Computers & Industrial Engineering 176 (2023) 108852

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Multi-orbit routing and scheduling of refuellable on-orbit servicing space robots

Susan E. Sorenson [*], Sarah G. Nurre Pinkley

Department of Industrial Engineering, 4207 Bell Engineering Center, 1 University of Arkansas, Fayetteville, 72701, Arkansas, United States of America

A R T I C L E I N F O

Keywords:
On-orbit servicing
Multi-orbit
Optimization
Routing
Scheduling

A B S T R A C T

We present the Multi-Orbit Routing and Scheduling of Refuellable Space Robots for On-Orbit Servicing optimization problem which determines how to best route and schedule a fleet of highly maneuverable and refuellable space robot servicers to complete a set of tasks orbiting in space. We formulate this problem as a mixed-integer linear program and seek to maximize the weighted number of completed tasks subject to constraints related to the movements of the space robots, refueling depots, and tasks. We present and demonstrate algorithms for constructing the network and model using case studies with data based on satellites operating in the Low Earth, Mid Earth, and Geosynchronous Earth Orbits. Our results indicate the benefit of considering multiple orbits and policies related to the number and starting locations of robot servicers and refueling depots.

1. Introduction

Soon after the Hubble Telescope was launched in 1990, scientists discovered a defect in one of its mirrors which made the images it collected useless (NASA, 2021a). This defect motivated one of the earliest realizations of on-orbit servicing (OOS) and in December 1993, NASA astronauts repaired and replaced parts, restoring the operational capability of the telescope (NASA, 2021a). NASA astronauts serviced the Hubble Telescope four additional times over the next 16 years (NASA, 2021a). More recently, Northrup Grumman executed two OOS missions for Intelsat satellites in 2020 and again in 2021 (Northrup Grumman, 2020). Their vision is to establish a fleet of servicers in Geosynchronous Orbit to address most any servicing need (Northrup Grumman, 2020). Although not yet commonplace, over the past two decades, researchers and engineers have made great strides in the technology which will eventually make OOS the norm, rather than a novel occurrence. We hypothesize that in the not too distant future, corporations will have on-orbit refueling depots and fleets of refuellable space robots capable of accomplishing satellite servicing missions across Low Earth Orbit (LEO), Mid Earth Orbit (MEO), and Geosynchronous Orbit (GEO). The routing and scheduling of a set of OOS tasks over an upcoming day, week, or month will become a more common and important problem. This work aims to demonstrate the formulation and solvability of this scheduling challenge using a Mixed Integer Linear Program (MILP). We begin the discussion with a review of the OOS technology which relates to the formulation of our problem.

NASA continues developing future concepts and architectures, such as the Lunar Gateway and manned Mars missions which will require significant use of OOS and manufacturing. One of these concepts is the On-orbit Servicing, Assembly and Manufacturing-1, a robotic spacecraft designed to operate in LEO, equipped with the tools, technologies, and techniques needed to extend satellites' lifespans—even if they were not originally designed to be serviced on-orbit (NASA, 2021). In 2019, as part of a clean space initiative, the European Space Agency (ESA) announced a call for proposals from European companies to develop OOS robots for the safe removal of ESA-owned satellites (ESA, 2018). We point the reader to Li et al. (2019) for a review of over 130 launched or proposed engineering developments related to OOS missions. Additionally, Corbin et al. (2020) provides a more recent review of international OOS developments for On-orbit Servicing, Assembly or Manufacturing.

In the future, as OOS missions grow in number and magnitude, any long term OOS undertaking will require a space based refueling depot so that on-orbit robot servicers can refuel themselves and conduct On Orbit Refueling (OOR) and OOS of other satellites. In late 2020, NASA awarded several contracts to companies for the development and/or demonstration of space and lunar based fuel depots (NASA, 2020). An aerospace company based in the United Kingdom, Thales Alenia Space, will build a chemical refueling station with an expected launch in 2027 (British Broadcasting Corporation, 2020). Although fuel depots to support OOS missions are behind OOS technologies, they are

* Corresponding author.
E-mail address: sesoren68@gmail.com (S.E. Sorenson).

https://doi.org/10.1016/j.cie.2022.108852
Received 27 March 2022; Received in revised form 2 September 2022; Accepted 26 November 2022
Available online 14 December 2022

undoubtedly in development and eventually will be able to support OOS missions as envisioned in this work.

We present the Multi-Orbit Routing and Scheduling of Refuellable Space Robots for On-Orbit Servicing (MORSO) problem which determines how to schedule and route a small fleet of highly maneuverable refuellable space robots to complete a set of OOS tasks spanning multiple orbits in space. We formulate MORSO as a new MILP optimization model which seeks to maximize the weighted number of completed tasks over a set time horizon. This model is complex because the tasks and refueling depots move through an orbit over time. Our model considers these complexities and we present a novel way to represent the movement of tasks and refueling depots over time. We present an algorithm to construct a network with arcs representing different types of orbital maneuvers and with this network as an input we present general, flexible algorithms to create data representing the movement of tasks and refueling depots over time and space. Using these data sets, we demonstrate the model using three case studies.

Main Contributions. The main contributions of this work are as follows: (i) We are the first to consider OOS over multiple orbits with moving tasks and moving refueling depots; (ii) We formulate a mixed integer linear program to optimize the routing and scheduling of space robots to accomplish OOS; (iii) We develop and demonstrate flexible algorithms for the creation of model parameters which include a multiple orbit network, and a novel way to represent the movement of tasks and refueling depots; (iv) We create robust case studies based on current operational satellites in LEO, MEO, and GEO and perform extensive computational experiments (v) and present example insights about robot servicers, task completion, and their use of refueling depots.

The remainder of this work is as follows. In Section 2, we summarize literature related to the optimization of different types OOS missions. In Section 3, we provide the algorithms to construct the network arcs and to create the data representing the moving tasks and refueling depots and then present a formal definition of the model. In Section 4, we demonstrate the model and algorithms using a case study with data based on the orbital parameters of current satellites and provide operational insights. In Section 5, we present conclusions and areas for further study.

2. Literature review

In this section, we summarize the literature relevant to the optimization of the routing and scheduling of autonomous satellites, space robots, to accomplish On-Orbit Servicing (OOS). OOS involves at least two participants, one who does the servicing and another that receives the service. We denote those who do the servicing as a servicer and those who receive the service as a task. Although refueling is a type of OOS, when we use the term refueling we refer to an action that a servicer will take at a space based refueling depot to replenish themselves for the purpose of accomplishing OOS missions which may or may not include the on-orbit refueling of tasks.

Many works address a class of on-orbit refueling (OOR) problem in which there is no a priori designated "servicer". Instead, any participant can act as a servicer to any other participant. In these works, the refueling is the sharing of fuel which is already on board one of the participants and usually occurs at a location away from both participants' starting locations. Dutta and Tsiotras (2007) sought to minimize the overall ΔV expended during maneuvers with a greedy random adaptive search procedure and Dutta and Tsiotras (2008) also sought to minimize the overall ΔV by optimally matching participants based on their fuel levels. Later, Dutta and Tsiotras (2010) found a lower bound of the overall ΔV consumption using a network flow optimization. Du et al. (2015) used a Mixed Integer Non-linear Programming (MINLP) formulation, with non-linear ΔV costs using the Tsiolkovsky rocket equation, to minimize the ΔV consumed and solved the problem using a Multi-island Genetic Algorithm. Shen and Tsiotras (2005) applied a matching algorithm to minimize the mission time for a single orbit peer-to-peer refueling problem. Yu et al. (2013) addressed a similar problem by solving two sub-problems. The first problem optimized assignments to equally distribute the fuel across all participants and the second minimized the overall ΔV cost. In all of these works, the focus was on a single orbit OOR without any designated servicers, using fuel which is already on board the participants. In our work, we have multiple designated servicers, which can refuel and replenish at orbiting fuel depots located across multiple orbits in our network.

Some authors address problems with a single designated servicer which must complete a set of OOR or OOS tasks. These problems also focus on the minimization of time and/or ΔV, but differ in how the model is formulated. Alfriend et al. (2006) addressed a single orbit, single servicer, OOR problem as a Traveling Salesman Problem (TSP) while Bourjolly et al. (2006) and Gürtuna and Trépanier (2003) used a Vehicle Routing Problem (VRP) formulation to model a multi-orbit single servicer, OOS problem. Zhang et al. (2014) considered a multi-objective optimization model, seeking to simultaneously minimize the ΔV and mission time while routing a single servicer to tasks in multiple orbits within the same orbital band. Zhou et al. (2017) sought to determine the refueling order, the optimal refueling time, and the optimal orbital transfers while also minimizing the ΔV and mission time. Yu et al. (2017) optimized the scheduling of a single servicer to multiple tasks in GEO, using a multi-objective optimization model seeking to minimize the fuel used, maximize the number of refueling events, and maximize the sum of the weights of the tasks to be completed. Zhao et al. (2017) used a MINLP model, with non-linear ΔV costs, to address an OOR problem with a single servicer in which the tasks requiring fuel moved to a servicing area for the OOR. These works all had only one servicer to accomplish the OOR and OOS tasks while minimizing the time and/or ΔV. Our work has multiple servicers operating in multiple orbital bands and we seek to maximize the weighted number of completed tasks.

Our work is the first to route and schedule multiple refuellable servicers across multiple orbital bands to accomplish multiple tasks moving over time. Our work is unique in that the servicers, the tasks, and the refueling depots are all moving throughout the network over time. Zhou et al. (2015) formulated a time-fixed OOR problem with multiple refuellable servicers based only in GEO. Although the servicers were moving the tasks were not and the model is not changing over time. Daneshjou et al. (2017) examined a time-fixed, single orbit, multiple servicer OOS problem using a multi-objective optimization model to determine optimal servicer orbital parameters to minimize the ΔV and time used. Zhang et al. (Oct 2019) used a location routing problem to solve an OOR problem with multiple refueling depots and multiple servicers, but their problem was in a single orbit and also time-fixed. Hudson and Kolosa (2020) addressed a single orbit multiple servicer OOS problem in GEO as a moving task TSP looking to maximize lifetime profit. Sarton du Jonchay et al. (2020) examined a single orbit multiple servicer OOS mission using a strategic level MILP model with decisions about servicer types and servicing tools. Although their work is not time-fixed, they focus on strategic decisions to minimize costs and do not have moving tasks and refueling depots. With the literature summarized, we proceed by formalizing the Multi-Orbit Routing and Scheduling of Refuellable Space Robots for On-Orbit Servicing (MORSO) problem notation and mathematical model.

3. Problem statement and methodology

We seek to solve the Multi-Orbit Routing and Scheduling of Refuellable Space Robots for On-Orbit Servicing (MORSO) problem by determining the maximum number of weighted tasks which can be accomplished within a set time horizon. Currently, for on-orbit servicing, fuel is valuable commodity which we implicitly consider when maximizing the number of completed tasks. It is not beneficial to excessively use fuel because this takes time to refuel which directly results in less completed tasks. A novel aspect of the problem is that the tasks, servicers and fuel depots are all are orbiting over time. We represent several orbits using a network and optimize the routing, scheduling, and refueling of space robot servicers through the network over time to complete tasks. We proceed by explaining the network, tasks, and robot servicers and present algorithms for the construction of each. We then define the decision variables, parameters and sets, and model.

3.1. Problem formulation and algorithm development

We consider a set of orbits and represent this set as a connected network of nodes, N and arcs, A. We represent each orbit with a subset of the nodes. We represent the nodes as stationary locations in space thus we treat the orbital parameter of true anomaly as a fixed value, but in convention it would indicate the position on the orbit that an object occupies at a given time. We connect a node i with another node j with an arc (i,j) if an orbital maneuver is possible. There are many maneuvers possible. We proceed by describing the four maneuvers we considered for our network. However, we note and emphasize that the model is suitable for networks with arcs representing other possible maneuvers which are outside of the scope of this work.

