Optimasi Rute Grand Tour Tata Surya Menggunakan A* Search dan Dynamic Programming dengan Approximation Model Gravity Assist

An-Dafa Anza Avansyah 13524038[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524038@itb.ac.id, [2]andafaanzaa@gmail.com

***Abstrak*— Optimasi rute Grand Tour dimodelkan sebagai pencarian pada ruang status diskret menggunakan orbit lingkaran dua dimensi dan pendekatan gravity assist. A* Search, Dynamic Programming multilabel, dan Greedy dibandingkan pada skenario Voyager-like dari Bumi menuju Neptunus. Hasil pengujian menunjukkan bahwa A* dan Dynamic Programming menghasilkan rute lengkap sesuai urutan kunjungan, sedangkan Greedy cenderung menghasilkan biaya lebih tinggi akibat keputusan lokal. Model yang digunakan bersifat abstraksi komputasional, bukan simulasi astrodinamika presisi.**

***Kata kunci*—A Star Search, Dynamic Programming, Greedy, Gravity Assist, Grand Tour, Optimasi Rute, Voyager.**

I. PENDAHULUAN

Perkembangan komputasi memungkinkan berbagai persoalan dunia nyata dimodelkan sebagai persoalan pencarian dan optimasi. Dalam banyak kasus, keputusan terbaik tidak dapat ditentukan hanya dengan memilih pilihan yang tampak paling dekat atau paling murah pada saat itu, karena setiap keputusan dapat memengaruhi kondisi dan kemungkinan pilihan selanjutnya. Oleh karea itu, strategi algoritma seperti pencarian berbasis graf, heuristic, dan pemrograman dinamis menjadi penting untuk menyelesaikan persoalan yang memiliki ruang solusi besar dan saling bergantung.

Salah satu domain yang menarik untuk dikaji dari sudut pandang algoritmik adalah perencanaan rute wahana antariksa. Berbeda dengan rute di permukaan Bumi, lintasan antarplanet tidak hanya bergantung pada jarak antarobjek, tetapi juga pada pergerakan planet terhadap waktu. Sebuah planet yang tampak dekat secara geometris belum tentu menjadi tujuan transfer yang baik apabila posisi orbitnya tidak mendukung. Sebaliknya, planet tertentu dapat dimanfaatkan sebagai titik bantu apabila konfigurasi orbitnya memungkinkan wahana memperoleh perubahan energi melalui *flyby*.

GAMBAR 1. VOYAGER 1

Ekplorasi antariksa menunjukkan bahwa persoalan lintasan tidak hanya ditentukan oleh jarak geometris, tetapi juga oleh konfigurasi benda langit, waktu keberangkatan, dan perubahan energi orbit selama perjalanan. Salah satu contoh penting adalah misi Voyager. National Aeronautics and Space Administration (NASA) menjelaskan bahwa misi Voyager memanfaatkan konfigurasi langka planet luar untuk melakukan Grand Tour, yaitu kunjungan berurutan ke planet raksasa menggunakan gravitasi tiap planet untuk menuju planet berikutnya [1]. Teknik *gravity assist* memungkinkan wahana antariksa menambah atau mengurangi momentum sehigga energi orbitnya berubah tanpa mengandalkan pembakaran propelan besar[2]. Voyager 2 kemudia menjadi wahana yang mempelajari Jupiter, Saturnus, Uranus, dan Neptunus dari jarak dekat [3].

Dalam konteks pemodelan algoritma, masalah tersebut dapat dipandang sebagai pencarian rute pada ruang status. Setiap state merepresentasikan posisi wahana, waktu model, himpunan planet yang sudah dikunjungi, jumlah *flyby*, dan informasi kecepatan. Setiap perpindahan antarplanet memiliki biaya yang ditentukan oleh jarak, estimasi kebutuhan perubahan kecepatan, penalty *alignment*, waktu tempuh, jumlah *flyby*, serta keuntungan sederhana dari *gravity assist*.

