Applications Of Graphs In Trajectory Design

Abstract for the 25th International Symposium on Space Flight Dynamics

Juan Arrieta
Jet Propulsion Laboratory, California Institute of Technology
Juan.Arrieta@jpl.nasa.gov

Graphs are widely studied data structures and have found applications in many corners of human endeavor. Their pervasiveness has led to a strong theoretical foundation, a broad collection of algorithms, the availability of high-quality software, and the creation of standards for their storage and visualization.

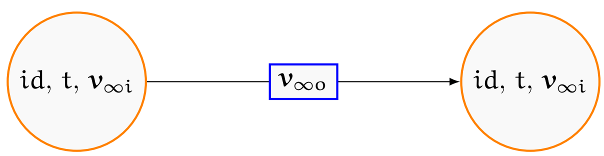
In domains such as the study of social networks the relationship is instinctively analogous: friends and friendships seem to be naturally represented by nodes and edges. In other domains, such as the calculation of sparse Jacobians, the analogy is more subtle.

During the development of methods and tools for the preliminary design of gravity tours in the Jovian system, I found three main applications of graphs which have proved valuable. In my contribution to the conference I will expand on these three aspects and how they are coming together in the implementation of a tool for the design of gravity tours.

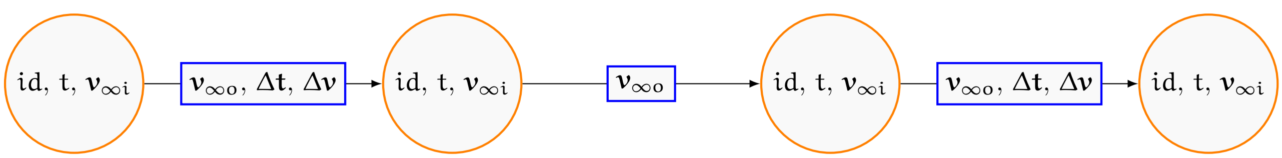
Representation of Gravity-Assisted Tours

The design of trajectories for missions like the Voyagers, Galileo, and Cassini entails a complex piece-wise construction process aiming to *join* various flybys together. A mission designer typically starts from given incoming hyperbolic conditions at some body, generates various possible subsequent flybys, selects a specific one, and continues in this manner until a satisfactory multiple-flyby trajectory has been created. The process is complex because one can generate subsequent flybys using various *transfer strategies*, such as resonant, non-resonant, and π-transfers, and these can be subject to different modeling considerations, such as the fidelity of the gravity field or the use of maneuvers.

I will present a technique for *representing* such process using directed graphs. The graph representation enables efficient calculation, storage, and data mining. In this technique nodes represent incoming conditions at some flyby body, and each edge represents the transfer between two nodes. For example: in the patched-conic approximation a node contains a body



identifier, $\mathtt{id}$, a timepoint, $\mathtt{t}$, and an incoming hyperbolic velocity vector, $\boldsymbol{v}_{\infty i}$; an edge contains an outgoing hyperbolic velocity vector, $\boldsymbol{v}_{\infty o}$, and references to the *from* and *to* nodes. Deceivingly simple, this representation enables the implementation of complex behaviors such as (1) the generation of nodes in either depth-first or breadth-first manner, (2) the isolation of *tours* within the graph as a list of edges, and (3) the *serialization* of the graph to various standard formats for storage and visualization. The graph format accommodates for additional information *attached* to nodes and edges to represent

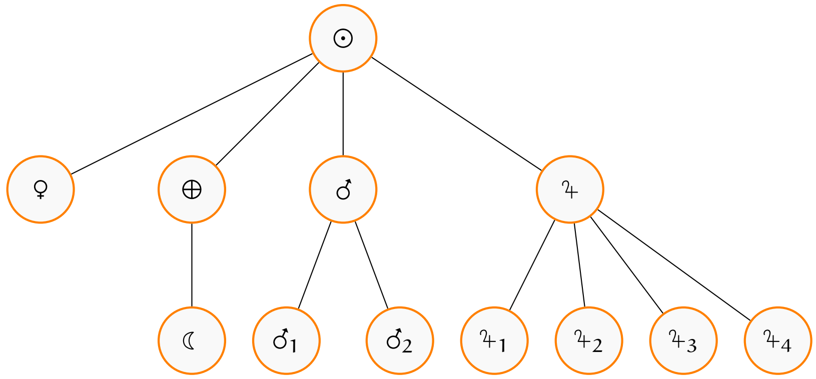


additional transfer events. For example: a Δv executed Δt time units from the flyby time could be added to some edges to denote maneuver locations for non-ballistic transfers.

