Guide to the Asteroid Mining Mission Optimizer

Rory Lipkis

Version 1.0

Introduction

This guide contains an overview of the purpose and basic structure of the Asteroid Mining Mission Optimizer (AMMO) as well as an appendix, which elaborates on several important concepts and discusses the effect of various program parameters on the overall performance. It is still incomplete.

Purpose

The AMMO is a powerful tool which, in its final form, will be able to plan an efficient and profitable distribution of resources within a specified interplanetary economy. Because of the large timescales involved in space travel, it is highly beneficial to plan long-term schedules, rather than deciding on a short-term basis.

Program Structure

(i) **The updater** accepts a set of orbital elements with their respective epochs, and updates them to the beginning of the schedule.

(ii) **The preprocessor** accepts a set of updated orbital elements, corresponding to Mars and a collection of bodies in strictly heliocentric orbits, such as asteroids or other planets. For each site, the program creates two porkchop plots of the entire schedule duration, for transfers from and to Mars. From each heatmap, the program locates all local minima below the defined cutoff and compiles a list of all possible transfers for each direction.

These lists, along with other program constants, are placed into an `struct`, which allows quick and efficient passing and referencing in functions. This property is especially important for the optimizer, which relies on deep function recursion.

(iii) **The optimizer** is a recursive program that determines the globally near-optimal schedule for a single spacecraft shuttling material from mining sites to Mars.

For a given location and time, the program determines the available actions. For a spacecraft at Mars, the available actions are transfers to any of the other sites. For a spacecraft at another site, the option is to return to Mars. In both cases there is an additional option of terminating the schedule early.

For every available action, the program references the precompiled transfer lists and searches for the next available k transfers. For each of these transfers, the program advances to that time and location and recursively repeats the process.

Throughout the exploration, the program keeps track of both the current cumulative cost of the schedule, as well as the optimal cost (the lowest cost found so far). If the cost ever exceeds the optimal cost, the program backtracks, abandoning the current branch. If the program reaches the end of the schedule, it updates the optimal cost.

When the exploration has finished, the optimizer returns the optimal cost and schedule.

(iv) **The postprocessor** compiles the optimal schedule into a series of graphs and metrics.

Appendix

(i) **The search algorithm** for the optimizer is formally a depth-first shortest-path algorithm with backtracking and opportunity cost. This is an unusual choice, since the majority of popular shortest-path graph-traversal algorithms are breadth-first. However, the choice is justified for two reasons:

(a) The graph is a tree with a potentially extremely high branching factor. A breadth-first traversal would quickly exhaust the machine memory, without providing any benefits.

(b) The backtracking element allows the vast majority of potential branches to be eliminated. Since the tree is stored implicitly, in the form of the preprocessed lists which are assembled into a tree structure at runtime, backtracking eliminates large amounts of potential computation.

The backtracking element, which is indispensable to the success of the optimizer, relies on the condition that cumulative cost can only *increase* during the tree traversal. With this condition satisfied, if the current cost is higher than the optimal cost, the program can ascertain that no extension of the current schedule could possibly be optimal.

(ii) **The cost function** is carefully designed to account for several factors:

(a) Every transfer carries a delta-V cost.

(b) Every time the spacecraft returns to Mars, it carries a payload of mined materials with a certain revenue.

(c) In order for the program to explore mining schedules, it must allow for returns on investment. It must be willing to make a decision that carries a delta-V cost, with the knowledge that there will be returns in the future.

(d) The search algorithm necessitates that all costs be positive.

The last two factors are particularly problematic, and are efficiently solved with the introduction of an opportunity cost.

(iii) **The opportunity cost** r_{op} is an accumulation rate of cost that is applied equally to all traversals of the decision tree. It represents a effective penalty for inactivity, and can be understood through several example calculations.

(a) For exploration to occur at a given point, the cost of terminating the schedule early must be greater than the cost of transferring to another site (otherwise the backtracking element will immediately abandon the exploratory branch).

$$r_{\text{op}}(T_{\text{f}} - T) > \Delta V_1 + r_{\text{op}}\Delta T_{\text{tr},1}$$

This inequality can be simplified to yield

$$r_{\text{op}} > \frac{\Delta V_1}{T_{\text{f}} - T_{\text{arr}}}$$

where T_{arr} is the time of arrival at the destination. This means that for a fixed r_{op}, the program be more cautious exploring near the end of the schedule, where terminating the schedule early is less costly.

(b) For a given exploration schedule to be ultimately chosen, the cost of terminating the schedule early must also be greater than the cost of transferring to another site, staying there for a period of time, returning with a revenue, and then terminating the schedule early.

$$r_{\text{op}}(T_{\text{f}} - T) > \Delta V_1 + \Delta V_2 + r_{\text{op}}(T_{\text{f}} - T) - R$$

This inequality can be simplified to yield

$$R > \Delta V_1 + \Delta V_2$$

This means that the choice of r_{op} ultimately has no effect on the optimal schedule found, which is a desirable property. The only determinant of a given schedule's optimality is the balance of revenue and cost.

(c) For the cost of return to be positive,

$$\Delta V_2 + r_{\text{op}}(T_{\text{wait}} + \Delta T_{\text{tr},2}) - R > 0$$

This inequality is necessary for unconditional algorithmic stability. The tightest bound is considered, in which the transfer requires no energy and occurs in the minimum transfer duration, leaving immediately after arrival at the asteroid. In this case,

$$R < r_{\text{op}} \Delta T_{\text{tr,min}}$$

Therefore, performance is guaranteed if r_{op} is chosen such that

$$r_{\text{op}} > \frac{R}{\Delta T_{\text{tr,min}}}$$

(iv) **The revenue** is the value of the materials returned to Mars. In order to be directly compared with the delta-V cost, it needs a price p, which sets its value in units of delta-V. The value is then given by

$$R = pM$$

where M is the quantity of material returned. The price is set by the specifics of the interplanetary economy, that is, the current demand for various resources. If price is sufficiently high,

$$pM = R > \Delta V_1 + \Delta V_2$$

and a given transfer will be considered for inclusion in the optimal schedule. Therefore, in order to explore as much as possible, p can be set to

$$p > \frac{2\Delta V_{\text{max}}}{M_{\text{max}}}$$

This condition guarantees that all transfers will be considered "worth" the investment. Note, however, that this is not necessarily the desired or valid situation. The demand for the material will ultimately determine p.

Although seemingly simple, the price variable is the most important element of the entire program. In effect, it allows for complex economies to be embedded in the simulation. For instance, differing values of p can exist for differing materials, which are found at different sites (differing values of M can also be implemented, to reflect the respective mining capacities). Although it is not currently supported, the program can be easily extended to allow for multiple delivery sites. These sites would have differing demands, and therefore different values of p.

Ultimately, simply by making p a function of source, destination, and time, a complete interplanetary economy can be specified, reflecting the various demands in multiple locations for multiple materials, and how the demands change in time. The program would then determine the most efficient and profitable distribution schedule for that economy. This functionality will be developed in the next few weeks.

(v) **The sub-option branching factor** k is the number of future transfers that are considered for a given destination at a given time. Occasionally, a later transfer will arrive sooner or have a lower cost. However, this effectively increases the branching factor by a factor of k, since all future transfers are relatively feasible, and thus are not rapidly pruned by the backtracking function. As a result, it is recommended to set $k = 1$, except for simple systems.