We add directed arcs to A to represent orbital maneuvers between nodes which share and do not share the same orbit. We add a directed arc (i,j) for an *orbiting maneuver* between nodes i and j which share the same orbit and are adjacent in the rotation direction of the orbit. When nodes i and j share the same orbit but are not adjacent in the rotation direction of the orbit, we add a directed arc (i,j) for a *phasing maneuver*. When nodes i and j are in different orbits, differ by 180° in their stationary true anomaly value, and are on the same plane and we add directed arc (i,j) as a *Hohmann transfer*. Finally, we add a directed arc (i,j) for a *Hohmann transfer with an inclination change* when nodes i and j are in different orbits, differ by 180° in their stationary true anomaly value, and are not on the same plane (inclination). For each arc $(i,j) \in A$, we associate known input parameter values. We denote τ_{ij} and ϕ_{ij} as the time and fuel (ΔV) needed to move along arc (i,j), respectively. We denote Ψ_{ij} as the per time period fuel needed to traverse arc (i,j) where $\phi_{ij} = \Psi_{ij}\tau_{ij}$. In Algorithm 1 we present our general pseudocode which dictates the construction of arcs in a network, given a set of nodes and their associated orbital parameters. Our algorithm only includes the four types of orbital maneuvers we considered, but can be modified to fit orbital maneuvers of any type. To consider other maneuvers, the Ψ_{ij} and τ_{ij} values for each arc (i,j) are needed. To further aid in the explanation of the Arc Creation Algorithm, we provide a detailed example in Appendix. For this work, we assume the time and ΔV costs for servicers to rendezvous with their tasks are 0, however, these values could be included. Note, when we use the term fuel, we are referring to ΔV, or the scalar amount of velocity required for a spacecraft of unit mass to accomplish a maneuver. For this reason, throughout this paper we use fuel and ΔV interchangeably. Please see Fig. 1 for a toy example depicting the network parameters.

Over time, through the network, tasks, servicers, and fuel depots are all moving. We proceed by describing the novel aspect of our work in which we model the movement of tasks, servicers, and fuel depots over time and space. We consider a discrete, finite time horizon, T, over which tasks are to be completed. We discretize this time horizon into distinct time intervals from $t = 0, \dots, |T|$.

We denote B as the set of tasks where each task $v \in B$ is prioritized by a weight w_v. Because tasks are orbiting, each task is at a different location (node in the network) at different times. Thus, we represent each task $v \in B$ with a set of sub tasks, B_v, where each sub task $k \in B_v$ is represented by a node and a time pair, i.e., (n_{vk}, t_{vk}). This indicates that task v will be at node n_{vk} at time t_{vk}. Thus, for a given task v and its associated starting node and orbit, we first create a sub task $k \in B_v$. Next, we iteratively move in the rotational direction of the orbit along the input set of arcs representing *orbiting maneuvers*. With each new node visited, we create a sub task pair based on the node location and current time calculated based on the traversal time of the orbiting maneuver arc. We continuously proceed with this process until the time horizon is reached. We present explicit details of these steps

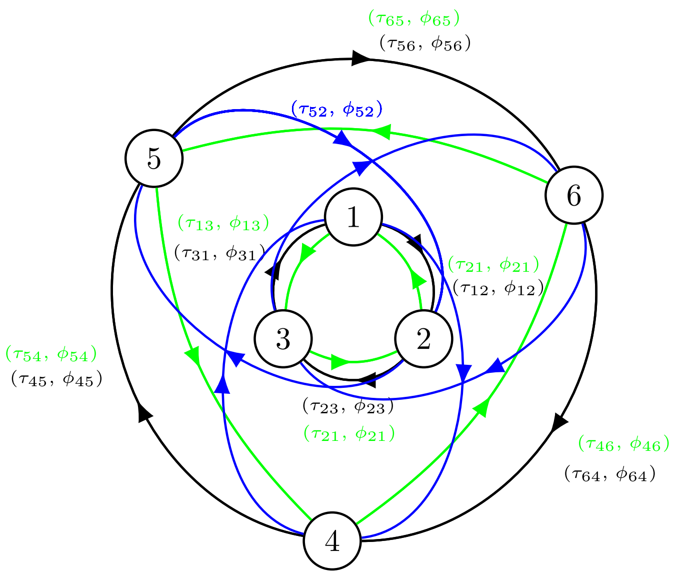

Fig. 1. Consider a toy network to illustrate the network parameters. The network has 6 nodes: $N = 1,2,3,4,5,6$ and 24 arcs: $A = \{$(1,2) (2,3) (3,1) (4,5) (5,6) (6,4) (2,1) (3,1) (1,2) (4,6) (5,4) (6,5) (1,4) (4,1) (2,5) (5,2) (3,6) (6,3)$\}$. Arcs representing orbiting arcs are colored black. Arcs representing Hohmann transfer arcs are colored blue. Arcs representing phasing maneuvers are colored green and each arc is labeled with its corresponding time and fuel costs (τ_{ij}, ϕ_{ij}).

Algorithm 1 Arc Creation Algorithm

```
1:  Create arc set $A = \emptyset$.
2:  Input $\Delta t$ as the time period length, $N$ as the node set, and $T$ as the time horizon.
3:  for $i, j \in N,\ i \neq j$ do
4:      Set $\Delta\theta$ = the angle between nodes $i$ and $j$.
5:      if $i$ and $j$ are in the same orbit and are adjacent in rotation direction of orbit then
6:          Create $(i,j)$ representing an orbiting maneuver.
7:          Set $P$ as the period of orbit for nodes $i$ and $j$.
8:          Set $\tau_{ij} = \frac{\Delta\theta}{360} \times \frac{P}{\Delta t}$, $\phi_{ij} = 0$, $\Psi_{ij} = 0$.
9:      else if $i$ and $j$ are in the same orbit and are not adjacent in rotation direction of orbit then
10:         Create $(i,j)$ representing a phasing maneuver.
11:         Calculate the $\tau_{ij}$ = time and $\phi_{ij} = \Delta V$ as shown in Appendix.
12:     else if the inclination of $i$ = inclination of $j$ AND the semi-major axis of $i \neq$ semi-major axis of $j$ AND $\Delta\theta = 180$ then
13:         Create $(i,j)$ representing an Hohmann transfer.
14:         Calculate the $\tau_{ij}$ = time and $\phi_{ij} = \Delta V$ as shown in Appendix.
15:     else if $i \neq j$ AND the inclination of $i \neq$ inclination of $j$ AND the semi-major axis of $i \neq$ semi-major axis of $j$ AND $\Delta\theta = 180$ then
16:         Create $(i,j)$ representing a Hohmann transfer with an inclination change.
17:         Calculate the $\tau_{ij}$ = time and $\phi_{ij} = \Delta V$ as shown in Appendix.
18:     end if
19:     if $\tau_{ij} \leq T$ then
20:         Add $(i,j)$ to $A$
21:     end if
22: end for
23: Return arc set $A$.
```

in our Sub Task Creation Algorithm, Algorithm 2. We consider task v complete when any one subtask $k \in B_v$ is complete, and therefore no additional subtasks would need to be completed for this task. To complete a subtask, a servicer must arrive at node n_{vk} at time t_{vk}. In other words, a servicer must leave any node i in the network along arc (i, n_{vk}) starting at time $t_{vk} - \tau_{in_{vk}}$. To aid in the explanation of the sub task creation, we provide an example in Appendix and a graphical representation of a task moving over time is shown in Fig. 2. The arcs are orbiting arcs and the nodes are indicated by numbers. We see a task advancing over time through the network, the node with a circle indicates the task position at the time indicated. These are the same nodes and orbiting arcs shown in Fig. 1.

Algorithm 2 Sub Task Creation Algorithm

```
1: Input $|T|$ as the time horizon and network $(N, A)$.
2: Input task set $B$ where each task $v \in B$ has a known starting node
   and orbit.
3: for each task $v \in B$ do
4:   Create sub task set $B_v = \emptyset$.
5:   Set $current\_node$ = starting node.
6:   Set $current\_time = 0$.
7:   while $current\_time \leq |T|$ do
8:     Add sub task to $B_v$ associated with the ($current\_node$,
       $current\_time$).
9:     Set $next\_node$ = to the next node on the orbit in the rotational
       direction of the orbit.
10:    Set $current\_time = current\_time + \tau_{current\_node\ next\_node}$.
11:    Set $current\_node = next\_node$.
12:  end while
13: end for
14: Return $B_v$, $\forall v \in B$.
```

To complete the tasks, we consider a set of identical robot servicers, D, which move through the network along arcs $(i, j) \in A$ over time. Each servicer $d \in D$ begins the time horizon at a starting node $s_d \in N$ and we decide which first arc (s_d, j) the servicer moves along. If a task starting node equals s_d for any servicer $d \in D$ then this task is marked complete, and removed from the set of tasks, prior to solving the model. For all other tasks, the servicers must move through the network to complete the tasks. We model the movement of each robot servicer as flow throughout the network which must adhere to flow conservation (i.e., after moving on arc (s_d, j) the servicer can only move along arcs $(j, i) \in A$). Thus, over time, we determine if servicer d starts to move along arc $(i, j) \in A$ starting at time t. We assume that each servicer needs fuel to move throughout the network. Thus, over time, we track the fuel on-board of each servicer with f_{dt} where the maximum fuel each robot servicer $d \in D$ can carry is F_d. Thus, a servicer can only traverse arc (i, j) starting at time t if $f_{dt} \geq \phi_{ij}$.

Orbiting throughout the network are a set of fuel depots where servicers can refuel. Just like the tasks, the fuel depots are moving along an orbit and are located at specific locations at specific times. Refueling depots only move on orbiting arcs, thus, instead of associated refueling with node locations, we instead designate some of the orbital maneuver arcs as refueling arcs based on time. We represent this set of arcs and times indicating where and when refueling of robot servicers can occur with the set A_t^R. The triplet $(i, j, t) \in A_t^R$ indicates that a robot servicer traversing arc $(i, j) \in A$ starting at t can be refueled while traversing that arc. When traversing a refueling arc, the amount of fuel that a servicer takes on is based on the servicer and the arc length. We let R_d^F be the number of time periods for a servicer d to completely refuel to fuel level F_d. With this, we let $\mathcal{F}_{dij}$ be the maximum amount of fuel that servicer d can take on while traversing arc $(i, j, t) \in A_t^R$ between time t and $t + \tau_{ij}$. For a given refueling depot r with an associated starting node and orbit, we designate the refueling arcs in accordance with Algorithm 3. In this algorithm, we iteratively move in the rotational direction of the orbit, increment time based on the input arc τ_{ij} values, and designate refueling arcs. We can continue in this manner, creating triplicates of the form (i, j, t) stopping when $t \geq T$. We show this using pseudocode in Algorithm 3. To aid in the explanation of the refueling arc creation, we provide an example in Appendix A.4 and a graphical representation of a refueling depot moving over time and the associated arcs which become refueling arcs is shown in Fig. 2. These are the same nodes and orbiting arcs shown in Fig. 1.

Algorithm 3 Refueling Arc Designation Algorithm

```
1: Input $|T|$ as the time horizon and network $(N, A)$.
2: Input refueling depot set $R$ where each $r \in R$ has a known starting
   node and orbit.
3: Create refueling arc set $A_t^R = \emptyset$ for all $t \in T$.
4: for each depot $r \in R$ do
5:   Set $current\_node = starting\_node$.
6:   Set $next\_node$ = to the next node on the orbit in the rotational
     direction of the orbit.
7:   Set $current\_time = 0$
8:   while $current\_time \leq |T|$ do
9:     Add arc ($current\_node, next\_node$) to $A_{current\_time}^R$ as a refueling
       arc.
10:    Set $current\_time = current\_time + \tau_{current\_node\ next\_node}$.
11:    Set $current\_node = next\_node$.
12:  end while
13: end for
14: Return $A_t^R$, $\forall t \in T$.
```

3.2. Parameters, sets, and mathematical model

With the detailed problem formulation and algorithms presented, we continue with the formal definition of the decision variables, parameters and sets, and model. The decision variables, parameters, and sets used in the mathematical model were introduced at the start of this work. As we consider this finite time horizon (i.e., one day, one week), we must determine what occurs when $t \geq |T|$. We assume that robot servicers may finish at any node in the network at $|T|$. We model this assumption through the addition of a super sink node E where each node $i \in N$ is connected to E with a directed arc (i, E) with unit traversal time and no fuel requirements. With this design, in the model, we force each robot to be at node E at time $|T|$. We assume that the robot servicers return to a refueling depot after time $|T|$ and do not force this into the model, as maximizing the weighted number of completed tasks within a set time horizon is our objective. The return trip to a refueling depot after the time horizon does not contribute to task completion. We model this problem using a mixed-integer linear programming (MILP) formulation as follows.