Makalah ini tidak bertujuan membangun *simulator* astrodinamika presisi. Sebaliknya, makalah ini merancang

model komputasional sederhana agar strategi algoritma dapat diuji pada persoalan rute *interplanetary*. Model menggunakan orbit lingkaran dua dimensi, estimasi transfer berbasis biaya relatif, dan pendekatan *gravity assist* berbasis rotasi vektor kecepatan relative. Dengan model tersebut, tiga algoritma dibandingkan, yaitu A* Search, Dynamic Programming multilabel, dan Greedy *baseline*.

II. Landasan Teori

A. Tata Surya dan Gerak Orbit Planet

Tata Surya terdiri atas Matahari sebagai pusat gravitasi utama dan berbagai benda langit yang bergerak mengelilinginya, termasuk planet, satelit alami, asteoroid, komet, dan objek kecil lainnya. Dalam perencanaan lintasa antarplanet, posisi planet terhadap Matahari menjadi salah satu faktor penting karena planet tidak berada pada posisi tetap.

Pada makalah ini, planet dimodelkan sebagai benda yang bergerak pada orbit lingkaran dua dimensi terhadap matahari. Model tersebut merupakan penyederhanaan dari orbit sesunggunya yang berbentuk *elips*. Penyederhanaan ini dipilih agar persoalan dapat difokuskan pada strategi pencarian rute dan optimasi algoritmik[7], [8], bukan pada simulasi astrodinamika yang presisi. Setiap planet *p* memiliki tiga parameter utama, yaitu jari-jari orbit rata-rata R_p, periode orbit T_p, dan sudut awal $\theta_{0,p}$. Jari-jari orbit dinyatakan dalam *astronomical unit* (AU), yaitu satuan jarak yang secara umum merepresentasikan jarak rata-rata Bumi ke Matahari. Periode orbit menyatakan waktu yang diperlukan planet untuk mengelilingi Matahari satu kali.

Dengan asumsi orbit lingkaran dan kecepatan sudut konstan, posisi sudut planet *p* pada waktu *t* dapat dihitung sebagai berikut.

$$\theta_{p(t)} = \theta_{0,p} + \frac{2\pi t}{T_p}$$

Di mana $\theta_{p(t)}$ adalah posisi sudut planet pada waktu *t*, $\theta_{0,p}$ adalah sudut awal planet, dan T_p adalah periode orbit planet. Jika posisi planet direpresentasikan pada bidang Kartesisus dua dimensi, maka koordinat planet dapat dihitung dengan persamaan berikut.

$$x_{p(t)} = R_p \cos(\theta_{p(t)})$$

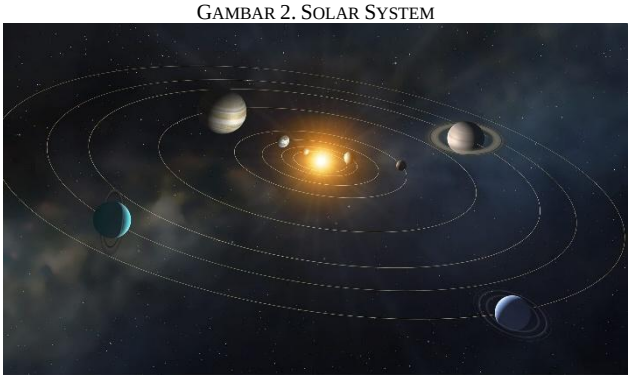
$$y_{p(t)} = R_p \sin(\theta_{p(t)})$$

Dengan demikian, posisi planet *p* pada waktu *t* dapat dinyatakan sebagai vektor posisi:

$$r_{p(t)} = (x_{p(t)}, y_{p(t)})$$

Model ini memungkinkan program menghitung posisi planet pada tahun tertentu. Dalam implementasi, waktu misi dinyatakan dalam satuan tahun model, sedangkan periode orbit planet dapat dikonversi dari hari ke tahun. Walaupun sederhana, model ini cukup untuk memperlihatkan bahwa konfigurasi planet berubah seiring waktu dan memengaruhi biaya transfer antarplanet.

