Advanced Optimization for Interplanetary Logistics

Dr. Christine Taylor

Post Doctoral Associate, Aeronautics & Astronautics

Massachusetts Institute of Technology, United States

c_taylor@mit.edu

Diego Klabjan

Professor of Industrial Engineering and Management Sciences

Northwestern University, United States

d-klabjan@northwestern.edu

Hamed Mamani

Operations Research Center

Massachusetts Institute of Technology, United States

hamed@mit.edu

Abstract

This chapter demonstrates a model and solution methodology for designing operational planning for interplanetary exploration missions. A primary question for space exploration mission design is how to best design the logistics required to sustain the exploration initiative. To answer this question, an architectural decision method has been created. The model presented in this chapter is capable of analyzing a variety of mission scenarios over an extended period of time with the goal of defining beneficial mission architectures that enable space logistics. This model can be utilized to evalu-

ate different logistics trades, such as a possible establishment of a push-pull boundary, which can aid in supply pre-positioning. Based on the model, a state-of-the-art solution methodology has been designed and implemented. The model is demonstrated on an Apollo-style mission to both provide an example and validate the methodology.

1 Introduction

On January 14th, 2004, President Bush set forth a new exploration initiative to achieve a sustained human presence in space. Included in this directive is the return of humans to the Moon by 2020 and the human exploration of Mars thereafter, President George W. Bush (2004). The President has tasked NASA with the development of a sustainable space transportation system that will enable continual exploration of the Moon, Mars and "beyond".

Inherent to the problem of transporting people to the Moon, Mars, and "beyond" is sustaining the people and the operations while in transit and at the respective destinations. Especially for long-term missions, the amount of consumables required becomes a significant issue in terms of mass in Low Earth Orbit (LEO), which translates to mission cost. In order to develop a sustainable space transportation architecture it is critical that interplanetary supply chain logistics be considered.

The goal of the interplanetary supply chain logistics problem is to adequately account for and optimize the transfer of supplies from Earth to locations in space. Although the supply items, herein called commodities, themselves may be of low value on Earth, the consideration of these commodities is of high importance and can directly impact the mis-

sion success. As such, it is desirable to find low cost yet reliable methods of transporting these supplies to their destinations.

The space exploration missions will evolve over time, which will generate an increased demand at in-space locations. In order to develop a sustainable architecture it is necessary to recognize the interdependencies among missions and how this coupling could affect the logistics planning. By viewing the set of missions together, as a space network, and optimizing the operations of the transportation system that provides the logistics for the exploration missions, a reduction in cost can be achieved, which promotes a more sustainable system architecture.

There exists a great deal of literature on the design of transportation networks on Earth, for example, see Ball *et al.* (1995), Desrosiers *et al.* (1995). Many of the tools and methods of terrestrial logistics can be extended to space networks. Specifically, time expanded networks represent a method for modeling transportation systems that are operated over time, see for example Ahuja *et al.* (1993). Using this modeling technique, the physical network is expanded and time is incorporated directly into the network definition.

In order to effectively communicate the model, an extensive terminology is developed in Section II. Specifically, the network definition is presented as well as the description of the time expanded network. Furthermore, the commodities or supplies and the elements or physical containment and propulsion units used to transport the commodities are detailed. Section III describes the components of the interplanetary logistics problems. Section IV presents the in-space network optimization model and the optimization methodology used to solve this problem. Section V details the manifesting problem and solution methodol-

ogy. Section VI presents the launching problem and solution methodology. In Section VII, the solution methodology is applied to an example of an Apollo-style mission to both explain the implementation and validate the presented methodology. Section VIII reviews the contributions of this methodology and describes continuing work in this area.

2 Problem Definition

The goal of interplanetary logistics is to determine feasible mission architectures to satisfy the demand generated by the needs of exploration. The key concept of the interplanetary logistics problem is that the demand of crew, consumables, equipment, and other exploration requirements at in-space locations drives the mission requirements. Therefore, the first required input for the interplanetary logistics problem is the definition of these supplies. For example, if the exploration mission is a sortie style mission to investigate a particular location, the demand might consist of a few crew members at a specific location and the supplies necessary to both support the crew and enable the exploration activities.

Given the demand of the mission, it is necessary to determine how and when the supplies on Earth will be transported to the in-space locations. As missions become more complex and evolve over a period of time, a solution may become less obvious. The goal is to minimize the cost of all missions, and therefore, it is necessary to define all pathways and structures used for transport and allow the optimizer to analyze the different architectures to select the best one.

Given this information, the interplanetary logistics problem is to determine low cost mis-

sion architectures, including the underlying logistics network, that satisfy the exploration demand. The generated solution details the scheduling and assignment of supplies to vehicles for in-space transport and launch scheduling requirements. More importantly, however, the output can be used to determine a push-pull boundary for the supplies, the potential of a specific location, either on a surface or in-space, for storing supplies, benefits of in-situ resource utilization over multiple missions, or even the sensitivity of mission architectures to changes in vehicle parameters.

The first step in developing a model for interplanetary logistics is defining a nomenclature that describes the components of the problem. The problem fundamentally consists of three main components: the commodities or supplies that must be shipped to satisfy a mission demand, the elements or physical structures used to both hold and move the commodities, and the network or pathways the elements and commodities travel on. The following sub-sections define the parameters that describe each of these components.

2.1 Networks

In order to transfer the commodities and elements from the origin node to the destination node, the trajectories must be defined. The purpose of the interplanetary logistics model developed is to analyze the multiple choices available for routing all of the commodities and elements to determine the best logistics architecture. To model the different available trajectories, a space network model is created to represent the possibilities available for transferring commodities to their respective destination. The following subsections detail the development of the space network utilized to form the presented model.

2.1.1 Static Network

The physical network, or static network, represents the set of physical locations, or nodes, and the connections, or arcs, between them. The physical nodes, or static nodes, represent the different physical destinations in space, including the origin and destination of all the commodities, as well as the possible locations for transshipment. Three types of nodes have been identified: Body nodes, Orbit nodes, and Lagrange point nodes. This classification distinguishes the type of information required to define a node of each type. The physical arcs, or static arcs, represent the physical connections between two nodes, that is, an element can physically traverse between these two nodes. We define an arc (si, sj) to be a static arc that represents a feasible transfer from static node si to static node sj.

The mathematical description of the static network is given next.

- Define the static network as a graph GS, where $GS = (NS, AS)$.
- Define the set of nodes, $NS = \{s1, \ldots, sn\}$, in the static network corresponding to the aforementioned physical locations.
- Define the set of arcs, $AS \subseteq NS \times NS$, in the static network as the possible transportation links between the static nodes .

An example of an Earth-Moon static network is provided in Figure 1. In this figure, we can see the connection of the Earth surface nodes to the Earth orbit node, representing launches and returns. Similarly, the lunar surface nodes are connected to the lunar orbit

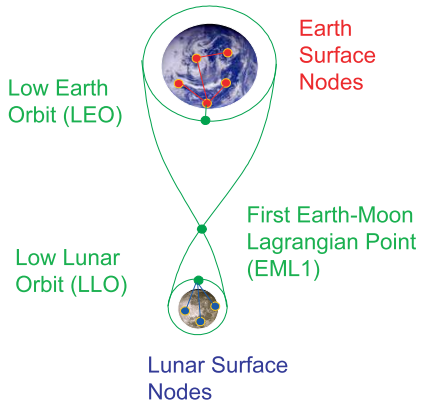

Figure 1: Depiction of an Earth-Moon Static Network

node, representing descent and ascent trajectories. In addition, the orbit nodes, as well as the first Earth-Moon Lagrangian point are connected by in-space trajectories.

2.1.2 Time Expanded Network

In order to analyze sequences of missions that evolve over an extended period of time, and to account for the time-varying properties that can arise in certain astrodynamic relationships, we introduce time expanded networks as a modeling tool. In the time expanded network, the absolute time interval under consideration is discretized into T time periods of length Δt. A copy of each static node is made for each of the time points and the nodes are connected by arcs according to the following rules:

- the arc must exist in the static network,

- the arc must create a connection that moves forward in time, and
- the arc must represent a feasible transfer, with respect to the orbital dynamics.

The mathematical description of the time expanded network is given below.

- Define the time expanded network as a graph $\mathcal{G}$, where $\mathcal{G} = (\mathcal{N}, \mathcal{A})$.
- Define the set of nodes in the time expanded network as $\mathcal{N} = \{i = (si, t) \mid si \in NS, t = 1, \ldots, T\}$. For a given node $i \in \mathcal{N}$, let $s(i)$ and $t(i)$ denote the physical node and the time period corresponding to node i, i.e., if $i = (si, t)$, then $s(i) = si$ and $t(i) = t$.
- Define the set of arcs in the time expanded network as $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$. An arc $a = (i, j) = ((si, t), (sj, t + T^t_{si,sj}))$ exists if and only if (1) $si \neq sj$, there exists an arc (si, sj) in the static network, and the transit time from static node si to static node sj starting at time t is $T^t_{si,sj}$, or (2) $si = sj$ and $T^t_{si,sj} = 1$. Arcs with $si \neq sj$ are called the transport arcs in the time expanded network and arcs with $si = sj$ are called the waiting arcs.