3.2.1. Parameters/sets

T	:	Set of time periods, where $\lvert T\rvert$ is the last time period
N	:	Set of nodes , $i \in N$
E	:	Super sink node, $E \in N$
B	:	Set of tasks, $v \in B$
Γ	:	Set of refueling depots, $r \in \Gamma$
B_v	:	Set of sub tasks associated with $v \in B, k \in B_v$
A	:	Set of directed arcs, $(i, j) \in A$
A_t^R	:	Set of refueling arcs at time $t \in T, A_t^R \subset A$
D	:	Set of robot servicers, $d \in D$
s_d	:	Starting node of servicer $d \in D$
F_d	:	Maximum fuel capacity of servicer $d \in D$, in ΔV

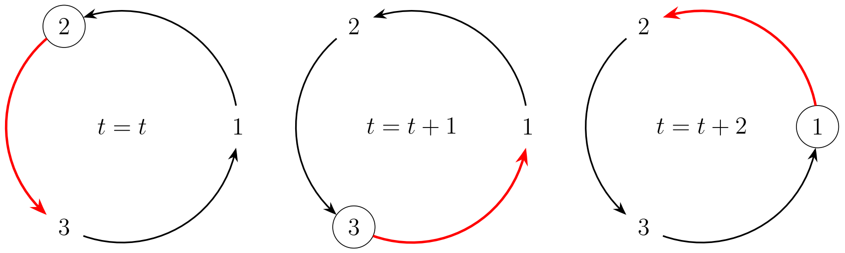

Fig. 2. Figure depicting the movement of refueling depots or tasks over time using the inner three nodes and the associated orbiting arcs from Fig. 1. Consider a task is at node 2 at time t, then at node 3 at time $t+1$, and at node 1 at time $t+2$. This task would have 3 subtasks at these nodes at these times. For this task to be completed a servicer would need to be at one of these nodes at the corresponding time. Using the same figure, we can now consider a refueling depot is at node 2 at time t, then at node 3 at time $t+1$, and at node 1 at time $t+2$. This would make the arc between nodes 2 and 3 a refueling arc at times t to $t+1$. Next the arc between nodes 3 and 1 would become a refueling arc at times $t+1$ to $t+2$ and we would proceed similarly for nodes 1 to 2 at times $t+2$ to $t+3$.

R^F_d :	The number of time periods for servicer d to completely refuel to fuel level F_d, for $d \in D$
w_v :	Weight of task $v \in B$
n_{vk} :	The node location of sub task $k \in B_v$ of task $v \in B$
t_{vk} :	Time period of sub task $k \in B_v$ of task $v \in B$
τ_{ij} :	Time periods to traverse arc $(i,j) \in A$
ϕ_{ij} :	Fuel to traverse arc $(i,j) \in A$, in ΔV
Ψ_{ij} :	Per time period fuel needed to traverse arc $(i,j) \in A$, in ΔV
$\mathcal{F}_{dij}$:	$\frac{\tau_{ij}}{R^F_d} * F_d$, The maximum amount of fuel that servicer d takes on while traversing arc $(i,j) \in A^R_t$, for $d \in D$

3.2.2. Decision variables

$$\beta_{vk} = \begin{cases} 1, & \text{if sub task } k \text{ of task } v \text{ is completed, for } v \in B \\ 0, & \text{otherwise} \end{cases}$$

f_{dt} = fuel level of robot d at time t; for $d \in D$ and $t \in T$

$$y_{dijt} = \begin{cases} 1, & \text{if robot servicer } d \text{ initiates move on arc } (i,j) \\ & \in A, \text{ at time } t \in T \\ 0, & \text{otherwise} \end{cases}$$

3.2.3. Model

$$\boxed{\max \sum_{v \in B} w_v \sum_{k \in B_v} \beta_{vk}} \tag{1}$$

$$\text{s.t.} \sum_{k \in B_v} \beta_{vk} \le 1, \qquad \text{for } v \in B \tag{2}$$

$$\sum_{d \in D} \sum_{\substack{i:(i,n_{vk}) \in A \\ t_{vk} - \tau_{in_{vk}} \ge 0}} y_{din_{vk}t_{vk}-\tau_{in_{vk}}} \ge \beta_{vk}, \qquad \text{for } v \in B, k \in B_v \tag{3}$$

$$\sum_{\substack{j:(j,i) \in A \\ t-\tau_{ji} \ge 0}} y_{djit-\tau_{ji}} - \sum_{j:(i,j) \in A} y_{dijt} = \begin{cases} -1, & \text{if } i = s_d \text{ and } t = 0 \\ 1, & \text{if } j = E \text{ and } t = |T| \\ 0, & \text{otherwise} \end{cases}$$
$$\text{for } i \in N, d \in D, t \in T \tag{4}$$

$$f_{dt} \le f_{dt-1} - \left(\sum_{(i,j) \in A} \sum_{s = \max\{t-\tau_{ij},0\}}^{t} \left(y_{dijs} * \Psi_{ij} \right) + \sum_{\substack{(i,j) \in A^R_t \\ t-\tau_{ij} \ge 0}} \left(y_{dijt-\tau_{ij}} * \mathcal{F}_{dij} \right) \right)$$
$$\text{for } t \in T \setminus |T|, d \in D \tag{5}$$

$$f_{d0} = F_d, \qquad \text{for } d \in D \tag{6}$$

$$0 \le f_{dt} \le F_d, \qquad \text{for } d \in D, t \in T \tag{7}$$

$$y_{dijt}, \quad \beta_{vk} \in \{0,1\}, \quad \text{for } d \in D, (i,j) \in A, t \in T, v \in B, k \in B_v \tag{8}$$

In Equation, (1) we present the objective function seeking to maximize the weighted number of tasks completed. In Constraints (2), we ensure that for each task, at most one sub task is completed. In Constraints (3), we link the movement of robot servicers with the completion of subtasks. To complete subtask k of task v, servicer d must depart from node i in order to arrive at node n_{vk} at time t_{vk}. To do this, servicer d must depart from node i at time $t_{vk} - \tau_{in_{vk}}$ which is the task completion time less the time cost of the arc from node i to node n_{vk}. Thus the subscripts on y ensure that servicer d moves at the correct time to complete subtask k of task v. In Constraints (4), we balance the flow of robot servicers through the network by ensuring that each robot servicer starts at their designated start node, only leave a node they are at, and must finish at the super sink E (i.e., any node in the network because each node $i \in N$ is connected to E with arcs (i,E)). As in Constraints (3), we use the subscripts on y to track where and when servicer d is moving so that flow balance is not violated. In Constraints (5), we update the fuel on-board of each servicer at each time based on the prior time period fuel level, movement, and refueling decisions. In Constraints (6), we set the starting fuel level of all robot servicers to the maximum amount. In Constraints (7), we force the fuel level of each servicer over time to be between 0 and the maximum capacity. In Constraints (8), we place the binary restriction on some decision variables.

We now present a generalized framework for using the MORSO model and the algorithms for the construction of the network arcs, subtasks, and refueling arcs. Given a set of tasks and their known orbits, choose a time horizon and determine the node locations in N. To determine the node locations in N, the nodes should be "fixed" locations in space located on orbits which intersect or which coincide with the task orbits. Next, determine the orbital parameters associated with the nodes as these will be needed to calculate the arc time and fuel costs.

With the network nodes, time horizon, and time step determined, the next step is network arc construction using Algorithm 1. The Arc Creation Algorithm should be adjusted to include the types of maneuvers which apply to the problem under consideration. In our algorithm, we only considered four maneuver types, but there are many more which can be considered. Use Algorithm 1 to construct the network arcs. For an example of this please see our detailed case study in Section 4.1.

With the network created, if the problem has refueling, the next step is to determine which and when some arcs will be available for refueling. Use the network nodes: N, the starting locations for the refueling depots, network arcs, time horizon, and the time step as inputs into Algorithm 3 to construct the sets A^R_t. We provide a detailed example of the use of this algorithm in Appendix A.4. At this point there is only one more set of data to construct, the subtask data.

To construct the subtask data, use the network nodes: N, the starting locations for the tasks, the network arcs, the time horizon, and the time step as inputs into Algorithm 2 to construct the subtask sets

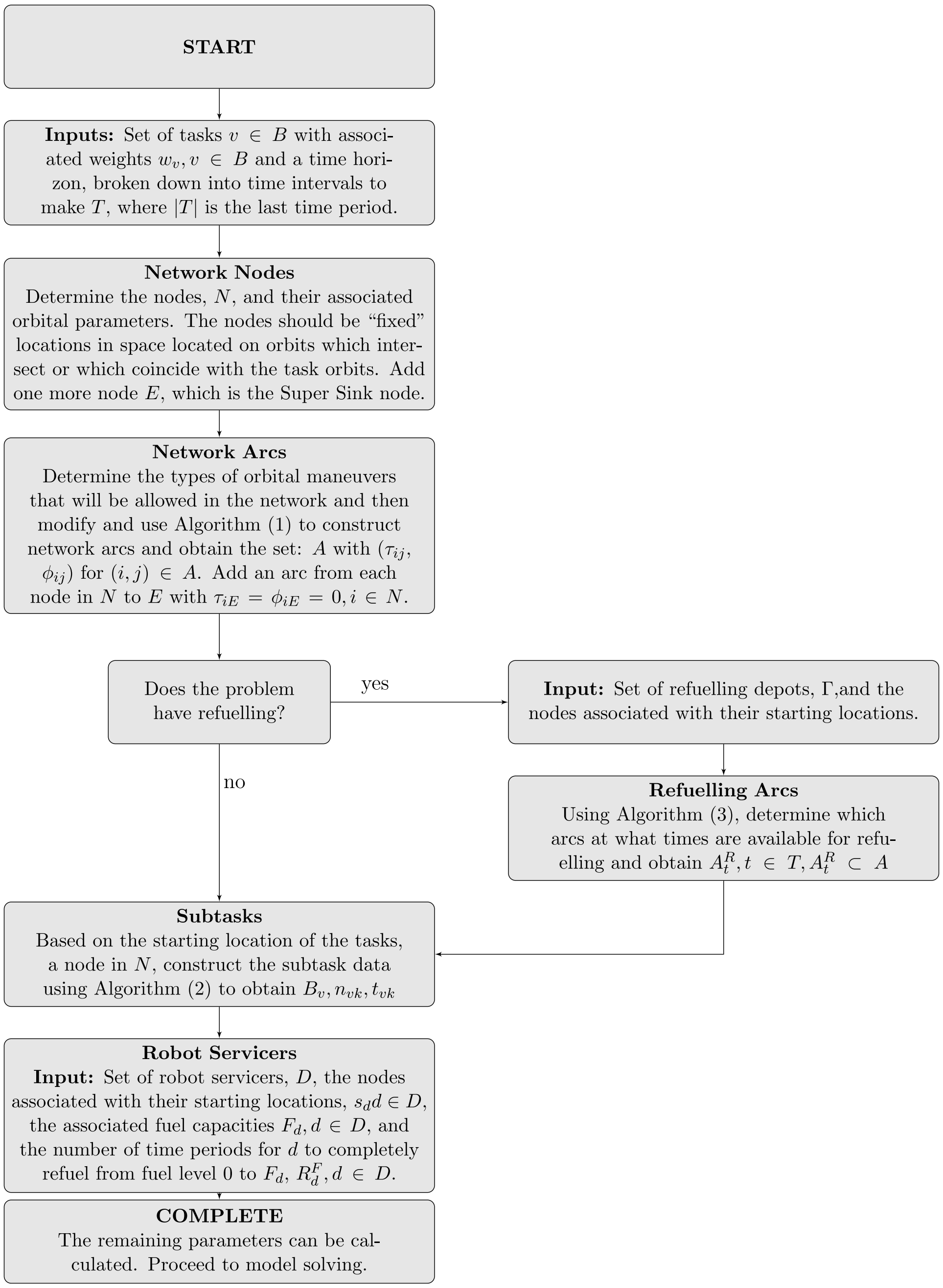
START
Inputs: Set of tasks $v \in B$ with associated weights $w_v, v \in B$ and a time horizon, broken down into time intervals to make T, where $|T|$ is the last time period.
Network Nodes
Determine the nodes, N, and their associated orbital parameters. The nodes should be "fixed" locations in space located on orbits which intersect or which coincide with the task orbits. Add one more node E, which is the Super Sink node.
Network Arcs
Determine the types of orbital maneuvers that will be allowed in the network and then modify and use Algorithm (1) to construct network arcs and obtain the set: A with (τ_{ij}, ϕ_{ij}) for $(i,j) \in A$. Add an arc from each node in N to E with $\tau_{iE} = \phi_{iE} = 0, i \in N$.
Does the problem have refuelling?
yes
no
Input: Set of refuelling depots, Γ,and the nodes associated with their starting locations.
Refuelling Arcs
Using Algorithm (3), determine which arcs at what times are available for refuelling and obtain $A_t^R, t \in T, A_t^R \subset A$
Subtasks
Based on the starting location of the tasks, a node in N, construct the subtask data using Algorithm (2) to obtain B_v, n_{vk}, t_{vk}
Robot Servicers
Input: Set of robot servicers, D, the nodes associated with their starting locations, $s_d d \in D$, the associated fuel capacities $F_d, d \in D$, and the number of time periods for d to completely refuel from fuel level 0 to F_d, $R_d^F, d \in D$.
COMPLETE
The remaining parameters can be calculated. Proceed to model solving.