Gambar 2. Solar System



Selain posisi, program juga membutuhkan pendekatan kecepatan orbit planet. Untuk orbit lingkaran, kecepatan orbit dapat dipandang sebagai vektor tangensial terhadap lintasan orbit. Besar kecepatan orbit relatif dapat dihitung dari keliling orbit dibagi periode orbit.

$$|v_p| = \frac{2\pi R_p}{T_p}$$

Arah vektor kecepatan tegak lurus terhadap vektor radius planet. Jika planet bergerak berlawanan arah jarum jam, maka pendekatan vektor kecepatannya dapat ditulis sebagai berikut.

$$|v_{p(t)}| = |v_p|(-\sin(\theta_{p(t)}), \cos(\theta_{p(t)}))$$

Kecepatan ini tidak digunakan sebagai prediksi fisika presisi, tetapi sebagai parameter pendekatan untuk menghitung efek relative *gravity assist* dan perbedaan kecepatan antartransfer.

B. Gravity Assist

Gravity assist adalah teknik *flyby* yang memanfaatkan gerak planet untuk mengubah energi heliosentris wahana. Dalam kerangka relative terhadap planet, wahana dapat dipandang datang dengan suatu vektor kecepatan realtif, kemudian keluar dengan besar kecepatan relatif yang mendekati sama tetapi arah yang berubah akibat interaksi gravitasi. Namun, ketika vektor kecepatan tersebut dilihat kembali dalam kerangka Matahari, perubahan arah tersebut dapat menghasilkan perubahan besar kecepatan heliosentris wahana. Dengan kata lain, planet dapat bertindak seperti "perantara momentum" yang membuat wahana memperoleh atau kehilangan energi orbit tanpa harus menggunakan pembakaran propelan besar[2], [7].

Secara konseptual, terdapat dua kerangka acuan yang digunakan untuk menjelaskan *gravity assist*. Kerangka pertama adalah kerangka heliosentris, yaitu kerangka

acuan yang melihat posisi dan kecepatan relatif terhadap Matahari. Kerangka kedua adalah kerangka relatif planet, yaitu kerangka acuan yang bergerak bersama planet yang sedang dilewati. Jika v_{in} adalah kecepatan wahana sebelum *flyby* dalam kerangka Matahari dan v_p adalah kecepatan planet, maka kecepatan relatif wahana terhadap planet dapat ditulis sebagai berikut.

$$v_{rel,in} = v_{in} - v_p$$

Dalam model sederhana program, *flyby* dimodelkan sebagai rotasi vektor kecepatan relatif sebesar sudut assist tertentu. Sudut assist tidak dihitung dari massa planet, radius *flyby*, atau parameter gravitasi planet, tetapi dipilih dari himpunan sudut diskret yang telah ditentukan pada konfigurasi misi. Jika rotasi vektor sebesar α dinyatakan sebagai *rotate(v, α)*, maka kecepatan relatif setelah *flyby* adalah:

$$v_{rel,out} = \text{rotate}(v_{rel,in}, \alpha)$$

Kemudian kecepatan tersebut dikembalikan ke kerangkan Matahari dengan menjumlahkannya kembali dengan kecepatan planet.

$$v_{out} = v_p + v_{rel,out}$$

Nilai keuntungan *assist* (*assist gain*) dihitung dari selisih besar kecepatan heliosentris sesudah dan sebelum flyby.

$$\text{gain} = |v_{out}| - |v_{in}|$$

Jika *gain* bernilai positif, *flyby* tersebut dianggap memberikan tambahan kecepatan heliosentris dalam model. Jika *gain* bernilai negatif, *flyby* dianggap mengurangi kecepatan heliosentris. Pada program, nilai ini digunakan sebagai salah satu komponen fungsi biaya. Semakin besar *assist gain*, biaya rute dapat berkurang sesuai bobot yang ditentukan pada konfigurasi misi.

Perlu ditekankan bahwa model ini merupakan pendekatan komputasional. Model tidak menghitung lintasan hiperbolik sebenarnya, parameter gravitasi planet, batas radius *flyby*, pengaruh banyak benda langit secara simultan, maupun koreksi orbit. Dengan demikian, model *gravity assist* pada makalah ini berfungsi sebagai abstraksi algoritmik agar efek *flyby* dapat dimasukkan ke dalam proses pencarian rute.

