes. The edge-reduction strategy provided a modest improvement over the full optimization model, averaging an objective value of 373.1. As expected, the full optimization model, due to its significantly greater complexity, resulted in the poorest solutions, averaging 398.8 minutes.

Additionally, in some instances, the optimization solver was unable to find a solution to the problem in the specified time limit due to the computational complexity of the full optimization problem. The timeout without a solution occurred in 48% of the full optimization problem’s simulations, and 23% of simulations using the edge reduction strategy. However, in no instances of using the node reduction strategy did a timeout occur.

A second Monte Carlo simulation was performed involving sixty simulations of a single transport agent and service agent, and \( P = 12 \) phases, allowing all optimization routines to run until a globally optimal solution is found. In this manner, the Monte Carlo simulation characterized how the two heuristic reduction strategies (34) and (35) impact the discovery of the globally optimal schedule by eliminating transit paths. In all cases, the globally optimal solution to the full optimization problem was identical to the globally optimal solution of the edge reduction strategy (35). The node reduction strategy (34) did eliminate the globally optimal solution in all instances, however this elimination increased the minimum solution’s objective function value.
by an average of only 3.45%, and maximum of 7.7%.

VI. CONCLUSIONS

We have presented a framework for developing efficient schedules for multiple heterogeneous vehicles we call the service agent transport problem. The goal of the vehicles tasked in the framework is to perform servicing operations within an area when one type of agent acts as a transport vehicle for a second type of agent performing the actual servicing operations. The framework utilizes mixed-integer linear programming techniques to create novel constraints required to develop a tasking schedule for the vehicles including docking, deployment, and movement actions performed throughout the area. Future work includes extending the framework to include probabilistic constraints in order to account for uncertainty in the length of time required to perform tasks, as well as exploring decentralized optimization strategies to increase the scalability of the optimization.
framework.

REFERENCES