Fig. 3. Flow chart depicting the steps to use the algorithms for creating the model parameters in this work.

B_v. For a detailed example of subtask creation see the case study in Section 4.2.1 or see the algorithm use example in Appendix A.4. We depict a visual summary of these steps in Fig. 3.

Next, we demonstrate the validity and benefits of the model with a case study.

4. Data and computational results

In this section, we use case studies to demonstrate and solve the Multi-Orbit Routing and Scheduling of Refuellable Space Robots for On-Orbit Servicing (MORSO) Mixed Integer Linear Program (MILP) model to gain insights about the possible strategic and operational decisions of multiple servicer on-orbit servicing (OOS) with refueling. To deduce these insights, we performed case studies using a data set we created to represent the current active satellites found in Low Earth Orbit (LEO), Mid Earth Orbit (MEO), and Geosynchronous Orbit (GEO) (Space-Track.org, 2021). Using this data set, we varied the number of refueling depots, robot servicers, the number of tasks, robot servicer maximum fuel capacity, robot servicer starting locations and network configurations. In the following, first, we describe the details of constructing the network used for our case studies. Next, we describe the details, results, and insights of our three case studies investigating general behavior, refueling, and inter orbit movements.

4.1. Network construction

For this work, we conducted three cases studies, each with similar networks in that the nodes for all three case studies were the same but the number and types of arcs were not. The network nodes were chosen based on where our notional tasks were located so that the tasks would intersect. In this section, we present an in depth breakdown of the network for the largest case study, called the *Core Case Study* and then

Table 1
Orbital parameters for the nodes used in all case studies. Nodes are needed for every orbit on which a task or refueling depot is located.

Orbit id	Nodes Start, … ,End	Semi-major axis (km)	Inclination	RAAN	Based on	Period (min)	Separation (min/deg)
1^L	$1, \dots, 6$	6652.550	52.986	290.269	Starlink	90	15.0/60.0
2^L	$7, \dots, 12$	6652.550	52.989	120.183	Starlink	90	15.0/60.0
3^L	$13, \dots, 19$	6652.550	53.000	64.290	Starlink	90	15.0/60.0
4^M	$19, \dots, 42$	26610.000	0.000	0.000	GPS	720	30.0/15.0
5^M	$43, \dots, 66$	26610.000	55.000	300.000	GPS	720	30.0/15.0
6^M	$67, \dots, 90$	26610.000	55.000	45.000	GPS	720	30.0/15.0
7^M	$91, \dots, 114$	26610.000	55.000	225.000	GPS	720	30.0/15.0
8^G	$115, \dots, 162$	42241.000	0.000	0.000	WGS	1440	30.0/7.5

[$\#^i$] Indicates the orbit ID number and orbit location where L represents Low Earth Orbit, M represents Mid Earth Orbit, and G represents Geosynchronous Orbit.

explain the differences between the networks for the two additional case studies.

To create the network for the Core Case Study, we begin by selecting the time horizon and the tasks. Our created dataset has a time horizon of 24 h and is broken into $T = 96$ distinct time units where one time unit represents 15 min. We choose the orbits for the network based on where our tasks are operating. For the cases studies presented here, we had tasks in eight different orbits which span the three orbits closest to Earth. The band closest to Earth, LEO, includes orbits with a semi-major axis of 6,558 to 8,378 kilometers. The next band called MEO, includes orbits with a semi-major axis of 8,378 to 42,158 kilometers. Finally GEO, is for objects with semi-major axes over 42,158 kilometers. Next, we describe the specifics of the nodes in each orbital band.

We considered three different circular orbits in the LEO band, denoted $1^L, 2^L$, and 3^L. Each of these orbits have a semi-major axis of 6652.55 km and a period of 90 min. We placed nodes 15 min (1 time unit) apart on the orbit, so that each orbit has 6 nodes each spaced 60 degrees apart. True anomaly indicates where on an orbit, in degrees, an object is located. Because our nodes are "fixed" locations in space and tasks and servicers move "between" nodes, we treated the true anomaly as a constant position. As an example, using orbit 1^L, the nodes are spaced 60 degrees apart, so we have 6 nodes with "fixed" true anomalies of $0°, 60°, 120°, 180°, 240°, 300°$. The inclination and RAAN values in the LEO orbits were chosen based on where SpaceX Starlink (Space-Track.org, 2021) satellites reside, as if our tasks were on the same orbits as the SpaceX Starlink satellites.

We considered four different circular orbits in the MEO band, denoted $4^M, 5^M, 6^M$, and 7^M. The nodes within each of these orbits have a semi-major axis of 26,610 kilometers and a period of 720 min or 12 h. We placed nodes 30 min (2 time units) apart so that each orbit has 24 nodes. There are 96 total nodes in MEO on four different circular orbits. The inclination and RAAN values in the MEO orbits were chosen based on where the U.S. Global Positioning System (GPS) (Space-Track.org, 2021) satellites reside, as if our tasks were located on the same orbits.

Lastly, we considered one GEO orbit, denoted 8^G which has a semi-major axis of 42,241 kilometers and a period of 1440 min, or 24 h. We placed nodes 30 min (2 time units) apart resulting in 48 nodes. The node locations on the orbit were determined as they were for LEO and MEO. The inclination and period in the GEO orbit were chosen based on where the U.S. Wideband Global SATCOM 9 (WGS) (Space-Track.org, 2021) satellite resides, as if our tasks were located on the same orbit. Fig. 4 shows a visualization of all nodes and their associated orbits relative to Earth.

In Table 1, we show the parameters of the orbits and nodes for all case studies. The first column is the orbit ID and the second column shows the nodes associated with each orbit. The middle three columns are the orbital parameters for each orbit. Eccentricity is not included because all orbits were assumed circular. However, we note that this is not a necessary assumption for the model outlined in Section 3. Instead, this is an assumption we made when creating the case study data set. The fourth column indicates on which real-world satellite the parameters were based. Finally, the last two columns show the period and spacing of the nodes on that orbit. We constructed equidistant nodes for each orbit so that the time between two nodes which are adjacent in the rotation direction of the orbit is 1 or 2 time units, 15 or 30 min (Δt), respectively. To accompany Table 1, in Fig. 4, we present a visualization of the nodes relative to Earth. With the node locations for the case study established, we proceed by explaining our arc construction.

To construct the network arcs for the Core Case Study, we considered four different types of orbital maneuvers: *Orbiting Maneuvers, Hohmann Transfers, Phasing Maneuvers,* and *Hohmann Transfers with an inclination change*. With the set of nodes (N) and the time period length (Δt) as inputs, we constructed the network arcs using Algorithm 1. For the Core Case Study, we consider $T = 24$ h. In Algorithm 1 we exclude any arcs with a traversal time $\geq T$. After following the algorithm using our set of 162 nodes, the network had $|A| = 1,381$ total directed arcs for the 24 hour time horizon. Additionally, each node is connected to a sink node for $1,381 + 162 = 1,543$ total arcs.

In Table 2, we show the arcs for the set of nodes used in our case studies. The arcs are organized by the direction and the type of maneuver. For the Core Case Study, any arcs with a time cost which exceeded the time horizon of 24 h were excluded.

In Table 2, the first column indicates the direction of the maneuvers as a change in the orbit altitude. The next column indicates the type of maneuver, we only considered four types of maneuvers in this work, but any number of maneuver types could be considered in the network construction as only the time and ΔV for each arc are needed. The next two columns indicate the starting and ending orbital bands. The fifth column, "Count" shows the number of arcs which fall into that category. The "Time" columns show the minimum, mean, and maximum traversal times for arcs in that category. The "ΔV" columns show the minimum, mean, and maximum ΔV costs for arcs in that category. The final columns show the minimum, mean, and maximum ΔV per hour costs for arcs in that category. Because the weight of the robot servicer is unknown, ΔV is used throughout the case study, but this could easily be adjusted once the weight and other properties of the chosen robot servicer are known. The time and ΔV costs shown were calculated as outlined in Curtis (2019).

As eluded to earlier, we also conducted two smaller case studies. We examined the effect of refueling in the *Refueling Case Study* and the effect of inter-orbit transfers in the *Single vs. Multi-Orbit Case Study* on the number of tasks completed. In later sections, we present the detailed factors and levels for each of these case studies, but now we will talk about the network differences.

The network for the Refueling Case Study used the same node set, N, as the Core Case Study, but used a 48 hour time horizon. Thus, any arcs with a traversal time, $\tau_{ij} \geq 48$ h were excluded. After following the algorithm using our set of 162 nodes, the network had $|A| = 3,590$ total directed arcs for the 48 hour time horizon plus the arcs connected to a sink node for $3,590 + 162 = 3,752$ total arcs. Additionally, one half of the runs allowed refueling arcs in the network while the other half did not allow refueling arcs.

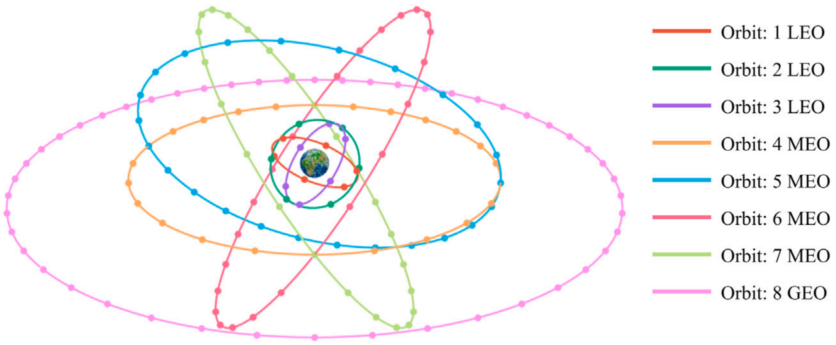

Fig. 4. Node and orbit locations relative to Earth for nodes used in this work.

Table 2
Summary of all arcs possible for nodes used in case studies.