In the time expanded network, a path p is defined as a sequence of nodes. In particular, let $f(p)$ and $l(p)$ denote the first node and the last node of path p, respectively.

Using the static network depicted in Figure 1, we can create the time expanded network in Figure 2. Here, the time expanded network is notional as not all arcs are represented, but how the trajectories evolve in time can be readily seen.

To account for the fact that on certain transfer arcs two burns occur, we slightly modify this time expanded network. We first introduce a new fictitious static node labeled fic.

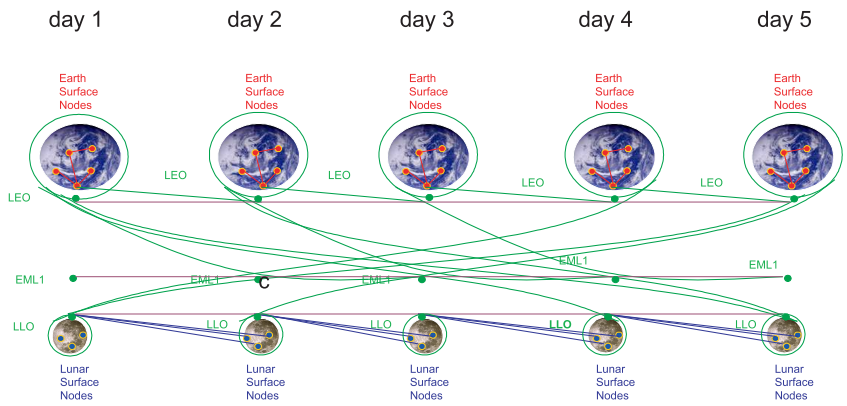

Figure 2: Depiction of an Earth-Moon Time Expanded Network

Note that this node is not related to the actual static network. On every transfer arc $(i, j), s(i) \neq s(j)$ requiring two burns we add a new auxiliary node $k = (fic, t)$ with two arcs; one connects i to k and the other one k to j. The value of t is irrelevant. In this new network, each arc (i, j) with $s(i) \neq s(j)$ corresponds to a single burn. All such arcs are called *burn arcs* and we denote the set of all burn arcs as $\mathcal{A}_B$.

The fuel mass fraction, which represents the ratio of the fuel mass to the initial mass, for element m to execute the burn corresponding to arc $a \in \mathcal{A}_B$ is defined as

$$\phi_a^m = 1 - \exp\left(\frac{-\Delta V_a}{I_{sp}^m g_0}\right),$$

which is taken from the rocket equation Battin (1999). Here, g_0 is Earth's sea-level gravity, I_{sp}^m is the specific impulse of element m and ΔV_a is the change in velocity (ΔV) on arc a.

2.2 Commodities

The goal of the space logistics project is to determine how to meet the demand for the exploration missions. As such, we are investigating how to optimally ship multiple types of commodities. For the purpose of this logistics problem, a commodity is defined as a high-level aggregate of a type of supply, such as crew provisions. Thus, we define a set of $k = 1, \ldots, K$ commodities, each with the following parameters.

- Denote the demand of each commodity as d^k.
- Denote the origin of each commodity as so^k where $so^k \in NS$.
- Define the destination of each commodity as sd^k where $sd^k \in NS$.
- Define the availability interval of each commodity as $to^k = \left[sto^k, eto^k\right]$, where sto^k is the starting time of the interval, eto^k is the ending time of the interval, and $sto^k, eto^k \in \{1, \ldots, T\}$.
- Define the delivery interval of each commodity as $td^k = \left[std^k, etd^k\right]$, where std^k is the starting time of the interval, etd^k is the ending time of the interval, and $std^k, etd^k \in \{1, \ldots, T\}$.
- Define the unit mass of each commodity as m^k when it arrives at the destination.
- Define the unit volume of each commodity as v^k when it arrives at the destination.
- Define the number of specified waiting sequences, that is, the number of locations that the commodity must visit, as nw^k.

By defining a waiting sequence as part of the commodity input, a number of wait arcs along the path can be specified, which allows on-route destinations to be designated. For each waiting arc sequence l where $0 \leq l \leq nw^k$ the following parameters must be defined.

- Define the static node of the wait sequence as sw_l^k.
- Define the required waiting time period as pw_l^k.
- Define the wait interval for each wait sequence as $tw_l^k = \left[stw_l^k, etw_l^k\right]$, where stw_l^k is the starting time of wait interval l of commodity k, etw_l^k is the ending time of wait interval l of commodity k, and $etw_l^k - stw_l^k \geq pw_l^k$.

For example, if we want to specify that a crew must stay on the moon for a period of five days between June 10, 2015 and June 20, 2015, we would set sw_l^k to the static node that corresponds to the specific lunar surface location of interest, pw_l^k to five (assuming each period corresponds to one day), stw_l^k to the time point corresponding to June 10, 2015, and etw_l^k to the time point corresponding to June 20, 2015. Here, we assume that k corresponds to the crew in question and $l = 1$ if this is the only such requirement. It is important to note that in this model a crew member is treated as a commodity. In practice, crewed missions are treated differently during mission planning, however, for the purposes of the architectural design tool created by this model, crew can be considered a commodity with highly restrictive parameter values. By narrowing the availability and delivery windows for a crew commodity, the feasible shipment pathways are limited and reasonable architectures for crewed flights can be obtained.

2.3 Elements

In order to ship the commodities from the origin to the destination locations, we require 'containers' to both hold the commodities and provide propulsion to move the mass through space. These components can be abstracted to a single definition of an element. Elements are physical, indivisible functional units that transport the commodities from origin to destination. An element is classified by the amount of commodity capacity and propulsive capability it possesses. Elements can be divided into two classes: non-propulsive elements $\mathcal{M}_N$ and propulsive elements $\mathcal{M}_P$. The element parameters are (see Figure 3) as follows.

- The maximum fuel mass of a propulsive element m, $m \in \mathcal{M}_P$ is denoted by mf^m.
- The structural mass of element m is denoted by ms^m.
- The mass capacity of element m is denoted by CM^m.
- The volume capacity of element m is denoted by CV^m.
- The cost of element m is denoted by $Cost^m$.

If an arc is selected in the space network, then several elements can be transported on it. The collection of elements on the arc form a vehicle, or stack. If the arc requires a burn in order to traverse it, then for each burn there must be at least one element in the stack that provides propulsion.

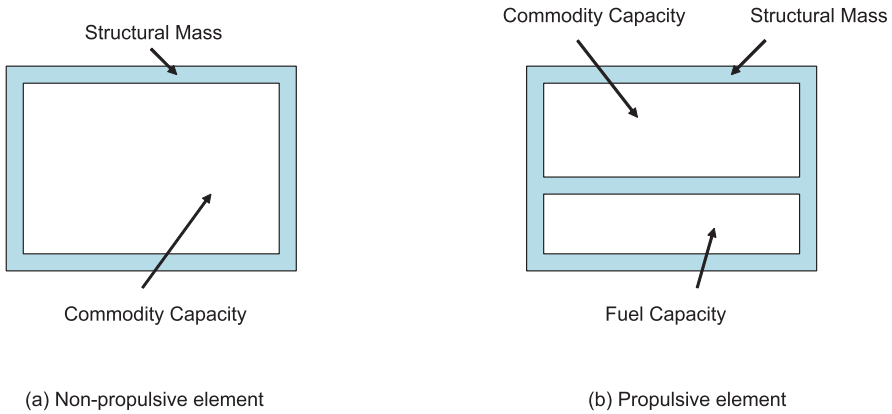

Figure 3: Element Representation

3 Problem Decomposition

The execution of a space mission requires logistical decisions at every step. Logistical decisions are required to accumulate all of the required commodities for space missions, as well as procure and assemble all elements at the launch site. However, since at the time of launch, all of the items required to perform a space mission are co-located at the launch pad, the terrestrial logistics can be decoupled from the interplanetary logistics model. Therefore, the interplanetary logistics model encompasses all of the logistical decisions required between the launch pad and the in-space locations.

There are numerous decisions made during space missions. Although, from a system

perspective, it would be desirable to make all of these decisions concurrently, due to computational limitations, this is not a tractable approach. Instead, the interplanetary logistics model is decomposed into three fundamental components: launch scheduling, manifesting, and in-space network optimization.