C. *Transfer Antarplanet dan Fungsi Biaya*

Transfer antarplanet pada program direpresentasikan sebagai perpindahan dari satu planet sumber ke planet tujuan. Setiap transfer memiliki beberapa atribut, yaitu waktu keberangkatan, waktu kedatangan, waktu tunggu, waktu tempuh, jarak antarplanet, estimasi perubahan kecepatan, penalti *alignment*, *assist gain*, dan biaya leg. Satu transfer dapat dipandang sebagai sisi berbobot pada graf ruang status.

Jarak antara planet sumber a dan planet tujuan b pada waktu tertentu dapat dihitung dari vektor posisi kedua planet. Jika posisi planet *a* adalah $r_{a(t)}$ dan posisi planet *b* adalah $r_{b(t')}$, maka jarak Euclidean antarplanet adalah:

$$d(a,b) = \left\| r_{b(t')} - r_{a(t)} \right\|$$

Dalam model nyata, lintasan antarplanet tidak berupa garis lurus karena wahana bergerak dalam medan gravitasi Matahari. Namun, untuk kebutuhan algoritmik, jarak Euclidean dipakai sebagai salah satu komponen biaya relatif. Semakin jauh planet tujuan dari planet sumber, semakin besar biaya transfer yang diberikan oleh model.

Selain jarak, program juga menggunakan delta-v proxy. Delta-v dalam astrodinamika menyatakan perubahan kecepatan yang diperlukan wahana untuk melakukan manuver. Dalam program ini, delta-v proxy bukan delta-v fisik sebenarnya, melainkan skor relatif yang merepresentasikan kesulitan perubahan kecepatan berdasarkan kondisi model. Delta-v proxy digunakan agar transfer dengan perbedaan kondisi kecepatan yang besar diberi biaya lebih tinggi[7].

Komponen lain yang digunakan adalah *alignment penalty*. Penalti ini merepresentasikan ketidaksesuaian konfigurasi posisi planet. Dalam konteks sederhana, transfer dianggap lebih baik apabila planet sumber dan tujuan berada dalam konfigurasi yang mendukung perpindahan. Sebaliknya, apabila konfigurasi posisi kurang sesuai, transfer diberi penalti lebih besar. Dengan demikian, program tidak hanya memilih planet berdasarkan jarak rata-rata, tetapi juga mempertimbangkan posisi planet pada waktu model tertentu.

Secara umum, biaya satu leg dalam program dapat dituliskan sebagai kombinasi berbobot dari beberapa komponen.

$$C_{leg} = w_d\, \Delta v_{proxy} + w_t\, time + w_a\, alignment + w_f\, flyby + w_g\, gain$$

di mana C_{leg} adalah biaa satu transfer, Δv_{proxy} adalah estimasi perubahan kecepatan relatif, *time* adalah waktu tempuh, *alignment* adalah penalti konfigurasi, *flyby* adalah penalti jumlah *flyby*, dan gain adalah keuntungan *gravity assist*. Parameter w_d, w_t, w_a, w_f, dan w_g adalah bobot yang dapat diatur melalui file konfigurasi misi.

Biaya total rute diperoleh dari penjumlahan biaya seluruh leg.

$$C_{total} = \sum C_{leg}$$

Tujuan optimasi adalah menemukan rute yang memenuhi batasan misi dan memiliki nilai C_{total} sekecil mungkin.

D. *Graf dan State Space Search*

Persoalan pencarian rute Grand Tour dapat direpresentasikan sebagai pencarian pada graf berbobot. Pada graf biasa, simpul dapat menyatakan planet dan sisi dapat menyatakan perpindahan antarplanet. Namun, untuk persoalan ini, representasi tersebut belum cukup karena biaya transfer tidak hanya bergantung pada planet asal dan tujuan, tetapi juga pada waktu, urutan kunjungan sebelumnya, jumlah *flyby*, dan kondisi kecepatan wahana[10].

Oleh karena itu, program menggunakan konsep state space. Setiap state merepresentasikan kondisi parsial misi pada suatu titik pencarian. Secara umum, state dapat dinyatakan sebagai:

$$S = (p, t, V, k, v)$$

dengan *p* sebagai planet saat ini, *t* sebagai waktu model, *V* sebagai himpunan atau progres planet yang telah dikunjungi, *k* sebagai jumlah *flyby* yang telah digunakan, dan *v* sebagai pendekatan kecepatan masuk wahana. Dari suatu *state*, algoritma dapat membangkitkan *state* baru dengan memilih planet tujuan berikutnya dan pilihan waktu tunggu tertentu.