Direction	Maneuver type	Starting orbit	Ending orbit	Count	Time (h) Min	Time (h) Mean	Time (h) Max	Δ V Min	Δ V Mean	Δ V Max	ΔV per hour Min	ΔV per hour Mean	ΔV per hour Max
Ascending	Hohmann	MEO	GEO	24	12.00	12.00	12.00	1.54	1.54	1.54	0.13	0.13	0.13
	Hohmann w/Inclination Change	LEO	MEO	72	6.00	6.00	6.00	7.42	7.64	8.24	1.24	1.27	1.37
		LEO	GEO	18	12.00	12.00	12.00	7.02	7.02	7.02	0.59	0.59	0.59
		MEO	GEO	72	12.00	12.00	12.00	4.46	4.46	4.46	0.37	0.37	0.37
Descending	Hohmann	GEO	MEO	24	6.00	6.00	6.00	1.63	1.63	1.63	0.27	0.27	0.27
	Hohmann w/Inclination Change	MEO	LEO	72	0.75	0.75	0.75	3.32	3.54	4.14	4.43	4.71	5.52
		GEO	LEO	18	0.75	0.75	0.75	2.15	2.15	2.15	2.87	2.87	2.87
		GEO	MEO	72	6.00	6.00	6.00	3.63	3.63	3.63	0.61	0.61	0.61
Same	Orbit	LEO	LEO	18	0.25	0.25	0.25	0.00	0.00	0.00	0.00	0.00	0.00
		MEO	MEO	96	0.50	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00
		GEO	GEO	48	0.50	0.50	0.50	0.00	0.00	0.00	0.00	0.00	0.00
	Phasing	LEO	LEO	75	2.00	3.74	5.00	0.74	1.83	2.39	0.37	0.49	0.54
		MEO	MEO	2116	13.00	28.54	46.00	0.10	0.84	1.29	0.01	0.03	0.03
		GEO	GEO	2209	25.00	56.55	94.00	0.04	0.65	1.04	0.00	0.01	0.01
Overall				4934	0.25	37.03	94.00	0.00	0.98	8.24	0.00	0.14	5.52

Table 3
The number of arcs for runs in the Single vs. Multi-Orbit Case Study.

Network configuration	Time horizon (h)	Number of arcs
Multi-orbit	24	1381
	36	2582
Single orbit	24	1009
	48	3218
	72	3986
	96	4934

The network for the Single vs. Multi-Orbit Case Study also used the same node set, N, as the Core Case Study, but had multiple different time horizons of $24, 36, 48, 72,$ and 96 h. In the same manner as described previously, for each run, the network was reconfigured to exclude arcs which exceeded the designated time horizon for that run. In addition, for the single orbit runs we exclude the arcs representing Hohmann transfers and Hohmann transfers with an inclination change and only considered orbiting and phasing maneuvers. All runs in the Single vs. Multi-Orbit Case Study allowed refueling. The number of arcs for the Single vs. Multi-Orbit Case Study are shown in Table 3 broken down by the network configuration and time horizon. In all case studies which allow inter orbit transfers, there are 48 Hohmann transfer arcs, 324 Hohmann transfer with an inclination change arcs, and 162 orbiting arcs.

4.2. Core case study

4.2.1. Core case study: Test design and data generation

In the previous section, we covered the network generation for each of our three case studies. With the network established, we will now focus on the Core Case Study and the details of how the data for the tasks, subtasks, and refueling arcs were created. We begin with the test design as it contains the information needed for data generation.

To generate the data for the tasks, subtasks, and refueling arcs, we need to know the number and starting location of the tasks, the number and starting locations for the refueling depots, and the number, fuel capacity, and starting location of the robot servicers. All of this information for each of the case studies is in the case study test design. For this case study, the Core Case Study, we examined the largest set of factors and levels hoping to draw big picture insights. In Table 4, we show the factors and levels used in the Core Case Study. We considered between one and five fuel depots and between one and five robot servicers. For all case studies, the refueling depot starting location, was some combination of the following: $12, 32, 80, 125, 150$. These nodes are in orbits $2^L, 4^M, 2^M,$ and 8^G. We also assumed that robot servicers always started at a fuel depot. Thus, these are also the values for s_d. When the number of robot servicers equals the number of depots, one servicer started at each depot. We enumerated all cases for one to five depots and their starting locations and this resulted in 31 cases for the combined factor of Fuel Depot Starting Location. We also considered the robot servicer fuel capacity and placed one or two tasks per orbit which resulted in 8 or 16 tasks. We continue with the generation of the data for the refueling arcs and subtask locations.

Now with the parameters for the refueling depot(s) from the test design, starting location and orbit of the refueling depots, the time horizon ($|T|$), the set of nodes (N), and the set of arcs (A) we have all the inputs to use Algorithm 3 to determine the elements of A_t^R. Fig. 5 presents a visual example of refueling depot movement which is implemented in Algorithm 3. Next we generate the subtasks from the tasks. In our set of arcs A there are 162 arcs which are orbiting

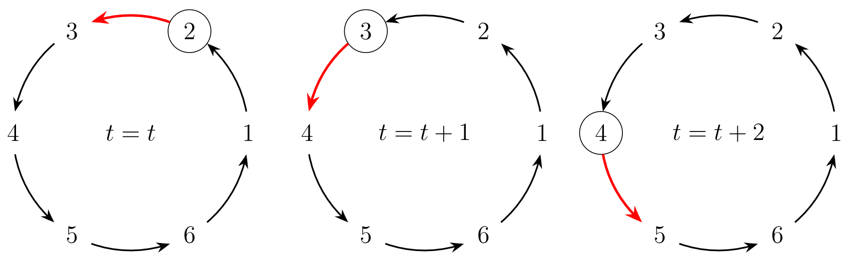

Fig. 5. Movement of refueling depots or tasks over time.

Table 4
Factors and levels for the Core Case Study.

Factor	Levels
Fuel depot starting locations	31 cases[a]
Number of robot servicers	1, 2, 3, 4, 5
Robot servicer fuel capacity (ΔV)	10, 15, 25
Number of tasks	8, 16

[a]The *Fuel depot starting locations* factor is a combination of factors for the number of servicers, number of depots, and the fuel depot starting locations.

Table 5
Tasks and their associated starting nodes for a 16 task run in the Core Case Study.

Orbit	Tasks	Starting nodes
1^L	1, 9	1, 4
2^L	2, 10	7, 10
3^L	3, 11	13, 16
4^M	4, 12	19, 31
5^M	5, 13	43, 55
6^M	6, 14	67, 79
7^M	7, 15	91, 103
8^G	8, 16	115, 139

$\#^i$ indicates the orbit ID number and orbit location where L represents Low Earth Orbit, M represents Mid Earth Orbit, and G represents Geosynchronous Orbit.

Table 6
When and where refueling arcs or sub task completion occur for Fig. 5.

Activity possible	Where	When
If the figure represents depots		
Refueling	Arc $(2,3)$	$t=t$
Refueling	Arc $(3,4)$	$t=t+1$
Refueling	Arc $(4,5)$	$t=t+2$
If the figure represents tasks		
Sub task location	Node 2	$t=t$
Sub task location	Node 3	$t=t+1$
Sub task location	Node 4	$t=t+2$

maneuvers. Depending on the number of refueling depots and the time, every arc on an orbit that has a refueling depot becomes a refueling arc.

For all case studies, we considered runs with either $|B| = 8$ or $|B| = 16$ tasks indicating one or two tasks per orbit. We numbered the tasks sequentially and assigned a unit weight to each task. The locations of tasks are moving over time, thus given the location of a task (i.e., the node closest to its starting position) at the start of the time horizon, we can approximate where and when the task is in our network at a given time. When we had two tasks on the same orbit, we started them at nodes on opposite sides of the orbit. In application, the starting node for a task should be the node closest to its location on the orbit at the start of the time period. With all of the input parameters (the time horizon ($|T|$), the set of nodes (N), the set of arcs (A), the task set (B), and the tasks ($v \in B$) with their starting nodes and orbits (see Table 5) for Algorithm 2 we generated the subtasks, ($B_v, \forall v \in B$). Fig. 5 and Table 6 present a visual and tabular example of task movement. .

In Fig. 5, we depict the movement over time of a task or fuel depot moving through an orbit between times t to $t+2$. For this figure, we assume that the task (or fuel depot) is at node 2 at time t and the nodes in this orbit are $\Delta t = 15$ min (1 time unit) apart. On the far left we depict a task's subtask at node 2 or a refueling depot at node 2 at time t resulting in a refueling arc on $(2,3)$ at time t. In the middle we see the same subtask now at node 3 at time $t+1$ or a refueling arc on $(3,2)$ at $t = t+1$. On the right, as time proceeds, the same subtask is now at node 4 or a refueling arc at time $t+2$ on $(4,5)$. In Table 6, we present a tabular representation of Fig. 5 indicating where tasks could be completed or the robot servicers might refuel, depending on if the figure represents depots or tasks.

With the network, parameters and sets created for the Core Case Study, we used the test design in Table 4 to create a full factorial design with 930 runs ($31\times5\times3\times2 = 930$). All of these runs were conducted on a High Performance Computer (HPC) using IBM Decision Optimization for CPLEX (DOcplex). IBM DOcplex is an optimization software package written for use in Python. Next we present the results and insights from the 930 runs in the Core Case Study.

4.2.2. Core case study: Results and insights

Now we present insights about solving statistics, servicer behaviors, refueling, and network configuration for the Core Case Study. In every case study, the goal of the optimization model was to maximize the weighted number of completed tasks. In all results, when we say *completed tasks*, we are actually referring to the *weighted number of completed tasks*. This is interchangeable in our case study because all of our tasks had an equal weight of one. During the analysis, we sometimes examine and refer to another response variable: *proportion of weighted tasks completed*. This proportion allows a discussion of task completion without categorizing the results based on the number of tasks, because maximizing the completed number of tasks also maximizes this proportion. For each run, we calculated this value as follows: Proportion of Weighted Tasks Completed $= \frac{\text{Objective Function Value}}{\text{Number of Tasks}}$. For the Core Case Study, this proportion varied from $0.12 - 1.00$, with an overall mean of 0.59 and a standard deviation of 0.26. This means that over all 930 runs, between $12 - 100\%$ of the tasks were completed and that on average 59% of the tasks were completed.

To further examine the response, we fit a regression model with the main effects and the two factor interactions of our design factors shown in Table 4. The most influential factors, by far, on the proportion of weighted of tasks completed are the servicer starting fuel and the number of servicers, accounting for over 80% of the variability in the response. The completed tasks (objective values) results are shown in Table 7 grouped by the number of servicers and the fuel capacity as these were the two most influential factors.

Next, we will cover the solving statistics for the Core Case Study. We begin with a brief explanation of terminology and then proceed with the details of the solving statistics. The goal when solving any optimization model to find an *optimal* solution. For our problem, this translates to finding a solution for which the servicers complete as many tasks

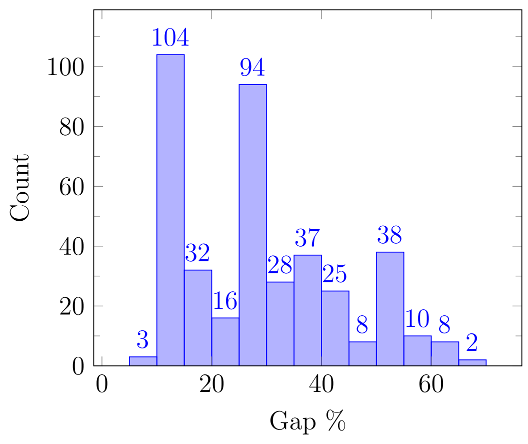

Fig. 6. Histogram of Gap Values.

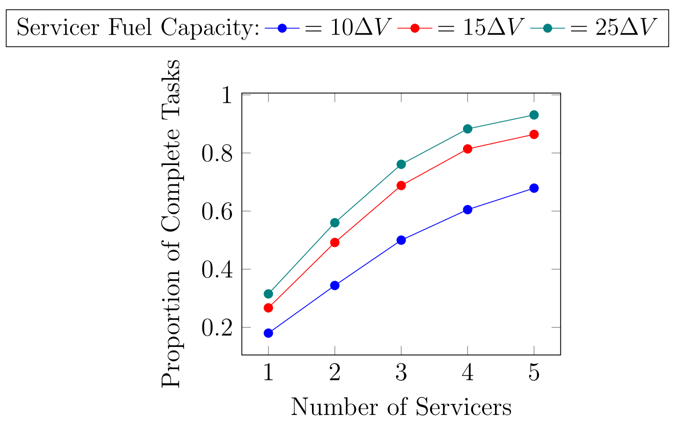

Fig. 7. How the number and fuel capacity of servicers affected task completion.