Launch is a highly constrained transportation activity, where besides traditional assignment and manifesting decisions, many additional constraints are necessary to model a feasible launch. For this reason the launch problem is decoupled at low Earth orbit (LEO), creating a boundary between launching and the in-space network optimization. This assumption is reasonable since for many mission architectures there exists a delay at LEO before proceeding to in-space destinations.

In-space network optimization, which is solved first, examines the entire mission design space of routing from LEO to all locations in-space. Due to the size of the time expanded network that is generated, this problem can become quite large, with hundreds of millions of variables and tens of thousands of constraints. The decision space of the in-space network optimization focuses concurrently on routing of both commodities and elements, and the assignment of elements to burns. During this phase we only require that the total mass and volume of commodities on an arc do not exceed the total available mass and volume capacity of the elements assigned to the arc in question.

Element packing or manifesting is performed after all of the commodities and element routes have been determined. Given the assignment of commodities and elements to routes optimized in the in-space network optimization, individual commodity units are assigned to the selected elements. Constraints focusing on feasible assignments are

considered while minimizing transfers between elements on route.

Network optimization and manifesting are followed by launching. The output of the network optimization consists of a set of elements and commodities required at LEO at particular points in time. Launching focuses on selecting the appropriate elements to perform the launch (i.e., defining the launch stack that brings all the commodities and elements to LEO). It captures the payload requirements for launch, and scheduling requirements for consecutive launches.

The remainder of this chapter discusses all the components in the order that they are applied in the overall algorithm.

4 Network Optimization

Having defined the scope of the problem and the key components of the problem, this section presents the crucial assumptions in the model, the model formulation, and the solution methodology for the network optimization component.

4.1 Assumptions

In order to define the mathematical model for the in-space network optimization, the modeling assumptions are first presented. The following assumptions about the behavior of elements are made, which are not prohibitive for practical purposes.

Consecutive Burns: When an element performs a burn, it is defined as an active element. An active element burns only on consecutive burns. Once an element be-

comes active, it stays active for a certain number of burns. As soon as it becomes passive (which means that it does not burn on a transport arc), it can no longer be active. Between two consecutive burns, an active element can be idle for an arbitrary length of time, by traveling on waiting arcs in the time expanded network. The number of consecutive burns is not constrained. This assumption clearly does not affect feasibility, but only additionally constrains the feasible set in a manner that is consistent with practice.

Fuel Consumption: We assume that before every initial burn, the active element is filled to capacity with fuel and after the burns are completed, the remaining fuel is expelled. This assumption does not compromise feasibility, but it might create suboptimal solutions.

Docking/Undocking: We assume that any two elements can be docked and undocked. In addition, if any cost is associated with these operations, it is not explicitly captured. If some elements cannot be docked together, then this must be imposed in a separate post optimization analysis.

The first two assumptions eliminate the need to track the consumption of fuel by each element allocated within the network. Enforcing the final assumption eliminates the requirement of tracking the position of each element and the underlying commodities in the stack, as the stack can continually reconfigure.

4.2 Network Optimization Formulation

Having defined the network, commodities, and elements, the in-space network model is presented next. The model is developed in three stages. First, the flow of commodities is defined and the constraints governing the commodity flows are presented. Next, the element flows are modeled with the corresponding constraints. Finally, the constraints governing the capacity and capability, which represent the coupling constraints between the commodities and elements, are developed.

4.2.1 Commodity Flows

Here, we specify the commodity flows, starting with feasibility requirements.

Commodity Path Feasibility For each commodity k it is possible to determine a set of feasible paths $\mathcal{P}^k$. For a given commodity k, path $p \in \mathcal{P}^k$ is feasible only if it originates at node $i = (so^k, t)$ with $t \in to^k$, terminates at node $j = (sd^k, t')$ with $t' \in td^k$, and contains the nodes $\left(sw_l^k, ts_l^k\right), \left(sw_l^k, ts_l^k + 1\right), \ldots, \left(sw_l^k, te_l^k\right)$, where $ts_l^k \in tw_l^k$, $te_l^k \in tw_l^k$, and $te_l^k - ts_l^k = pw_l^k$, for every l, $0 \leq l \leq nw^k$. Additional restrictions such as the total time duration can be easily incorporated in $\mathcal{P}^k$.

Commodity Flow Variables and Constraints We need to determine how many units of commodity k are transported on path p, for any k and $p \in \mathcal{P}^k$. Therefore, for every k and $p \in \mathcal{P}^k$ we have a decision variable $x_p^k \geq 0$ which specifies the number of units of commodity k on path p.

In order to satisfy the demand d^k of a given commodity x_p^k, we have

$$\sum_{p \in \mathcal{P}^k} x_p^k = d^k \qquad \text{for every commodity } k. \tag{1}$$

4.2.2 Element Flows

The other entities in the network optimization model are elements, which we elaborate upon next.

Element Flow Variables As defined in Section II, elements can be classified as non-propulsive or propulsive elements, based on whether the element can carry fuel. This distinction allows for two sets of element variables to be defined. For any non-propulsive element $m \in \mathcal{M}_N$, let us define the decision variable y_p^m such that

$$y_p^m = \begin{cases} 1 & \text{if non-propulsive element } m \text{ travels on path } p, \\ 0 & \text{otherwise,} \end{cases}$$

for each feasible path p in the time expanded network. For any propulsive element $m \in \mathcal{M}_P$, let us define $z_{p,q}^m$ as the decision variable such that

$$z_{p,q}^m = \begin{cases} 1 & \text{if element } m \text{ travels on path } p \text{ and is active during sub-path } q \text{ of path } p, \\ 0 & \text{otherwise,} \end{cases}$$

where p is any feasible path in the time expanded network and q is a sub-path of p. Note that $\sum_q z_{p,q}^m = 1$ if and only if element $m \in \mathcal{M}_P$ travels on path p. Clearly, every element path p must start at LEO. The set of all element paths can be further constrained if additional requirements, such as a return to LEO or locations on the Earth's surface, are required.

For each path p, the element m can be active on at most one sub-path q. Note that some arc $a \notin \mathcal{A}_B$ may be included in the active sub-path q, since an element can be active on a burn arc and then traverse waiting arcs before being active on a consecutive burn arc. Finally, it is possible for a propulsive element to be utilized as a non-propulsive element. For this situation, q is empty.

Element Flow Constraints The element flow constraints govern the feasibility of element selections. The following constraints govern both propulsive and non-propulsive elements as indicated.

- A non-propulsive element can only travel on a single path,

$$\sum_{p} y_p^m \leq 1 \qquad m \in \mathcal{M}_N. \tag{2}$$

- For active elements, we constrain at most one element to be active on any burn arc,

$$\sum_{m \in \mathcal{M}_P} \sum_{p} \sum_{q: a \in q} z_{p,q}^m \leq 1 \qquad a \in \mathcal{A}_B. \tag{3}$$

- A non-propulsive element $m \in \mathcal{M}_N$ can travel on an arc a only if there is an active element on that arc,

$$\sum_{p: a \in p} y_p^m \leq \sum_{m' \in \mathcal{M}_P} \sum_{p} \sum_{q: a \in q} z_{p,q}^{m'} \qquad a \in \mathcal{A}_B, m \in \mathcal{M}_N. \tag{4}$$

- A propulsive element $m \in \mathcal{M}_P$ can travel on an arc a only if there is an active element on that arc,

$$\sum_{p: a \in p} \sum_{q} z_{p,q}^m \leq \sum_{m' \in \mathcal{M}_P} \sum_{p} \sum_{q: a \in q} z_{p,q}^{m'} \qquad a \in \mathcal{A}_B, m \in \mathcal{M}_P. \tag{5}$$

4.2.3 Capacity

For space travel, it is necessary that all commodities be transferred by elements. As such, we must relate the amount of commodities (both mass and volume) present on an arc to the total capacity available on the arc, which is given by the assigned elements. The total mass capacity of an arc is defined as the sum over all elements on the arc multiplied by their respective mass capacities. Since propulsive and non-propulsive elements are defined differently, it is necessary to account for elements of each type separately. The total commodity mass on an arc is simply the sum over all commodities on the arc multiplied by the commodity mass. Similar constraints are required to ensure that the volume capacity is satisfied as well. Constraints (6) and (7) define the mass and volume capacity requirements, respectively.

$$\sum_{k} \sum_{p:a\in p} m^k x_p^k \leq \sum_{m\in\mathcal{M}_P} \sum_{p:a\in p} \sum_{q} CM^m z_{p,q}^m + \sum_{m\in\mathcal{M}_N} \sum_{p:a\in p} CM^m y_p^m \quad a \in \mathcal{A} \tag{6}$$

$$\sum_{k} \sum_{p:a\in p} v^k x_p^k \leq \sum_{m\in\mathcal{M}_P} \sum_{p:a\in p} \sum_{q} CV^m z_{p,q}^m + \sum_{m\in\mathcal{M}_N} \sum_{p:a\in p} CV^m y_p^m \quad a \in \mathcal{A} \tag{7}$$

4.2.4 Capability

The capability constraints determine if a given element has enough fuel to perform a burn, given the total mass on a burn arc. The constraints require that the total fuel of the active element performing the burn on a sub-path q must be enough to carry the total cumulative mass along every arc in q. Let q be an arbitrary sequence of possible consecutive burns and let $a^l = (i^l, j^l)$ be the lth burn arc in q for $l = 1, \ldots, |q|$. Here $|q|$ denotes the number

of arcs in q. Let $r(p,q)$ denote the sub-path along path p from the first node of p to the first node of q, if q is not empty.