Jika state S_i berpindah ke state S_j, maka perpinfahan tersebut memiliki biaya $\underline{C}(S_i, S_j)$ yang dihitung menggunakan model transfer. Dengan demikian, pencarian rute Grand Tour dapat dipandang sebagai pencarian lintasan minimum dari state awal menuju state tujuan pada graf ruang status.

State awal adalah kondisi ketika wahana berada di planet awal pada tahun peluncuran. *State* tujuan adalah kondisi ketika wahana telah mencapai planet target dan memenuhi seluruh *required visit*. Pada mode ordered, required visit harus dipenuhi sesuai urutan yang diberikan. Pada mode unordered, *required visit* dapat dipenuhi dalam urutan bebas selama seluruh planet wajib dikunjungi.

Ruang status dapat tumbuh sangat besar karena setiap *state* dapat bercabang ke beberapa planet kandidat dan beberapa pilihan waktu tunggu. Jika terdapat *n* kandidat planet dan *q* pilihan waktu tunggu pada setiap langkah, maka jumlah cabang kasar pada satu level dapat mencapai $n \times q$. Karena itu, pencarian membutuhkan strategi untuk memilih state yang lebih menjanjikan, menghindari eksplorasi *state* berulang, dan membatasi ruang pencarian.

E. A Search*

A* Search adalah algoritma pencarian graf berbobot yang menggunakan fungsi evaluasi untuk menentukan simpul mana yang akan dieksplorasi terlebih dahulu. Algoritma ini menggabungkan biaya aktual yang sudah ditempuh dengan estimasi biaya menuju tujuan. Fungsi evaluasi A* ditulis sebagai berikut.

$$f(n) = g(n) + h(n)$$

Pada persamaan tersebut, *g(n)* adalah biaya dari state awal sampai *state n*, sedangkan *h(n)* adalah estimasi biaya dari *state n* menuju tujuan. *State* dengan nilai *f(n)* terkecil diprioritaskan untuk diekspansi lebih dahulu. Dengan cara ini, A* tidak hanya mempertimbangkan biaya yang telah dikeluarkan, tetapi juga perkiraan prospek *state* tersebut terhadap tujuan akhir[4].

Pada program Grand Tour Optimizer, A* digunakan untuk mengeksplorasi ruang status misi. Setiap elemen pada *priority queue* menyimpan state misi, biaya kumulatif, rute parsial, daftar leg, waktu kedatangan, dan informasi kecepatan. Ketika sebuah state diekspansi, program membangkitkan kandidat transfer menuju planet lain yang diizinkan. Untuk setiap kandidat, program menghitung biaya leg, memperbarui waktu, memperbarui progres *required visit*, dan memasukkan *state* baru ke *priority queue* apabila masih memenuhi batasan misi.

Heuristik yang digunakan bersifat praktis. Program memperkirakan sisa biaya berdasarkan gap radial menuju *required visit* yang belum dipenuhi dan target. Jika r_p adalah radius orbit planet saat ini dan r_q adalah radius orbit planet tujuan berikutnya yang harus dicapai, maka salah satu bentuk estimasi sederhana dapat dipandang sebagai:

$$h(n) = \beta \left| r_q - r_p \right|$$

dengan β sebagai faktor skala biaya. Untuk beberapa planet tersisa, estimasi dapat dijumlahkan berdasarkan urutan required visit atau jarak radial menuju target. Heuristik ini membantu mengarahkan pencarian ke state yang lebih dekat secara radial terhadap progres misi.

Secara umum, kompleksitas A* bergantung pada jumlah *state* yang dibangkitkan dan jumlah sisi yang dievaluasi. Jika jumlah state yang mungkin adalah $|S|$ dan jumlah transisi adalah $|E|$, maka kompleksitas dapat ditulis secara kasar sebagai:

$$O(|E| \log |S|)$$

F. Dynamic Programming

Dynamic Programming adalah strategi algoritma yang menyelesaikan masalah dengan memecahnya menjadi submasalah yang saling bertumpang tindih. Hasil submasalah disimpan agar tidak perlu dihitung ulang. Strategi ini cocok untuk persoalan yang memiliki struktur keputusan bertahap dan prinsip optimalitas, yaitu solusi optimal suatu masalah dapat dibangun dari solusi optimal submasalahnya[5].