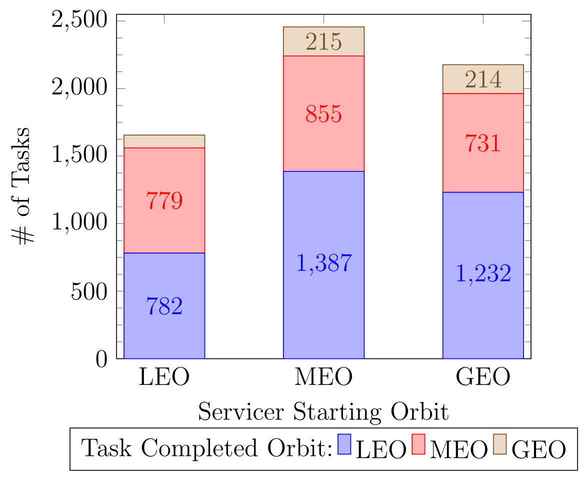

Fig. 8. Where tasks were completed versus the starting orbit of the completing servicer.

as possible. When solving our problem, the software stopped when it found an integer optimal solution to the problem, reached the time limit, or ran out of memory.

Of the 930 runs, 523 reached integer optimality. When solving, we set a 12 hour solving time limit for each of the 930 runs. If the solver could not find an optimal solution before the time limit was reached, the software reported the best objective function value at that time along with the *gap*. Given an optimal solution, the gap is a value which indicates that the current solution is off by no more than this percentage. Of the 930 runs, 404 reached the 12 hour time limit with an average gap of 28.4% and a range of 6.25 – 69.1%. This means that the solutions obtained at the end of the 12 h, in the worst case, differed from an optimal solution by 6.25 – 69.1%. A visualization of the gap distribution for these runs is shown in Fig. 6. Finally, there were 3 runs which ran out of memory on the high performance computer and failed to solve. The HPC has another partition which is designated *high-memory*. We ran the 3 runs with a 12 hour time limit on the *high-memory* partition and they all reached the 12 hour time limit. The combined solving results for all runs are shown in Table 7.

Each row of the table summarizes the outcome of the 31 runs for that set of factor levels, with the exception of the runs which ran out of memory and did not solve on the HPC. For the factor combinations which ran out of memory, those rows are marked with a number in the exponent which indicates the number of runs which ran out memory. The column *Mean 12 Hour Gap (%)* shows the averaged gap for each row. When the value in this column is zero, this means that all 31 runs resulted in an integer optimal solution. When the value in this column is not zero, this means that at least one of the 31 reached the 12 h time limit when solving. Each row shows the counts of the runs which were integer optimal or reached the time limit of 12 h. The *Mean 12 Hour Gap (%)* includes zero values in the average for those runs that reached integer optimality. This same logic follows for the *Mean Solve Time (hours)* column. For the rows which had runs which reached the 12 hour time limit, the solve time mean includes 12 h in the average.

In addition to solving insights, we can see how the number and configuration of the robot servicers impacts the number of completed tasks using the Objective Value columns. As would be expected, as the number of servicers increases so does the number of completed tasks. We examined this further using Fig. 7 and can see that the slopes of the lines decrease as the number of servicers increase indicating a diminishing return on the number of completed tasks. Fig. 7 also illustrates the decreasing impact of increasing the servicer fuel capacity. The increase in proportion of completed tasks when increasing the servicer fuel capacity from 10 to 15 is almost twice that of the increase when the fuel capacity increases from 15 to 25, illustrating the decreasing impact of increasing the servicer fuel capacity.

In addition to solving statistics and the impact of the factors on the response variable, we also examined robot servicer movements. In the 930 runs of the main test design, there were 2,790 total servicers and 11,160 tasks. On average, each servicer completed 2.3 tasks, refueled 2.8 times, and used 13.9 ΔV. The number of tasks completed by servicers ranged from 0−4. There were 3 servicers that completed no tasks and all of these cases occurred when there were 5 servicers available. These results are intuitive because when there are less servicers, the servicers available must complete more tasks. No servicer completed more than 4 tasks in this 24-hour time horizon indicating that it might be better to have more than one servicer when then are more than four tasks to be completed.

Another interesting observation from this case study is that where servicers started did not influence where they completed tasks. We examined the orbits where tasks were completed versus where the completing servicer started. Fig. 8 shows where tasks were completed as compared to where the completing servicer started and Table 8 shows a tabular representation. The plot shows the results for all completed tasks in the Core Case Study. The colors indicate the proportion of all tasks which were completed in that orbit. We know from Table 2 that the maneuvers to and within LEO use much less time allowing the servicers to move more and perhaps complete more tasks. All servicers completed the most tasks in LEO, followed by MEO and GEO and this was true across all servicer starting orbits indicating that no matter where a servicer starts, they proceed to lower orbits to accomplish tasks. The results here motivated the third case study which focused on the impact of inter orbit transfers.

Table 7
Solving results for the integer optimal and time limit reached runs for the Core Case Study.

Factors			Using CPLEX 12.10				Objective value		
Number of tasks	Number of servicers	Fuel capacity	Integer optimal solution	Reached 12 h	Mean 12 hr Gap (%)	Mean Solve time (h)	Min	Mean	Max
8	1	10	31	0	0.00	0.16	1	1.61	2
	1	15	31	0	0.00	0.12	2	2.48	3
	1	25	31	0	0.00	0.05	3	3.00	3
	2	10	13	18	28.89	8.19	2	3.06	4
	2	15	27	4	3.38	4.31	4	4.61	5
	2	25	24	7	5.00	4.16	5	5.00	5
	3	10	1	30	41.86	11.79	3	4.52	6
	3	15	5	26	16.83	10.95	6	6.32	7
	3	25	8	23	9.27	10.58	6	6.90	7
	4	10	1	30	28.23	11.63	4	5.71	7
	4	15	18	13	5.24	5.64	7	7.52	8
	4	25	29	2	0.81	1.14	7	7.90	8
	5	10	7	24	17.34	10.82	5	6.58	8
	5	15	31	0	0.00	0.70	7	7.97	8
	5	25	31	0	0.00	0.34	7	7.97	8
16	1	10	31	0	0.00	0.05	2	2.58	3
	1	15	31	0	0.00	0.04	3	3.52	4
	1	25	31	0	0.00	0.03	4	4.00	4
	2	10	16	13[2]	16.60	8.02	4	4.84	6
	2	15	27	4	1.70	3.72	6	6.52	7
	2	25	30	1	0.36	0.72	8	8.00	8
	3	10	2	28[1]	24.74	11.35	5	6.94	9
	3	15	25	6	4.56	4.35	9	9.39	10
	3	25	28	3	2.25	2.71	9	10.48	12
	4	10	3	28	40.98	11.74	6	7.94	10
	4	15	6	25	22.98	10.14	10	11.00	12
	4	25	5	26	16.98	10.60	10	12.45	14
	5	10	0	31	45.98	12.00	6	8.58	11
	5	15	0	31	26.65	12.00	9	11.71	13
	5	25	0	31	13.52	12.00	11	13.81	15

# indicates the number of runs which ran out of memory.

Table 8
Where tasks were completed versus the starting orbit of the completing servicer.

Refueling orbit	Servicer starting orbit		
	LEO	MEO	GEO
LEO	782	1387	1232
MEO	779	855	731
GEO	95	215	214

Table 9
Where servicers refueled versus their starting location (starting node).

Refueling orbit	Servicer starting orbit		
	LEO	MEO	GEO
LEO	629	76	12
MEO	5	1209	1
GEO	0	0	741

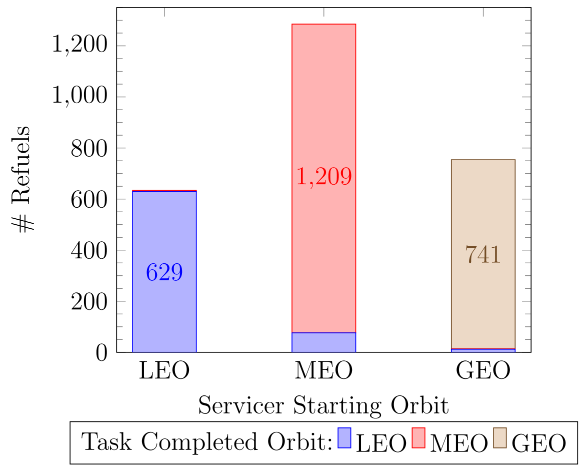

Fig. 9. Where servicers refueled versus their starting location (starting node).

Although the servicers preferred to work in LEO, they refueled most often in the orbits where they started. In the Core Case Study, there were 2673 refueling events. A refueling event occurred when the fuel level of a servicer increased from one time period to the next while traveling on a refueling arc. As shown in Section 3, in our model, there is no penalty for refueling, meaning that servicers can refuel as many times as they want. Servicers refueled on average 1.93 times. Fig. 9 shows where servicers started versus where they refueled and Table 9 shows a tabular representation of the same data. The distribution of the number of refueling events by the servicer fuel capacity is shown in Table 10. As would be expected, servicers with a smaller fuel capacity refueled more often. These results motivated the second case study which focused on refueling. In this section, we have shown a few examples of insights to be gained when using this model to solve a robot servicer scheduling and routing problem. Next, we present the details of the Refueling Case Study.

4.3. Refueling case study

Following the Core Case Study, we completed a smaller, more focused test design to determine how refueling affects the number of

Table 10
Distribution of refueling events by servicer starting fuel.

Servicer fuel capacity (ΔV)	Number of refueling events
25	498
15	883
10	1292
Total	2673

Table 11
Factors and levels for the focused Refueling Case Study.

Factor	Levels
Number of fuel depots	0, 5[a]
Number of robot servicers	1
Robot servicer fuel capacity	10, 15, 25
Servicer starting node	12-LEO, 32-MEO, 80-MEO, 125-GEO, 150-GEO
Number of tasks	8
Time Horizon (h)	48
Solving time limit (h)	72

[a]No runs without any fuel depots were accomplished in the original data set.

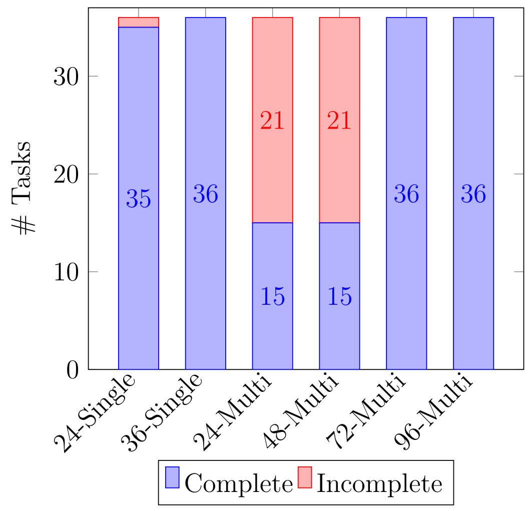

Fig. 10. Task completion by network configuration and time horizon.

weighted tasks completed. We proceed in the section with the test design and then present the results.

The test design for the Refueling Case Study was a full factorial design of 30 runs based on the factors and levels in Table 11. Half of the runs had 5 refueling depots and the other half had none. We also examined a longer time horizon than the main test design, using 48 h, with the hypothesis that refueling becomes more influential in a longer time horizon. Finally, because a longer time horizon makes the problem more difficult to solve, we extended the solving time limit to 72 h.

The results of the Refueling Case study indicated that the presence of refueling depots enabled the completion of more tasks, however, not as significantly as expected. Servicers, on average, completed on average $1-2$ more tasks when refueling depots were available. It is likely these results would have been more significant if solving difficulties were not present. Even with the longer solving time, only 19 of the 30 runs solved to integer optimality. None of the runs with refueling depots and the smallest fuel capacity of $10\Delta V$ solved to optimality within the 72 h time limit. All of the runs without refueling depots reached integer optimality within the 72 h time period. Thus, the $1-2$ task improvement with refueling depots is a lower bound because the true optimal solutions could be up to 39% improved in the runs which did not reach integer optimality. In Table 12, we show the results for all 30 grouped by the servicer fuel capacity and servicer starting orbit. As would be expected, the impact of refueling depots on task completion is most evident when the servicer fuel capacity is lower. Although on orbit refueling depots do not yet exist, these results show that refueling is necessary to fully enable the long term operation of robot servicers. In the next case study, we examine the effect that the inter-orbit transfers have on the weighted number of tasks completed.

4.4. Single versus multi-orbit case study

During the Core Case Study, we observed that when tasks were completed, they were most often completed by servicers which started in a different orbit (see Fig. 8). This observation combined with the multi-orbit nature of our problem motivated the creation and execution of a smaller case study focused on determining the difference in the weighted number of completed tasks when inter-orbit maneuvers are and are not allowed.