The resulting constraint family reads

$$\begin{aligned} mf^m \sum_p z^m_{p,q} + M\left(1 - \sum_p z^m_{p,q}\right) \geq \sum_{l=1}^{|q|} \Phi^m_{q,l} \times \Bigg[& \sum_{m' \in \mathcal{M}_P} \sum_{p: a^l \in p} \sum_{q'} ms^{m'} z^{m'}_{p,q'} + \\ & \sum_{m' \in \mathcal{M}_N} \sum_{p: a^l \in p} ms^{m'} y^{m'}_p + \\ & mf^m + \sum_{\substack{m' \in \mathcal{M}_P \\ m' \neq m}} \sum_p \sum_{q': a^l \in r(p,q')} mf^{m'} z^{m'}_{p,q'} + \\ & \sum_k \sum_{p: a^l \in p} m^k x^k_p \Bigg] \qquad m \in \mathcal{M}_P, \text{ path } q, \end{aligned} \tag{8}$$

where

$$\Phi^m_{q,l} = \phi^m_{a^l} \prod_{l'=l+1}^{|q|} (1 - \phi^m_{a^{l'}}),$$

and M is a big number. The derivation of these constraints is very technical and cumbersome and is therefore omitted.

4.2.5 The Complete Model

Since the cost to route commodities is negligible, we include only the cost associated with elements. The objective function reads

$$\min \sum_{m \in \mathcal{M}_p} \sum_{\substack{p \\ f(p)=s}} \sum_q Cost^m z^m_{p,g} + \sum_{m \in \mathcal{M}_n} \sum_p Cost^m y^m_p.$$

The model includes constraints (1) through (8). In addition, all x variables are nonnegative and all z and y variables are binary.

The constraints defined for the complete model do not represent an exhaustive list, but only the necessary constraints required to produce feasible solutions within the network model. Additional constraints governing the set of feasible arcs that can be traversed by a given element can be included to limit which elements travel on particular arcs (i.e., forbidding any element without a heat shield to return the Earth's atmosphere). Restrictions on commodity placement into elements can also be included for cases where feasibility is required, such as only housing crew in crew compatible elements.

4.3 Solution Methodology

The model just presented is complex and requires the implementation of a sophisticated algorithm in order to obtain good solutions. Due to the high number of variables and constraints, and the complexity of the model, a heuristic optimization method is employed. Although heuristics are not guaranteed to return optimal solutions, they often return good solutions quickly.

By understanding the structure of the problem and the potential solutions, the heuristic optimization algorithms can be tailored to the specific problem to enhance computational efficiency and quality of solutions. For the in-space network optimization, a series of heuristic optimization algorithms are employed to determine a complete solution to the routing and allocation problem. In this section, an overview of the heuristic optimization approach is presented, followed by a more detailed description of each component.

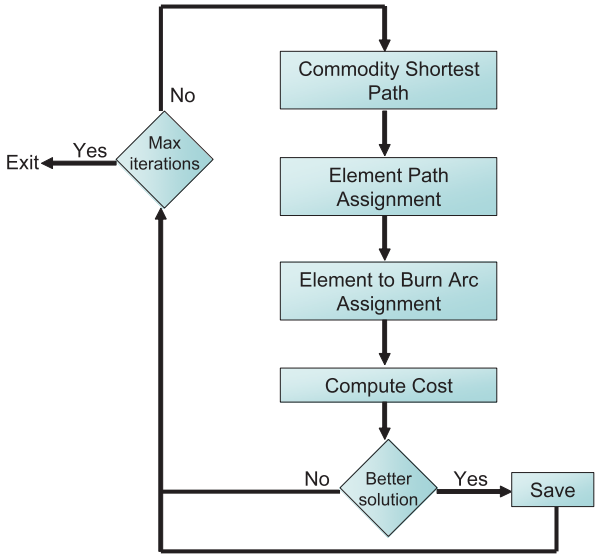

Figure 4: Flow Diagram of Heuristic Optimization

4.3.1 Heuristic Optimization Overview

The optimization of the in-space network design problem has three components: commodity routing, element routing, and burn-arc assignment. The commodity routing is performed first, since the entire architecture is driven by the commodity demand. Next, given the commodity paths through the network, elements are assigned to paths, such that all capacity constraints are satisfied. Finally, since at this point the mass of the elements and commodities are known for each arc in the network, the propulsive element assignment can be performed. At several points within the algorithm, randomization is utilized to generate different outcomes and therefore this procedure is iterated many times to evaluate the different outcomes. Figure 4 shows the flow of the optimization algorithm.

At each iteration, the heuristic determines a feasible set of commodity paths, element

paths and burn-arc assignments, sequentially. If a feasible architecture is found, the cost of the architecture is computed. This cost is evaluated against the cost of the best architecture obtained thus far in the optimization process. If a better architecture is obtained on the current iteration, it replaces the previous best architecture, otherwise, it is discarded. This process is performed until the maximum number of iterations is reached. The remainder of this section provides a detailed explanation of the three components of the heuristic optimization performed in each iteration.

Commodity Routing Commodity routing is performed by implementing a shortest path algorithm that proceeds as follows. A commodity is selected at random and an auxiliary network is constructed for each commodity. The auxiliary network connects a single source node to the nodes where a feasible path can begin and a sink node is connected to the nodes where a feasible path can terminate. For commodities that do not have a specified waiting segment, a single auxiliary network is defined where a source node connects the nodes defined by the availability interval and a sink node connects the nodes defined by the delivery interval.

Given a commodity with nw specified waiting segments, $nw + 1$ auxiliary networks are formed. The first auxiliary network created connects a sink node (sk^{nw}) to the delivery interval. The source node (sc^{nw}) is connected to the nodes in the nw'th waiting segment defined by (sw_{nw}, t), where $etw_{nw} - pw_{nw} \leq t \leq etw_{nw}$. This definition ensures that the defined path segment will be feasible with respect to the required waiting time period of the specified waiting segment. For each subsequent auxiliary network, the sink node

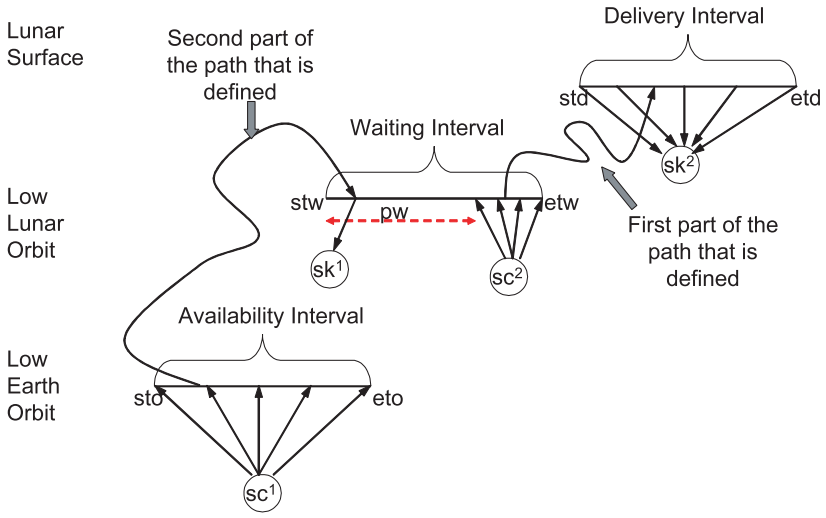

Figure 5: Auxiliary Network for a Commodity with a Single Waiting Segment

connects to the first node in the previously defined path segment and the source node is defined as above with respect to the currently considered waiting interval. In the final auxiliary network, the source node (sc^1) is connected to all nodes in the availability interval. Figure 5 depicts a simple example to clarify this construction.

For each defined auxiliary network, a cost is assigned to every arc. For the first selected commodity, the arc costs represent the ΔV of the arcs. Since decreased ΔV correlates to decreased fuel requirements, a shortest path algorithm is implemented to connect the source node to the sink node at lowest cost, or lowest accumulated ΔV. For the remaining commodities, the arc costs are defined as $\Delta V\,(1-df)^{a_N}$ where df is a specified parameter between zero and one and a_N is the number of times the particular arc has been chosen as an arc in another already assigned commodity path. The reduction in

cost for previously selected arcs reflects the desire to reuse previously selected arc.