Pada persoalan Grand Tour, Dynamic Programming dapat digunakan untuk menentukan urutan kunjungan planet. Jika hanya nama planet yang diperhatikan, state DP sederhana dapat ditulis sebagai:

$$dp[mask][last] =$$

biaya minimum untuk mengunjungi planet

Mask adalah representasi bit dari planet-planet wajib yang sudah dikunjungi, sedangkan last adalah planet terakhir yang dikunjungi. Transisi DP dapat ditulis sebagai berikut.

$$dp[mask \cup \{next\}][next] = \min(\mathrm{dp}[\mathrm{mask} \cup \{\mathrm{next}\}][\mathrm{next}], \mathrm{dp}[\mathrm{mask}[\mathrm{last}] + \mathrm{C}(\mathrm{last}, \mathrm{next}))$$

Model ini mirip dengan pendekatan yang digunakan pada variasi Traveling Salesman Problem. Namun, pada program ini, kondisi misi tidak cukup direpresentasikan oleh mask dan last saja. Waktu kedatangan dan vektor kecepatan juga memengaruhi biaya transfer berikutnya. Dua rute yang berakhir di planet yang sama dengan mask yang sama dapat memiliki kualitas berbeda apabila waktu kedatangan dan kecepatannya berbeda[6].

Untuk mengatasi hal tersebut, implementasi program menggunakan Dynamic Programming multilabel. Dalam pendekatan ini, satu state DP tidak hanya menyimpan satu nilai terbaik, tetapi menyimpan beberapa label kandidat. Setiap label memuat biaya, tahun kedatangan, rute parsial, daftar leg, dan vektor kecepatan. Dengan demikian, state yang sedikit lebih mahal tetapi tiba lebih awal atau memiliki kondisi kecepatan lebih menguntungkan tidak langsung dibuang.

Secara konseptual, label dapat ditulis sebagai:

$$L = (cost, t, route, legs, v)$$

Untuk setiap state, jumlah label dibatasi oleh parameter tertentu agar ukuran frontier tidak tumbuh terlalu besar. Pembatasan ini membuat DP multilabel menjadi kompromi antara kelengkapan eksplorasi dan efisiensi komputasi. Jika jumlah planet wajib adalah *m* dan jumlah label maksimum per state adalah l, maka jumlah state utama DP adalah sekitar $2^m \times m$, sedangkan jumlah label yang diproses dapat mencapai:

$$O(2^m \times m \times l)$$

G. *Greedy Algorithm*

Greedy Algorithm adalah strategi yang memilih keputusan terbaik secara lokal pada setiap langkah. Pada persoalan rute, Greedy dapat memilih planet berikutnya berdasarkan biaya leg terkecil dari state saat ini. Strategi ini memiliki kelebihan dari sisi kesederhanaan dan kecepatan karena tidak mengeksplorasi banyak kemungkinan secara mendalam[6].

Secara umum, langkah Greedy pada program dapat dinyatakan sebagai berikut. Dari state saat ini *S*, bangkitkan seluruh kandidat transfer *K*. Kemudian pilih kandidat *k* terbaik yang memiliki biaya lokal minimum:

$$k^* = \mathrm{argmin}\ C_{lrg}(S, k), untuk\ setiap\ k\ \in K$$

Setelah kandidat dipilih, state diperbarui dan proses berulang sampai target tercapai atau tidak ada kandidat layak. Greedy tidak menyimpan banyak alternatif masa depan. Akibatnya, keputusan yang tampak murah pada suatu langkah dapat menyebabkan rute berikutnya menjadi lebih mahal atau bahkan tidak *feasible*.