This section summarizes the details and results of the Single Versus Multi-Orbit (SVM) Case Study. We begin with the test design and then present the results and insights.

We call the network which allows inter orbit transfers a *multi-orbit network* and the network without inter orbit transfers a *single orbit network*. We show the SVM Case Study factors and levels in Table 13. We completed 36 runs using these factors and levels. We did not include time horizons greater than 36 hours for the multi-orbit network configuration due to solving issues observed in the Core Case Study. In contrast to the Core Case Study, we examined multiple different time horizons and only included runs with 5 servicers and 5 refueling depots. In the comparisons, we included tasks on orbits which had a servicer and excluded all tasks which would be unreachable without inter orbit transfers reducing the number of tasks to 4 or 8. To create the single orbit networks, we followed the arc creation algorithm as shown in Algorithm 1 for each time horizon, and then removed all arcs which allowed servicers to move between orbits or exceeded the time horizon.

The mean solving time for the runs with a single orbit network configuration was less than 13 min, while the mean solving time for runs with the multi-orbit network configuration was nearly 11 h. The results of the runs are shown in Fig. 10. Each bar shows the results from 36 possible tasks for that network configuration/time horizon combination.

For our network, the effect of the multi-orbit network is significant. Servicers operating on the multi-orbit network completed nearly as many tasks in 24 h as those operating on the multi-orbit network in 72 h. These results would likely translate to similar results with larger networks and longer time horizons which could justify using multi-orbit servicers at some point in the future as space technologies advance.

5. Conclusions

We are the first to consider the Multi-Orbit Routing and Scheduling of Refuellable On-Orbit Servicing Space Robots (MORSO) problem. We present a novel Mixed Integer Linear Program formulation seeking to maximize the weighted number of tasks completed within a given time horizon. We created new algorithms to model the movement of tasks and refueling depots over time and to generate the network data based on time and ΔV costs which come from orbital mechanics calculations. We performed computational tests using data created based on satellites which are currently on-orbit in LEO, MEO, and GEO and provide managerial insights which inform decision makers on the number and configuration of robot servicers needed for a set of tasks. We also demonstrated the benefit of on orbit refueling depots and the importance of inter-orbit transfers for task completion.

To validate MORSO, we completed three case studies using our network and were able to provide results concerning where robot servicers were completing tasks and how refueling depots and servicer

Table 12
How refueling depots affected task completion.

Servicer	Servicer	Without refueling		With refueling	
starting orbit	fuel capacity	Complete	Incomplete	Complete	Incomplete
LEO	10	1	7	3	5
	15	3	5	5	3
	25	5	3	6	2
MEO	10	5	11	7	9
	15	6	10	9	7
	25	10	6	10	6
GEO	10	6	10	7	9
	15	6	10	10	6
	25	10	6	11	5

Table 13
Factors and levels for the focused Single vs. Multi-Orbit Case Study.

Factor	Levels
Number of robot servicers	5
Robot servicer fuel capacity	10, 15, 25
Number of tasks	4,8
Network configuration	Multi-orbit, Single orbit
Time horizon (h)	24, 36, 48, 72

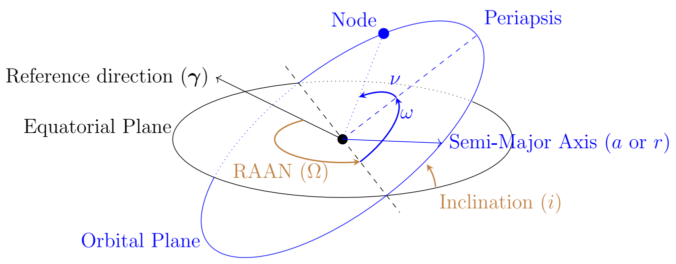

Fig. A.11. Orbital parameters.

starting locations influence tasks completions. These results translate into policy insights for the numbers and locations of both robot servicers and refueling depots. We also present analysis showing how network configuration impacts the types and numbers of movements that robot servicers make. Finally, we presented the significant impact of a multi versus single orbit network on task completions. These results are based on our network, however similar types of insights would be possible from larger network configurations, with more detailed maneuver costs. Incorporating additional maneuver types, precise rendezvous time and ΔV costs, and incorporating task times are definite possibilities for future research.

Another facet of this work is the initial evidence surrounding the solving difficulty of MORSO when constructed with larger networks, more servicers, more refueling depots, and longer time horizons. The solving difficulties encountered with even 24 hour time horizon empirically demonstrate that the MORSO may be hard to solve. To prove the MORSO is hard to solve is the first direction for follow-on research related to this work.

Although the technology for the MORSO does not yet exist, there are many additional areas of research such as the development of a heuristic to obtain optimal or near optimal solutions to the MORSO more quickly. With a heuristic, much longer time horizons and much larger networks could be investigated. Further research could also include examining the MORSO as a multi-objective optimization maximizing the number of tasks completed while minimizing time and ΔV expenditures. This work has demonstrated that this model can be used to effectively optimize the scheduling and routing of robot servicers to accomplish on-orbit servicing tasks across many orbits and presents algorithms which make it robust to larger and more complex network considerations.

CRediT authorship contribution statement

Susan E. Sorenson: Conceptualization, Methodology, Software, Writing – original draft. **Sarah G. Nurre Pinkley:** Supervision, Conceptualization, Methodology, Writing – review & editing.

Data availability

Data will be made available on request.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix

A.1. Orbital mechanics background: Parameters and constants

These orbital parameters and constants were used in the creation of the data sets in this work.

a or r: semi-major axis, distance from the center the Earth in km; orbital altitude
ν or θ: true anomaly, degrees, where on the "orbit" the node is located
i: inclination, degrees, the vertical "tilt" of the orbit
e: eccentricity, the "roundness" of the orbit
Ω: right ascension of ascending node (RAAN), degrees, the horizontal "tilt" of the orbit
ω: perigee (angle to periapsis, the location on the orbit that is closest to Earth)
μ: Earth's gravitational parameter in $\mathrm{km}^3/\mathrm{s}^s$ 398,600

NOTE: Circular orbits have $e = 0$ and do not have ω, because all points on the orbit have the same distance to Earth.(see Fig. A.11).

The period (the time) of a circular orbit is calculated as follows: $T = \frac{2\pi r}{\sqrt{\frac{\mu}{r}}}$, in seconds, where r is the semi-major axis of the orbit.

A.2. Orbital mechanics background: Hohmann transfer

A Hohmann transfer is used to make a change in orbital altitude, r on the same orbital plane. The arrival location on the destination orbit is at a position of 180 degrees forward or backward (ν), depending on where maneuver was initiated. The calculations for a change in altitude from circular orbit r_1 to the co-planar orbit r_2 are as follows (Curtis, 2019):

Table A.14
All possible arcs from node $i = 3$ with Maneuver types, Time and Fuel costs.

i	j	Maneuver type	Time cost τ_{ij}	ΔV (Fuel cost) ϕ_{ij}	Fuel per time period Ψ_{ij}
3	1	Phasing	17	2.09	0.12
3	2	Phasing	20	2.39	0.12
3	4	Orbit	1	0.00	0.00
3	5	Phasing	11	1.30	0.12
3	6	Phasing	13	1.74	0.13
3	39	Hohmann+incline	24	8.24	0.34
3	63	Hohmann+incline	24	7.44	0.31
3	87	Hohmann+incline	24	7.44	0.31
3	111	Hohmann+incline	24	7.44	0.31
3	155	Hohmann+incline	48	7.02	0.15

Table A.15
MORSO Core Case Study: Sub tasks for task 4.

Task	Sub task	Start time	End time	Weight
4	19	1	1	1
4	20	3	3	1
4	21	5	5	1
4	22	7	7	1
4	23	9	9	1
4	24	11	11	1
4	25	13	13	1
4	⋮	⋮	⋮	⋮
4	40	91	91	1
4	41	93	93	1
4	42	95	95	1

The angular momentum of the starting orbit is:

$$h_1 = \sqrt{2 * \mu}\sqrt{\frac{r_1 * r_1}{r_1 + r_1}} \text{ km}^2/\text{s} \tag{A.1}$$

The angular momentum of the transfer orbit is:

$$h_2 = \sqrt{2 * \mu}\sqrt{\frac{r_1 * r_2}{r_1 + r_2}} \text{ km}^2/\text{s} \tag{A.2}$$

The angular momentum of the destination orbit is:

$$h_3 = \sqrt{\mu r_2} \text{ km}^2/\text{s} \tag{A.3}$$

The velocity at the starting orbit is:

$$V_1 = \frac{\sqrt{2 * \mu}\sqrt{\frac{r_1 * r_1}{r_1 + r_1}}}{r_1} \text{ km/s} \tag{A.4}$$

The velocity at the transfer orbit is:

$$V_2 = \frac{\sqrt{2 * \mu}\sqrt{\frac{r_1 * r_2}{r_1 + r_2}}}{r_1} \text{ km/s} \tag{A.5}$$

The velocity at the destination orbit is:

$$V_3 = \frac{\sqrt{\mu r_2}}{r_2} \text{ km/s} \tag{A.6}$$

The ΔV for the maneuver is:

$$\begin{aligned} \Delta v_1 &= V_2 - V_1 \\ \Delta v_2 &= V_3 - V_2 \\ \Delta V &= |v_1| + |v_2| \end{aligned} \tag{A.7}$$

The time for the maneuver is:

$$\frac{T}{2} = \frac{2\pi r_2^{\frac{3}{2}}}{2\sqrt{\mu}} \tag{A.8}$$

A.2.1. Orbital mechanics background: Combined hohmann transfer with inclination change

This maneuver combines a change from r_1 to r_2 (altitude) and a change in i (inclination). Let the starting orbit (i) have parameters r_1 and i_1 and the destination orbit (j) have parameters r_2 and i_2. The calculations are as follows (Curtis, 2019):

The velocity at the starting orbit, i:

$$v_{i_1} = \sqrt{\frac{\mu}{r_1}} \text{ km/s} \tag{A.9}$$

The angular momentum at the destination orbit, j:

$$h_j = \sqrt{2\mu}\sqrt{\frac{r_1 * r_2}{r_1 + r_2}} \text{ km/s} \tag{A.10}$$

Intermediate calculations:

$$v_{i_2} = \frac{h_j}{r_1} \tag{A.11}$$

$$v_{j_2} = \frac{h_j}{r_2} \tag{A.12}$$

$$v_{j_3} = \sqrt{\frac{\mu}{r_2}} \tag{A.13}$$

The change in inclination can be done either at the starting orbit, the destination orbit or half and half. The lower the altitude, the more expensive the ΔV for the inclination change (Curtis, 2019). At the destination:

$$\begin{aligned} \Delta v_j &= \sqrt{v_{j_2}^2 + v_{j_3}^2 - 2 * v_{j_2} * v_{j_3} * \cos \Delta i} \\ &= \sqrt{\left(\frac{h_j}{r_2}\right)^2 + \left(\frac{\mu}{r_2}\right) - 2 * \frac{h_j}{r_2} * \sqrt{\frac{\mu}{r_2}} * \cos \Delta i} \end{aligned} \tag{A.14}$$

$$\Delta v_i = v_{i_2} - v_{i_1} = \frac{h_j}{r_1} - \sqrt{\frac{\mu}{r_1}} \tag{A.15}$$

$$\Delta V = \Delta v_i + \Delta v_j \tag{A.16}$$

At the origin:

$$\begin{aligned} \Delta v_i &= \sqrt{v_{i_1}^2 + v_{i_2}^2 - 2 * v_{i_2} * v_{i_1} * \cos \Delta i} \\ &= \sqrt{\frac{\mu}{r_1} + \left(\frac{h_j}{r_1}\right)^2 - 2 * \sqrt{\frac{\mu}{r_1}} * \frac{h_j}{r_1} * \cos \Delta i} \end{aligned} \tag{A.17}$$

$$\Delta v_j = v_{j_3} - v_{j_2} = \sqrt{\frac{\mu}{r_2}} - \frac{h_j}{r_2} \tag{A.18}$$

$$\Delta V = \Delta v_i + \Delta v_j \tag{A.19}$$

We calculated the costs for accomplishing the inclination change at both the starting orbit and the destination orbit and the result with the smaller ΔV.