The shortest path algorithm is run for each auxiliary network in the specified order of a given commodity until a feasible path is formed between nodes in the availability interval and in the destination interval. This process is then repeated for every commodity until all commodities have been assigned to paths.

Element to Path Assignment After the commodity paths are determined, the element to path assignment is performed for commodity carrying elements. However, in order to perform this assignment, some preliminary manipulations are necessary. Since the network has arcs that only proceed forward in time, the nodes, and therefore arcs, can be arranged based on this order. This order is known as the topological order, and the details can be found in many network modeling books, for example see Ahuja *et al.* (1993). A topological order of the nodes and arcs is necessary to ensure that all assignments on downstream connected arcs are determined prior to the current arc assignment.

For each arc in the topological order, the following procedure is conducted to ensure that the elements assigned to the arcs for carrying commodities satisfy the mass and volume requirements on each arc. Given an arc in the topological ordering, the total mass and volume of all commodities on that arc is readily computed. To select elements to contain these commodities we first examine all already considered arcs, where these arcs are forward in time based on the topological order, to determine if a previously assigned element can be reused to contain commodities on the current arc. This process is repeated until both the mass and volume capacity constraints are satisfied or until no existing elements

can be utilized.

If additional capacity is required, a new element is selected by utilizing ideas from a generalized random adaptive search procedure (GRASP). This algorithm utilizes information about the problem structure and intuition about the characteristics of 'good' solutions to aid the selection of commodity carrying elements. The algorithm proceeds as follows. One of the six score functions shown in (9) is selected uniformly at random, and each element is evaluated against the selected score function.

$$\begin{aligned} S_1^m &= \frac{Cost^m}{CM^m} & S_2^m &= \frac{(Cost^m)^2}{CM^m} \\ S_3^m &= Cost^m & S_4 &= \frac{\sqrt{Cost^m}}{CM^m} \\ S_5^m &= \frac{Cost^m}{(CM^m)^2} & S_6^m &= \frac{Cost^m}{\sqrt{CM^m}} \end{aligned} \tag{9}$$

The probability of selecting a given element is defined as the negative exponent of a given element's score (i.e., $e^{-S_i^m}$) divided by the accumulated probability of all elements. This probability distribution favors elements of low cost and high mass capacity. An element is then selected according to this probability distribution. The process of element selection is repeated until all the mass and volume requirements are satisfied for a given arc.

Figure 6 illustrates the element selection process using a simple example network. The numbers associated with the arcs and nodes represent the topological order of the nodes and the arrows represent the direction of movement through the network. Therefore, beginning in backwards topological order, arc 5 is first assigned element A to carry commodities, and next arc 4 is assigned element B to carry commodities. Therefore when

examining arc 3, first element A and then element B are selected to contain commodities on arc 3. If the mass and volume capacity constraints have not yet been satisfied, then a new element is selected using the GRASP algorithm, and allocated to the arc. In this example, element C is assigned to arc 3 to provide sufficient capacity on arc 3.

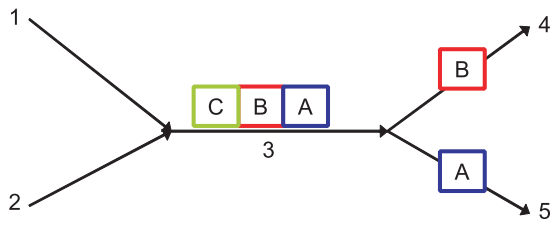

Figure 6: Illustration of Element to Arc Assignment

The element assignment process continues by working in backwards topological order until the mass and volume capacity constraints are satisfied on every arc. From this information, paths for each of the elements can be constructed.

Element to Burn Arc Assignment The final stage of the heuristic optimization is to assign elements to burn-arcs. An element can be assigned to perform a burn if the amount of fuel available in an element is enough to satisfy the capability constraints, and can therefore provide the required ΔV, given the total mass on the arc, as defined by the

rocket equation, Battin (1999). Since both the commodity paths and commodity-carrying element paths are known, the total mass on every arc is known.

Given an arc in the topological order, an element to burn arc assignment is performed as follows. First, forward connecting arcs are examined to determine if a previously allocated propulsive element that has already been utilized to perform a burn on a connecting burn arc can perform an additional burn. If an assignment has not been made, then a check of all elements on the current arc is performed to determine if a commodity carrying element could perform the burn. This second situation is distinguished from the first one because an element that has propulsive capabilities but is assigned to carry commodities is not automatically assumed to be fueled. Thus, selecting a commodity-carrying element to perform the burn requires additional mass be added to the current arc and all previous arcs in the element's path, to account for the fuel of this element.

If the assignment has not yet been made for the given arc, a new element must be added to the architecture to perform the burn. A new element is selected by again employing GRASP, as described above, using the six score functions provided in (9) with the replacement of fuel mass capacity (mf) for commodity capacity (CM).This situation repeats until a selected element satisfies the capability constraint for the given burn-arc. Since this element is new to the architecture, it is necessary to immediately define the path of the propulsive element and update the payload mass on every arc in the path up to this current burn-arc.

Figure 7 illustrates the element to burn-arc assignment process using the simple example network shown in Figure 6. Here, we notice that the elements on both arcs 4 and 5

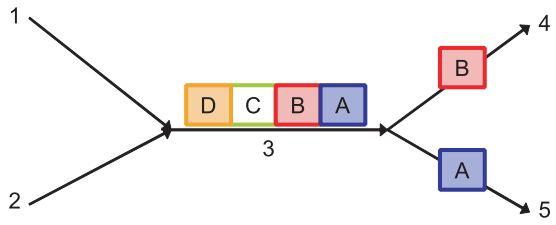

Figure 7: Illustration of Element to Burn Assignment

have been shaded to represent that these elements perform the burn on their assigned arcs and that this fuel must now be accounted for on arc 3. Following the procedure described above, elements A and B are checked to determine if they posses enough fuel to also perform the burn on arc 3. If neither of these elements have sufficient capability, element C is examined to determine *if* the element were fueled, would it be able to perform the burn. If an assignment is still required, the GRASP algorithm is employed to select a new element that is capable of performing the burn on arc 3. In this example, element D is assigned as a fueled element, as denoted by the shading.

5 Manifesting

The output of network optimization only determines the elements and commodities traveling on each arc in the time expanded network. The assignment of individual commodity units to each individual element is not provided. The goal of the manifesting problem is to process the output of network optimization and produce the final manifest. The problem is to appropriately assign individual commodity units to elements on each arc so that the capacity constraints of elements are not violated. There are additional requirements such as not assigning particular pairs of commodities to the same element (e.g., crew together with hazardous material) and considering element capabilities of carrying certain commodities (e.g., crew can only go into CEVs). The criterion is to keep a commodity unit in the same element as long as possible. Thus we minimize the number of transfers. An integer programming based mathematical model is obtained.

Note that the network optimization module does not assign a specific commodity unit to a path (it assigns batches of commodity units to paths). Manifesting is a two stage process. In the first stage, we assign each commodity unit to a path without performing the actual manifests. In the second stage we apply the integer programming model to minimize the number of transfers and carry out the manifests on each arc by following the paths computed in the fist stage.

Consider the example in Figure 8 including a single trajectory arc. Network optimization guarantees that the total commodity mass, in this example 550+150+200+20 kg, is less than or equal to the payload mass of all the elements, i.e. 600+500+100 kg. Manifesting

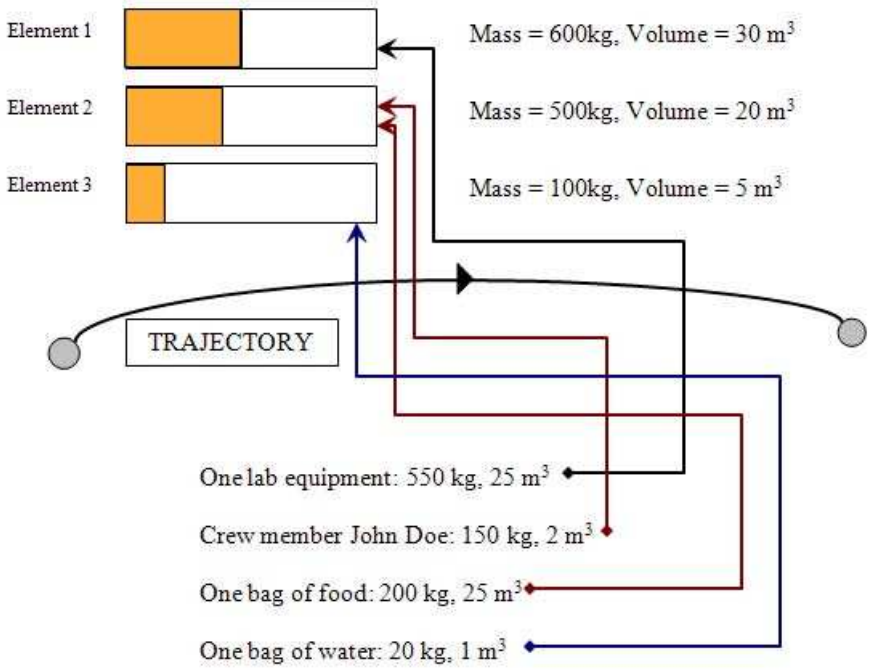

Figure 8: An Example of Manifesting

then assigns these commodity units (crew member, bags of food, etc.) to each individual element. Possible assignments are shown by the vertical lines.