III. PEMODELAN MASALAH

A. *Ruang Lingkup dan Asumsi*

Model dibatasi pada orbit planet berbentuk lingkaran, koplanar, dan dua dimensi. Posisi planet dihitung dari radius orbit, periode orbit, serta sudut awal, sedangkan data *ephemeris* nyata, *eksentrisitas*, *inklinasi*, *finite burn*, efek *n-body*, dan *patched-conic* tidak dimodelkan. Nilai delta-v proxy, *assist gain*, *alignment penalty*, dan total cost merupakan skor komparatif antarrute, bukan besaran operasional untuk misi nyata.

Penyederhanaan tersebut dipilih agar fokus penelitian berada pada perancangan state, fungsi biaya, dan perbandingan strategi algoritma. Seluruh solver menggunakan model transfer yang sama sehingga hasil eksperimen tetap dapat dibandingkan secara konsisten di dalam ruang lingkup model.

B. *Data Planet dan Model Orbit Dua Dimensi*

Setiap planet *p* direpresentasikan oleh radius orbit R_p dalam AU, periode orbit T_p, dan sudut awal $\theta_{0,p}$. Nilai radius dan periode yang digunakan merupakan parameter rata-rata yang disederhanakan dari data orbit planet. Tabel 1 memuat data utama yang digunakan program.

TABEL 1. KONFIGURASI PLANET

Planet	Radius	Periode (hari)
Mercury	0.387	87.969
Venus	0.723	224.701
Earth	1.000	365.256
Mars	1.524	686.980
Jupiter	5.203	4332.589
Saturn	9.537	10759.220
Uranus	19.191	30685.400
Neptune	30.070	60189.000

C. *Ordered dan Unordered Grand Tour*

Grand Tour pada program dapat dijalankan dalam dua mode, yaitu ordered dan unordered. Pada mode ordered, planet wajib harus dikunjungi sesuai urutan yang diberikan pada konfigurasi misi. Misalnya, untuk misi Voyager-like, urutan required visit adalah:

$$\mathrm{Jupiter} \rightarrow \mathrm{Saturn} \rightarrow \mathrm{Uranus} \rightarrow \mathrm{Neptune}$$

Dalam mode ini, state hanya dianggap mengalami progres apabila planet berikutnya sesuai dengan urutan required visit. Mode ordered cocok untuk meniru skenario historis

atau skenario misi yang memiliki urutan kunjungan tertentu.

Pada mode unordered, planet wajib dapat dikunjungi dalam urutan bebas. State menyimpan himpunan planet yang telah dikunjungi, dan misi dianggap selesai apabila seluruh planet wajib sudah tercakup serta target akhir tercapai. Mode ini memberi fleksibilitas lebih besar kepada algoritma untuk mencari biaya total lebih rendah, tetapi rute yang dihasilkan dapat berbeda dari rute historis atau skenario yang diinginkan.

Perbedaan kedua mode tersebut penting dalam analisis. Ordered mode menekankan kesesuaian dengan skenario Grand Tour tertentu, sedangkan unordered mode menekankan pencarian kombinasi rute dengan biaya lebih rendah dalam ruang solusi yang lebih bebas.

D. Konfigurasi Misi

Masukan utama program berupa *mission* JSON. Konfigurasi tersebut menentukan planet awal, planet target, daftar *required visit*, planet yang boleh menjadi *flyby*, tahun peluncuran, batas durasi misi, jumlah *flyby* maksimum, bobot fungsi biaya, sudut *assist*, kandidat waktu tunggu, dan mode kunjungan. Mode ordered mengharuskan planet pada *required visit* dikunjungi sesuai urutan, sedangkan mode unordered memperlakukan *required visit* sebagai himpunan sehingga *solver* dapat memilih urutan sendiri.

TABEL 2. KONFIGURASI JSON MISI

Parameter	Nilai
Start planet	Earth
Target planet	Neptune
Required visit	Jupiter-Saturn-Uranus-Neptune
Launch year	1977.0
Max mission years	25.0
Wait years	0.0; 0.5; 1.0
Assist angles	-60; -30; 0; 30; 60
Visit mode utama	-ordered