A.2.2. Orbital mechanics background: Phasing maneuver

A phasing maneuver is used for a change in the *true anomaly*. Let $\Delta\phi$ = the change in *true anomaly* angle between nodes i and j in radians. We can calculate the costs for a phasing maneuver as follows (Curtis, 2019):

$$\begin{cases} \theta_j - \theta_i < 0 & \Delta\phi = 360 + \theta_j - \theta_i \frac{\pi}{180} \\ \theta_j - \theta_i \geq 0 & \Delta\phi = \theta_j - \theta_i \frac{\pi}{180} \end{cases} \tag{A.20}$$

Table A.16
Example: Refueling arcs for depot starting at node 12.

From i	To j	Refueling time periods: t
12	7	0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90
7	8	1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91
8	9	2, 8, 14, 20, 26, 32, 38, 44, 50, 56, 62, 68, 74, 80, 86, 92
9	10	3, 9, 15, 21, 27, 33, 39, 45, 51, 57, 63, 69, 75, 81, 87, 93
10	11	4, 10, 16, 22, 28, 34, 40, 46, 52, 58, 64, 70, 76, 82, 88, 94
11	12	5, 11, 17, 23, 29, 35, 41, 47, 53, 59, 65, 71, 77, 83, 89, 95

The period of the starting orbit is:

$$T_s = \frac{2\pi r}{\sqrt{\frac{\mu}{r}}} \text{ seconds} \tag{A.21}$$

The mean motion of the satellite in its circular orbit is:

$$n = \sqrt{\frac{\mu}{r^3}} \tag{A.22}$$

k is the number of revolutions which the satellite performs in the phasing ellipse, here in this work we assume $k = 1$. The minimum k is 1. As k increases, the time increases, and the ΔV for the maneuver decreases. So then the phasing ellipse orbital period in seconds is Curtis (2019):

$$T_{\text{phasing ellipse}} = \frac{2k\pi + \Delta\phi}{kn} \tag{A.23}$$

And the semi major axis of the phasing ellipse is:

$$a(k,n,r)_{\text{phasing ellipse}} = \left(\frac{\mu \left(\frac{2k\pi+\Delta\phi}{kn} \right)^2}{4\pi^2} \right)^{\frac{1}{3}} \tag{A.24}$$

The ΔV to start and stop the maneuver is equal so the total ΔV is:

$$\Delta V = 2 * \left| \sqrt{\frac{2\mu}{r} - \frac{\mu}{a(k)_{\text{phasing ellipse}}}} - \sqrt{\frac{\mu}{r}} \right| \tag{A.25}$$

The time to complete the phasing maneuver is:

$$t_{\Delta\phi} = T_{\text{phasing ellipse}} * \frac{\Delta\phi}{2\pi} + k * T_{\text{phasing ellipse}} \tag{A.26}$$

A.3. Demonstration of algorithms presented in this work

A.3.1. Arc creation algorithm example

To aid the in the explanation of arc creation we provide a specific example as follows. Consider node $i = 3$ in orbit 1. Node 3 has a stationary true anomaly of 120°, a semi-major axis of 6652.55 (km), and an inclination of 52.986. For node $i = 3$ there are 10 arcs possible. Table A.14 shows the nodes that node $i = 3$ is connected to in column j. The third column indicates the type of maneuver. Node $i = 3$ is adjacent in the rotation direction of the orbit to node $j = 4$ on the same orbit therefore the arc $(3,4)$ between these nodes is an orbiting maneuver arc. Node $i = 3$ can make a phasing maneuver to any other node on the same orbit which is not adjacent in the rotation direction of the orbit, therefore node $i = 3$ is connected to nodes $j = 1,2,5,6$ by *phasing maneuver* arcs. Node $i = 3$ is connected to orbit 4 on the arc $(3,39)$, orbit 5 on the arc $(3,63)$, orbit 6 on the arc $(3,87)$, orbit 7 on arc $(3,111)$ and to orbit 8 on arc $(3,155)$ using *Hohmann transfer with inclination change* arcs.

A.4. Sub task creation algorithm example

An example of the subtasks for Task 4 is shown in Table A.15 below.

A.4.1. Refueling arc designation algorithm example

An example of the refueling arcs for a depot starting at node 12 is shown in Table A.16. The scenarios in this case study have 1 to 5 robot servicers available to accomplish tasks. The robots start the overall time period at one of the refueling depots. The movements of the robot servicers are controlled with the decision variable y_{dijt} as and their fuel level is controlled with the decision variable f_{dt}. Like the tasks, the fuel depots are also in orbit and so for a particular arc in a refueling orbit, we have times when that depot is available for refueling. For all case studies, we have 162 orbiting maneuver arcs. Thus, for a given depot, we can determine if and when each of these arcs is available to refuel robot servicers. Table A.16 shows on which arcs at which times that fuel depot 12 is available for refueling. Thus if a robot servicer needs to refuel, it would need to move on of these arcs at one of the times listed as these are the $(i,j,t) \in A_t^R$ for depot 12.

References

Alfriend, K. T., Lee, D. J., & Creamer, N. G. (2006). Optimal servicing of geosynchronous satellites. *Journal of Guidance, Control, and Dynamics*, *29*(1), 203–206.

Bourjolly, J., Gürtuna, Ö., & Lyngvi, A. (2006). On orbit servicing: A time dependent, moving target traveling salesman problem. *International Transactions in Operational Research*, *13*(5), 461–481.

British Broadcasting Corporation (2020). Space news: First ever space ‘petrol station’ to be built in UK. https://www.bbc.co.uk/newsround/54551557. Last Accessed: March 21, 2022.

Corbin, B. A., Abdurezzak, A., Newell, L. P., Roesler, G. M., & Lal, B. (2020). Global trends in on-orbit servicing, assembly and manufacturing (OSAM). IDA Document D-13161. https://www.ida.org/research-and-publications/publications/all/g/gl/global-trends-in-on-orbit-servicing-assembly-and-manufacturing-osam.

Curtis, H. (2019). *Orbital mechanics for engineering students*. San Diego: Elsevier Science & Technology.

Daneshjou, K., Mohammadi-Dehabadi, A. A., & Bakhtiari, M. (2017). Mission planning for on-orbit servicing through multiple servicing satellites: A new approach. *Advances in Space Research*, *60*(6), 1148–1162. http://dx.doi.org/10.1016/j.asr.2017.05.037.

Du, B., Zhao, Y., Dutta, A., Yu, J., & Chen, X. (2015). Optimal scheduling of multi-spacecraft refueling based on cooperative Maneuver. *Advances in Space Research*, *55*(12), 2808–2819. http://dx.doi.org/10.1016/j.asr.2015.02.025.

Dutta, A., & Tsiotras, P. (2007). A greedy random adaptive search procedure for optimal scheduling of P2P satellite refueling. In *AAS/AIAA space flight mechanics meeting* (pp. 07–150).

Dutta, A., & Tsiotras, P. (2008). A cooperative P2P refueling strategy for circular satellite constellations. In *AIAA SPACE 2008 conference & exposition* (p. 7643).

Dutta, A., & Tsiotras, P. (2010). Network flow formulation for cooperative peer-to-peer refueling strategies. *Journal of Guidance, Control, and Dynamics*, *33*(5), 1539–1549. http://dx.doi.org/10.2514/1.45570.

ESA (2018). Removing debris to demonstrate commercial in-orbit servicing. https://blogs.esa.int/cleanspace/2018/09/06/removing-a-debris-to-demonstrate-commercial-in-orbit-servicing/. Last Accessed: March 21, 2022.

Gürtuna, Ö., & Trépanier, J. (2003). On-orbit satellite servicing: A space-based vehicle on-orbit servicing routing problem. In *Operations research in space and air* (pp. 123–141). Springer, http://dx.doi.org/10.1007/978-1-4757-3752-3.

Hudson, J. S., & Kolosa, D. (2020). Versatile on-orbit servicing mission design in geosynchronous earth orbit. *Journal of Spacecraft and Rockets*, *57*(4), 844–850. http://dx.doi.org/10.2514/1.A34701.

Li, W. J., Cheng, D. Y., Liu, X. G., Wang, Y. B., Shi, W. H., Tang, Z. X., Gao, F., Zeng, F. M., Chai, H. Y., Luo, W.-B., Cong, Q., & Gao, Z. L. (2019). On-orbit service (OOS) of spacecraft: A review of engineering developments. *Progress in Aerospace Sciences*, *108*, 32–120. http://dx.doi.org/10.1016/j.paerosci.2019.01.004.

NASA (2020). NASA selects proposals to demonstrate in-space refueling and propellant Depot Tech. https://www.nasa.gov/directorates/spacetech/solicitations/tipping_points/2020_selections. Last Accessed: March 21, 2022.

NASA (2021). On-orbit servicing, assembly, and manufacturing 1 (OSAM-1). https://nexis.gsfc.nasa.gov/osam-1.html. Last Accessed: March 21, 2022.

NASA (2021a). About - hubble servicing missions. https://www.nasa.gov/mission_pages/hubble/servicing/index.html. Last Accessed: March 21, 2022.

Northrup Grumman (2020). Mission extension vehicle. https://news.northropgrumman.com/news/releases/northrop-grumman-and-intelsat-make-history-with-docking-of-second-mission-extension-vehicle-to-extend-life-of-satellite. Last Accessed: March 21, 2022.

Sarton du Jonchay, T., Chen, H., Gunasekara, O., & Ho, K. (2020). Rolling horizon optimization framework for the scheduling of on-orbit servicing operations under servicing demand uncertainties. In *ASCEND 2020* (p. 4131). American Institute of Aeronautics and Astronautics, Inc, http://dx.doi.org/10.2514/6.2020-4131.

Shen, H., & Tsiotras, P. (2005). Peer-to-peer refueling for circular satellite constellations. *Journal of Guidance, Control, and Dynamics*, *28*(6), 1220–1230.

Space-Track. org (2021). Space-Track.org. https://www.space-track.org. Last Accessed: March 21, 2022.

Yu, J., Ouyang, Q., Chen, X. Q., & Chen, L. H. (2013). Optimal peer-to-peer maneuvers for refueling satellites in circular constellations. *Applied Mechanics and Materials*, *290*, 41. http://dx.doi.org/10.4028/www.scientific.net/AMM.290.41.

Yu, J., Yu, Y. G., Huang, J. T., Chen, X. Q., & Liu, H. Y. (2017). Optimal scheduling of GEO on-orbit refuelling with uncertain object satellites. In *MATEC web of conferences. Vol. 114* (p. 3001). http://dx.doi.org/10.1051/matecconf/201711403001.

Zhang, J., Parks, G. T., Luo, Y. Z., & Tang, G. J. (2014). Multispacecraft refueling optimization considering the J2 perturbation and window constraints. *Journal of Guidance, Control, and Dynamics*, *37*(1), 111–122. http://dx.doi.org/10.4028/www.scientific.net/AMM.538.150.

Zhang, T. J., Yang, Y. K., Wang, B. H., Li, H. N., Li, Z., & Shen, H. X. (Oct 2019). Optimal scheduling for location geosynchronous satellites refueling problem. *Acta Astronautica*, *163*, 264–271. http://dx.doi.org/10.1016/j.actaastro.2019.01.024.

Zhao, Z., Zhang, J., Li, H. Y., & Zhou, J. Y. (2017). LEO cooperative multi-spacecraft refueling mission optimization considering J2 perturbation and target's surplus propellant constraint. *Advances in Space Research*, *59*(1), 252–262. http://dx.doi.org/10.1016/j.asr.2016.10.005.

Zhou, Y., Yan, Y., Huang, X., & Kong, L. (2015). Optimal scheduling of multiple geosynchronous satellites refueling based on a hybrid particle swarm optimizer. *Aerospace Science and Technology*, *47*, 125–134. http://dx.doi.org/10.1016/j.ast.2015.09.24.

Zhou, Y., Yan, Y., Huang, X., & Yang, Y. (2017). Multi-objective planning of a multiple geostationary spacecraft refuelling mission. *Engineering Optimization*, *49*(3), 531–548. http://dx.doi.org/10.1080/0305215X.2016.1190352.