In the reminder of this section we first present the overall algorithm and then we present details on the integer programming formulation, which is a key component.

5.1 The Algorithm

The overall manifesting problem can be modeled as a large-scale integer program, however we opted to develop a simpler algorithm, which is much easier to implement and is tractable. It uses integer programming but on a much smaller scale.

The algorithm is iterative and it combines randomization and integer programming. Each iteration consists of two stages. In the first stage we select paths for individual

commodity units. Note that network optimization specifies commodity flow at the commodity level, but not at the individual commodity unit level. The second stage then assigns each commodity unit to elements for each arc in order to minimize the number of transfers among elements. If arcs are traversed in a particular order, then transfers are well defined. The procedure is iterated by generating different commodity unit paths.

Consider a single commodity $k \in \{1, \cdots, K\}$ and all of its used arcs. These are given by network optimization. Let us denote the underlying acyclic subnetwork by G_k. Each arc has capacity equal to the amount of flow of commodity k. To generate commodity unit paths in G_k we proceed as follows. We start by generating a random path from the source to the destination in G_k. Since the network is acyclic, this can easily be done by randomly selecting an outgoing arc with positive capacity when building the path from the source to the sink. A unique commodity unit identifier is assigned to the underlying commodity unit and the path. After finding this first path, we adjust the capacities to reflect the fact that one unit is used on the arcs covered by the path. Now we select the second path in the similar fashion. We stop after routing the entire commodity demand.

This procedure is repeated for each commodity. After collecting all paths, in the next step of an iteration, we assign each commodity unit to elements on a per arc basis. We first sort the nodes in the topological order and we scan the nodes based on this order. When processing node i, all of its incoming arcs have already been assigned, i.e., the commodity units have already been assigned to elements on these arcs. Thus we can select outgoing arcs of node i one by one and compute an assignment. Assignments on incoming arcs allow us to capture the number of transfers for a given assignment on an

outgoing arc. On an arc the assignment problem is an integer program, which is given later, and it is relatively easy to solve. These steps are repeated until all arcs are scanned.

The overall algorithm iterates these steps. In each iteration different random paths are selected, which then yields a different number of transfers at the end of the iteration. The best solution after a desirable running time is selected and returned.

5.2 The Single Arc Integer Programming Formulation

In the reminder of this section we present the integer program for manifesting on each individual arc. We assume that we are given a set of individual commodity units on an arc and the underlying elements on this arc. In view of the previous discussion, all of these are given from previous steps of the overall algorithm.

For a given arc $(i, j) \in \mathcal{A}$ we define the following quantities.

$K^{(i,j)}$, **set of individual commodity units on arc** (i, j)**:** These are given from previous steps of the algorithm. We stress again that $K^{(i,j)}$ is the set of individual commodity units, hence for each individual commodity unit a unique identifier should be assigned,

$M^{(i,j)}$, **set of individual elements on arc** (i, j)**:** They are obtained based on the output of network optimization. Unlike commodities, elements are given directly as single entities from network optimization.

$E(k)$, **set of forbidden elements to contain commodity** k**:** Some elements can not carry specific commodities, e.g., crew can only travel within a CEV,

$\kappa(k)$, set of forbidden commodities to accompany commodity k in any element:

This models the situation when two commodities with highly different hazard levels should not travel together in the same element, e.g., it might not be desirable to carry a crew member in the same element as a highly hazardous commodity.

Values $E(k)$ and $\kappa(k)$ are part of the input. Note that they are specified at the commodity level. We define the following decision variables.

$$x_{u,m}^{(i,j)} = \begin{cases} 1 & \text{if commodity unit } u \in K^{(i,j)} \text{ is assigned to element } m \in M^{(i,j)} \text{ on arc } (i,j), \\ 0 & \text{otherwise.} \end{cases}$$

The cost is then defined as $c_{u,m}^{(i,j)}$, and it corresponds to the cost of assigning commodity unit u to element m on arc (i,j). This cost is 0 if commodity unit u is already in element m at node i and 1 otherwise. This information is obtained based on the fact that the preceding arcs to (i,j) in the topological sort have already been manifested.

A feasible solution to this problem must satisfy both mass and capacity constraints and it should also exclude any forbidden assignment of commodities to elements or commodities together. The goal is to find a feasible solution with the minimum cost. i.e., to minimize the number of transfers at node i. For a given unit $u \in K^{(i,j)}$ let $b(u) \in \{1, \cdots, K\}$ be the corresponding commodity.

The model reads

$$\min \sum_{u \in K^{(i,j)}, m \in M^{(i,j)}} c_{u,m}^{(i,j)} x_{u,m}^{(i,j)}$$

$$\text{s.t.} \sum_{m \in M^{(i,j)}} x_{u,m}^{(i,j)} = 1 \qquad u \in K^{(i,j)} \tag{10}$$

$$\sum_{u \in K^{(i,j)}} m^{b(u)} x_{u,m}^{(i,j)} \leq C M^m \qquad m \in M^{(i,j)} \tag{11}$$

$$\sum_{u \in K^{(i,j)}} v^{b(u)} x_{u,m}^{(i,j)} \leq C V^m \qquad m \in M^{(i,j)} \tag{12}$$

$$\sum_{m \in E(b(u))} x_{u,m}^{(i,j)} = 0 \qquad u \in K^{(i,j)} \tag{13}$$

$$x_{u,m}^{(i,j)} + \sum_{l \in \kappa(b(u))} x_{l,m}^{(i,j)} \leq 1 \qquad u \in K^{(i,j)},\ m \in M^{(i,j)} \tag{14}$$

$$x_{u,m}^{(i,j)} \in \{0,1\} \qquad u \in K^{(i,j)},\ m \in M^{(i,j)}.$$

Constraints (10) require that each commodity is assigned to an element. Inequalities (11) and (12) impose that the mass and volume capacities are not violated. Constraints (13) capture forbidden element-commodity configurations. Restrictions (14) impose the hazard level requirements. This model can be relatively easily solved by integer programming solvers. Once all solutions are gathered for all arcs, we obtain a complete manifest.

6 Launching

Network optimization and manifesting deal with operations beyond LEO. To complete the puzzle we need to specify how to bring all required commodity units and elements to LEO. After manifesting, we know the demand for each element and commodity unit and the corresponding suggested time at LEO. Launching specifies launch vehicles or stacks

and the corresponding launch times in order to bring the elements to LEO.

In the launch scheduling problem, we are given an a-priori set of demand points. Each demand point is specified by the requested time period, the element and commodity units inside it. All of these quantities are given as part of the output of the network optimization module and manifesting.

A solution is a schedule of launch events, i.e., a series of launches over time, where in each event only a single launch vehicle is launched, see Figure 9. Each event is characterized by the launch time, the launch space port (e.g., Kennedy Space Center, Russian launch ports), and the launch vehicle type. In addition, a full description of the launch vehicle's "cargo", i.e., the elements within the launch vehicle, should be provided. A solution has to satisfy all of the launch vehicle mass capacity, vehicle-element permissibility, and launch pad preparation and cleanup time constraints. On the other hand, we must meet all of the demand points or at least come as close as possible with respect to the suggested time. Among all feasible solutions the problem is to find a solution that minimizes the launch cost and penalty for deviating from demand point times. The formal description of the objective function and constraints is provided in Section 6.1.

It is possible to show that the launching problem is $\mathcal{NP}$-hard. Even more, it is very unlikely that there exists a constant time approximation algorithm for a simplified version of the launching problem (unless $\mathcal{P} = \mathcal{NP}$).