It would seem natural to represent spacecraft trajectories as graphs: world-events linked by transfers and, in fact, the leading software for trajectory optimization such as MALTO, CATO, Cosmic, SOCS, and Copernicus are all based on similar concepts: *break points* and *control points*; *nodes* and *stages*; *constraints* and *phases*. Their relationship to graphs, however, is often secondary and mostly in the form of a list of nodes and edges $[n, e, n, e, \ldots]$. I propose what I perceive to be a more cohesive methodology for representing different phases of a trajectory as a directed graph where the nodes denote the junction points (constraints) between different phases and the edges represent the dynamical model joining the nodes.

Modeling Multibody Gravity Using Undirected Graphs

Depending on their distance and potential, the various gravity sources acting on a body can be treated as barycenters, point masses, or distributed masses; the distinction can be important both for numerical and performance reasons. For example: it is reasonable for an Earth orbiter to consider the perturbations due to the Jovian system taken as the barycenter of Jupiter and the four Galilieans; this calculation entails one—and not five—ephemerides evaluations. Similarly, it may be reasonable to consider the Moon's distributed potential, but less reasonable to consider, for example, Saturn's oblateness. The decision



of what gravity sources to treat in what manner at a given time is delegated to a *gravity model*, which can also be used to determine the appropriate integration center.

I will present a technique for modeling multibody gravity using an undirected graph. It is inspired by the manner in which gravity is modeled in the MONTE library using k-ary trees, but differs from it in three major aspects: (1) treats the nodes homogeneously during the calculation, (2) makes the integration center the root node, and (3) refers all motion to a common center (normally, but not necessarily, the Solar System barycenter). The methodology can be implemented recursively and can store intermediate results to yield savings in ephemerides evaluation.

Graph Coloring for the Evaluation of Sparse Jacobians and Hessians

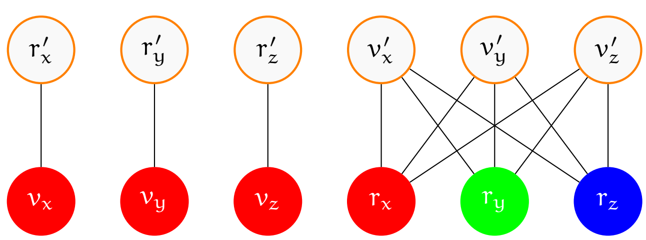
I will describe both the theoretical foundation and implementation of a utility that relies on graph coloring techniques for the numerical calculation of sparse Jacobians and Hessians. The theory is based on the original work of Curtis, Powell, and Reid; later generalized by Coleman and Moré; and extended and summarized by Gebremedhin, Manne, and Pothen: one can reduce the number of function evaluations required to calculate an sparse Jacobian if its columns are separated into structurally orthogonal groups, and finding such groups is equivalent to coloring a graph. For example: the Jacobian of the differential equations modeling the two body problem

$$\frac{d^2\mathbf{r}}{dt^2} = -\mu\frac{\mathbf{r}}{r^3} \tag{1}$$

can be separated into three groups (red, green, and blue) of structurally-orthogonal columns:

$$\text{structure}(\mathbf{J}) = \begin{bmatrix} 0 & 0 & 0 & \partial_{v_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \partial_{v_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \partial_{v_z} \\ \partial_{r_x} & \partial_{r_y} & \partial_{r_z} & 0 & 0 & 0 \\ \partial_{r_x} & \partial_{r_y} & \partial_{r_z} & 0 & 0 & 0 \\ \partial_{r_x} & \partial_{r_y} & \partial_{r_z} & 0 & 0 & 0 \end{bmatrix}. \tag{2}$$

Given a nominal value, one could evaluate the six columns in the Jacobian using three (as opposed to six) additional function evaluations: perturbing red (r_x, v_x, v_y, and v_z), then green (r_y), and finally blue (r_z). This choice of groups was obtained



from a distance-2 coloring of the variable-dependency graph: nodes separated by two edges must have different colors.

Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.