Due to the complexity of this problem, it is not possible to solve it to optimality. For this reason a heuristic methodology has been developed to circumvent this difficulty. We as-

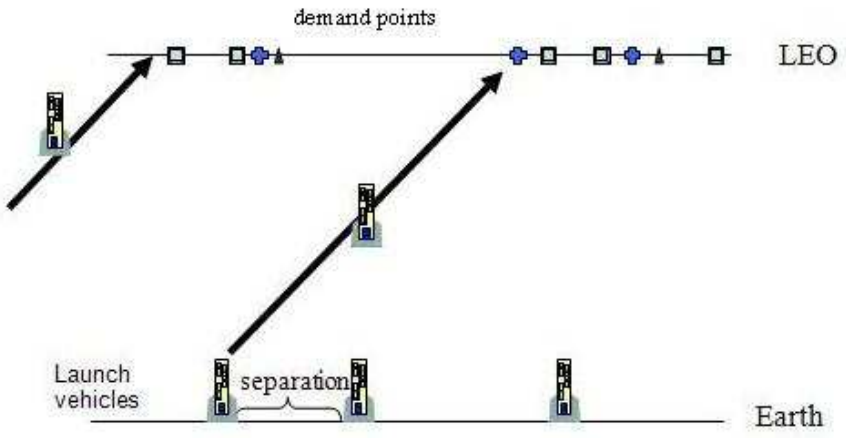

Figure 9: Launch Scheduling

sign penalties for an element to reach LEO before or after its due date. There is a trade-off in early arrivals. They might be desirable due to robustness or undesirable due to perishability. This trade-off can be controlled by appropriately setting the penalty functions. On the other hand, late arrivals should be heavily penalized.

6.1 Formulation

To clearly define the problem, we introduce the following quantities.

Elements ($\mathcal{M}_N \bigcup \mathcal{M}_P$)**:** The set of requested propulsive ($\mathcal{M}_\mathcal{P}$) and non-propulsive ($\mathcal{M}_\mathcal{N}$) elements at LEO. Each element may contain commodity units as its cargo. These "filled" elements at LEO are part of the output of manifesting and network optimization.

Vehicles ($\mathcal{V}$)**:** The set of all possible launch vehicles to be launched from earth to LEO. They are disposable after arriving to LEO and cannot be reused.

Launch or space ports ($\mathcal{LN}$)**:** Locations on Earth from which vehicles can be launched.

Demand points correspond to $\mathcal{M}_N \bigcup \mathcal{M}_P$. Each demand point $m \in \mathcal{M}_N \bigcup \mathcal{M}_P$ is characterized by:

1. mt^m: total mass of element m filled with commodity units. i.e., $mt^m = ms^m + \sum_u m^u$ where the summation is taken over all commodity units u inside element m (given by manifesting).

2. td^m - due time at LEO (given by network optimization).

In addition, for each m, we have the following parameters.

1. $V_m \subseteq \mathcal{V}$ - permissible (allowable) vehicles to carry element m, e.g., elements that contain crew cannot be launched by certain vehicles.

2. w^m - wait time penalty. If element m arrives at LEO before its due date td^m, a penalty of w^m per time unit is associated with it. This models loss of fuel of a propulsive element due to evaporation, slow "sinking" of elements towards Earth due to gravitation, and wait time penalties associated with perishable commodity units packed in the element. It could also encourage early arrivals to enforce robustness.

3. $\tilde{w}^m$ - late arrival penalty. If an element arrives at LEO after its due date td^m, this penalty per time unit is assigned. Typically it would be a very large number. If an element is delivered after the due date, this clearly requires a recourse and adjustments to the entire solution.

Launch vehicle $v \in \mathcal{V}$ is characterized by

1. MC^v - payload mass, i.e., the total mass of cargo that a vehicle can carry,

2. $Cost^v$ - cost of using vehicle v.

A space port or node corresponds to a launch pad at a launch site. We define:

1. $\mathcal{V}_n \subseteq \mathcal{V}$: permissible (allowable) vehicles to be launched from node n,

2. w_n^v: time needed to prepare the pad *before* launching vehicle v from node n,

3. $\bar{w}_n^v$: time needed to "clean-up" the pad *after* launch of a vehicle v from node n.

A solution is a schedule of launch events, i.e., a series of events over time, where in each event only a single vehicle is launched. Each event is characterized by the launch time, the launch node, the launch vehicle, and the set of elements the vehicle carries.

The decision variables are:

1. for every element $m \in \mathcal{M}_N \bigcup \mathcal{M}_P$ an assignment into vehicle $v(m) \in \mathcal{V}$,

2. for every vehicle $v \in \mathcal{V}$ the time $t(v)$ of launch (under the convention that if $t(v) = 0$, then this vehicle is not used),

3. for every vehicle $v \in \mathcal{V}$ an indicator $I(v)$ whether it is used or not (i.e., $I(v) = 1$ if and only if $t(v) > 0$),

4. for every vehicle $v \in \mathcal{V}$ the node $n(v)$ from where it is launched.

The problem reads

$$\text{minimum} \quad \sum_{v\in\mathcal{V}} Cost^v I(v) + \sum_{m\in\mathcal{M}_N \bigcup \mathcal{M}_P} (td^m - t(v(m)))^+ w^m + \sum_{m\in\mathcal{M}_N \bigcup \mathcal{M}_P} (t(v(m)) - td^m)^+ \tilde{w}^m$$

subject to.

$$v(m) \in V_m \qquad m \in \mathcal{M}_N \bigcup \mathcal{M}_P$$

$$\sum_{m:v(m)=v} mt^m \leq MC^v \qquad v \in \mathcal{V} \tag{15}$$

$$n(v) \in \mathcal{V}_n \qquad v \in \mathcal{V}, n$$

$$t(v) > 0 \rightarrow I(v) = 1 \qquad v \in \mathcal{V} \tag{16}$$

$$t(v) = 0 \rightarrow I(v) = 0 \qquad v \in \mathcal{V} \tag{17}$$

$$t(v) + \bar{w}_n^v + w_n^{v'} \leq t(v') \qquad n \in \mathcal{LN} \text{ and } v \in \mathcal{V}, v' \in \mathcal{V} \text{ such that } n(v) = n(v') = n \text{ and } t(v) < t(v') \tag{18}$$

$$t(v) \geq 0 \qquad v \in \mathcal{V}$$

Requirements (16) and (17) link indicator variables $I(v)$ with $t(v)$, i.e., they impose that the cost is incurred as soon as we launch a vehicle ($t(v) > 0$). Constraint (18) impose the preparation and cleanup times. The remaining constraints impose the basic feasibility restrictions. The problem is a combination of machine scheduling ((18) can be viewed as non-preemptive machine scheduling constraints) and bin packing (capacity constraints (15) and the fact that we minimize the number of vehicles).

6.2 The Heuristic

Due to the complex nature of the launching problem, we propose the following heuristic in order to produce a launch schedule. A launch is determined by the launch vehicle, elements to be launched, launch node, and the launch time. Instead of finding all these decisions at once, the main idea is to obtain them sequentially. The key is to first select the elements to be launched based on the due date. Next an appropriate low cost launch vehicle is selected to minimize the penalty, and at the end the launch time is computed. All these steps, which specify a single launch, are repeated until all demand points are launched. The entire heuristic is shown in Algorithm 1.

1: Keep up-to-date a list $D[m, td^m]$ of the demand points which are not yet assigned to a vehicle, sorted in ascending order of their due date.
2: Let D_k denote the set of the first k consecutive elements in D.
3: **while** D is not empty **do**
4: {In the following "for loop" we loop over the launch nodes}
5: **for all** i keep track of the time t_{n_i} after last cleanup of launch node $n_i \in \mathcal{LN}$ and **do**
6: Let $j := 0$
7: { Make the following loop repeat until either $\bar{V}_j^i = \emptyset$ or we reach the end of D}
8: **loop**
9: $j := j + 1$
10: Let $\bar{V}_j^i \subseteq \cap_{m \in D_j} V_m$ be the largest family of vehicles such that each $v \in \bar{V}_j^i$ in this family satisfies the mass constraint: $\sum_{m \in D_j} mt^m \leq MC^v$.
11: **end loop**
12: If necessary, correct j to be $j := j - 1$ in order that V_j^i is the last non-empty set.
13: **for all** $k = 1, ..., j$ **do**
14: Let $p_k^i := \min_{v \in \bar{V}_k^i} Cost^v(\bar{w}_{n_i}^v + w_{n_i}^v)$ {best estimate of the cost contribution}
15: Let $v_k^i \in \bar{V}_k^i$ be a vehicle realizing this cost contribution
16: **end for**
17: Let $cost^i := \min_{k=1,...,j} p_k^i \frac{1}{\sum_{m \in D_k} mt^m}$
18: Let k^i be an index realizing $cost^i$
19: **end for**
20: Let $cost^* := \min_i cost^i$ and let i^* be an index realizing this cost
21: Pack into $v_{k^{i^*}}^{i^*}$ the elements in $D_{k^{i^*}}$. Delete from D these elements.
22: Let

$$t^* = \operatorname{argmin}_{t \geq t_{n_{i^*}} + w_{n_{i^*}}^{v_{k^{i^*}}^{i^*}}} \sum_{m \in D_{k^{i^*}}} \left[(td^m - t)^+ w^m + (td^m - t)^- \tilde{w}^m \right].$$

Set launch time of $v_{k^{i^*}}^{i^*}$ to be t^*

23: Update $t_{n_{i^*}}$ to be t^* plus the necessary wait time after launching it, i.e., $\bar{w}_{n_{i^*}}^{v_{k^{i^*}}^{i^*}}$
24: **end while**

Algorithm 1: Launch Scheduling Heuristic

In step 1, filled elements are sorted so that those with closer due time are more desirable. In the heuristic we *always* launch elements in this order, e.g., if the third element in D is launched in a given solution, then also the first two elements in D have to be launched at the same time. In the while loop, steps 3-24, we assign all of the elements in the list to launch vehicles in the following manner. First, the set of all vehicles that can hold the

first j elements D_j in D is obtained for each j. This set is denoted by $\bar{V}_j^i$ as it depends on the space port i. This is indicated in the loop 8-11. Then for each j we identify a vehicle in $\bar{V}_j^i$ with the lowest estimated cost contribution. The direct cost $Cost^v$ is clear, however it needs to be combined with the penalty contributions, see step 14. At this stage for each k we have the most promising vehicle v_k^i (loop in 13-16). It is intuitive that launches carrying large masses are more desirable. To offset this effect we divide the estimated cost contribution by the total mass that launch vehicle v_k^i carries. Note that this is readily available from the definition of D_k. Then we choose the best vehicle based on this adjusted cost. This vehicle is selected among $\{v_k^i\}_k$ (steps 17 and 18). At the end of loop 5-19, for each space port i we have a single launch vehicle. Finally in step 20 we greedily select a single space port. This gives us a unique launch vehicle and the corresponding elements to be launched. In step 21 we update the data. The launch time then is selected in step 22.

7 Apollo 17 Example

For such a complex problem it is helpful for understanding the model to examine a well defined problem. Using the Apollo 17 elements, a simple example has been constructed to determine how the variables above would be defined. The example has three commodities that need to be sent to the Apollo 17 landing site. The commodity properties are listed in Table 1 and 2. Exploration encompasses all equipment required to perform the desired exploration. The “Pacific” ending node denotes the splash down node.

In addition to the commodity properties, the properties of the elements available to

Table 1: List of Commodities and Properties for Apollo 17 Example

Class of Supply	Demand	Starting Node	Starting Time Interval	Ending Node	Ending Time Interval	Mass	Volume
Explor.	42	LEO	1, 17	Lunar site	11, 17	10 kg	.5 m^3
Crew	2	LEO	1, 17	Pacific	11, 17	100 kg	2 m^3
Crew	1	LEO	1, 17	Pacific	11, 17	100 kg	2 m^3

Table 2: List of Commodities and Properties for Apollo 17 Example (cont.)

Class of Supply	# Wait Arcs	Wait Node	Wait Period	Wait Interval
Exploration	0			
Crew	1	Lunar site	3	7,13
Crew	1	Lunar Orbit	5	7,13

both contain and transport the commodities must be defined. A list of these elements is provided in Table 3.

Figure 10 depicts the solution for this example. As we can see, all three commodities are shipped together from LEO. Upon arrival at lunar orbit, the commodity associated with the two crew members travels directly to the surface, where it remains for three days. The remaining two commodities wait in lunar orbit until the exploration equipment can be delivered on day 11. The remaining crew member waits in lunar orbit to rejoin the other two crew members before returning to Earth.

Notice that in Figure 10 the crew travels in a lunar module (LM) descent stage. This is a

Table 3: List of Elements and Properties for Apollo 17 Example

Element Type	Fuel Mass	Isp (sec)	Structural Mass	Mass Capacity	Volume Capacity	Number Available	Cost (mil)
Saturn V 1st Stage	2,150,999	304	135,218	0	0	4	692
Saturn V 2nd Stage	451,730	421	39,048	0	0	4	307
Saturn V 3rd Stage	106,600	421	13,300	0	0	4	151
SLA	0	0	1,837	0	0	4	0.9
Command Module	0	0	5,806	100	1	4	148
Service Module	18,413	314	6,110	0	0	4	118
LM Descent Stage	8,156	311	1,984	500	5	4	57
LM Ascent Stage	2,358	311	2,189	100	1	4	79

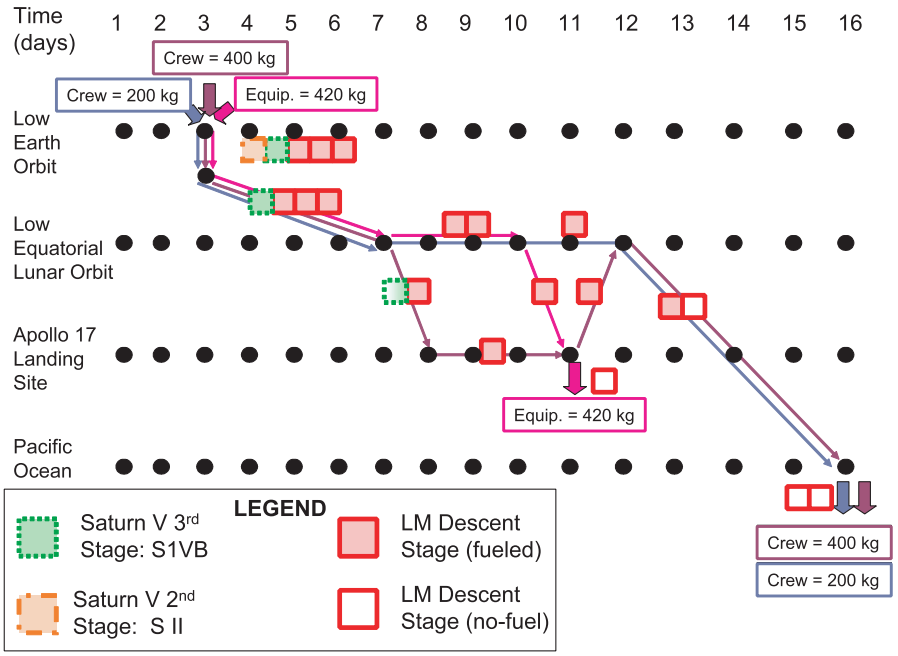

Figure 10: Apollo 17 Example

feasible solution since the LM has enough capacity to hold the two crew members. Considerations, such as a feasible element assignment for a given commodity are not handled within the in-space network optimization framework and are addressed in manifesting. In practice, several additional constraints must be captured, such as restrictions of having only selected elements on specified arcs and the fact that only particular elements can hold some commodities. We have already discussed such requirements in Section IV.

This solution is not only feasible, but represents a good architecture, given the constraints on the commodity paths. If the delivery interval were expanded for the exploration equipment, the optimizer could then choose to combine both the exploration equipment and the two crew members for the surface descent. However, increasing the waiting intervals for the two commodities corresponding to the crew would not effect the optimal solution, since it is desirable for the two commodities to travel together on the return trip to Earth.

The launching solution is not interesting since a single launch is required and it is not shown here.

8 Conclusion

In order for space exploration to be sustainable, interplanetary logistics must be considered during mission planning. Research conducted in the terrestrial logistics and operations research communities provides a wealth of modeling tools and solution approaches that can be extended to enable interplanetary logistics decisions. This work explores

the requirements necessary to define the interplanetary logistics problem and extends a modeling tool traditionally utilized in terrestrial logistics to incorporate the astrodynamic relationships and other aspects of space travel.

Using the time expanded network as a framework, a complex mathematical model was developed to incorporate the fundamental constraints of in-space transportation. Due to modeling complexities and problem size, a heuristic optimization algorithm was developed to explore the design space and find good solutions to the complex problem. This methodology was demonstrated for the example of an Apollo-style mission to both clarify and validate the model.

Continuing work on this methodology includes incorporating more fidelity into the model to more accurately capture the requirements of space travel. Specifically, the incorporation of gain and loss factors for commodities captures the dependance of the amount of a commodity on the shipment path. Including gain and loss factors creates a trade-off between pre-positioning of commodities and the extra commodity mass required to satisfy the specified demand. Improvements in the solution approach can also be obtained by utilizing the presented optimization methodology as an initial solution to a “global” and more robust optimization approach, such as branch-and-price. Finally, the design space can be expanded to include low-thrust propulsion elements, which in turn requires the definition of corresponding pathways in the network.

Acknowledgments

The authors would like to thank the following people for their contributions to this work:

- Miao Song assisted with the model development and implementation,
- Prof. Olivier de Weck and Prof. David Simchi-Levi provided funding for this research.

References

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall.

Ball, M., Magnanti, T., Monma, C., and Nemhauser, G., editors (1995). *Network models*. Elsevier Science.

Battin, R. H. (1999). *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. AIAA Education Series.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Handbook in Operations Research and Management Science, Network Routing*, pages 35–139. Elsevier Science Publishers.

President George W. Bush (2004). A renewed spirit of discovery: A president's vision for U.S. space exploration. Speech given on January 14.