IV. IMPLEMENTASI PROGRAM

A. Arsitektur program

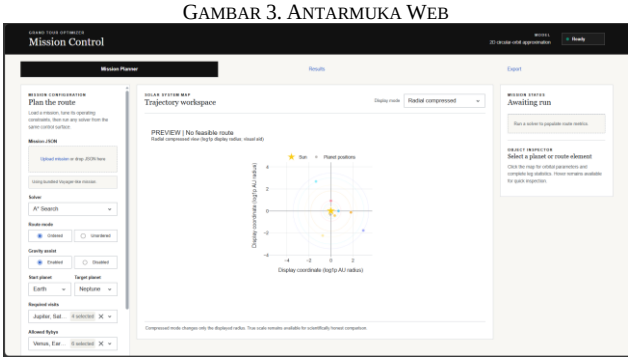
Program diimplementasikan menggunakan Python. Modul physics memuat model orbit, transfer, dan gravity assist. Modul solvers memuat implementasi A*, DP, dan Greedy. Modul backend menyediakan validasi konfigurasi dan fungsi eksekusi misi yang dipakai bersama oleh CLI dan web dashboard. Modul output menyimpan hasil dalam JSON, CSV, PNG, dan HTML. Modul visualization membuat plot statis dan interaktif, sedangkan app.py membangun antarmuka web Grand Tour Mission Control.

```
grand-tour-optimizer/
├── app.py
├── data/
│   ├── planets.json
│   └── missions/
├── src/
│   └── grand_tour_optimizer/
│       ├── backend.py
│       ├── physics/
│       ├── solvers/
│       ├── experiments/
│       └── visualization/
├── outputs/
└── tests/
```

B. Antamuka Web

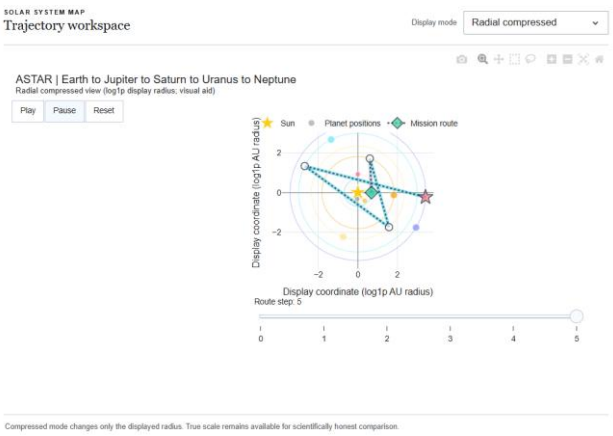
Antarmuka web menyediakan workflow utama bagi pengguna. Pengguna dapat mengunggah mission JSON, memilih *solver*, mengatur *ordered* atau *unordered* mode, mengaktifkan atau menonaktifkan *gravity assist*, menentukan planet awal dan target, memilih *required visit* serta *allowed flyby*, lalu menjalankan misi. Hasil ditampilkan melalui kartu metrik, tabel leg, catatan *feasibility*, dan visualisasi interaktif tata surya.

GAMBAR 3. ANTARMUKA WEB



Visualisasi peta mendukung beberapa mode tampilan, yaitu *true scale*, *radial compressed*, *inner* Solar System, *outer* Solar System, dan *route focus*. Mode *radial compressed* memakai transformasi tampilan agar planet dalam tetap terlihat ketika Neptunus ikut diplot. Transformasi ini hanya mengubah radius tampilan dan tidak mengubah perhitungan *solver*.

GAMBAR 4. VISUALISASI RUTE



C. Format Output

Setiap eksekusi menghasilkan ringkasan rute, total cost, waktu perjalanan, total delta-v proxy, total assist gain, jumlah expanded states, runtime, daftar leg, dan metadata konfigurasi. Hasil dapat diekspor sebagai JSON untuk analisis ulang, CSV untuk tabel, PNG untuk gambar statis, dan HTML untuk visualisasi interaktif mandiri.

V. Pengujian Dan Analisis

A. Konfigurasi Pengujian

Pengujian dilakukan menggunakan konfigurasi Voyager-like ordered Grand Tour dengan planet awal Earth, *target* Neptune, *required visit* Jupiter, Saturn, Uranus, dan Neptune, serta *allowed flyby* Venus, Earth, Mars, Jupiter, Saturn, dan Uranus. Tahun peluncuran model adalah 1977, batas durasi misi 25 tahun, batas *flyby* 6, kandidat waktu tunggu 0.0, 0.5, dan 1.0 tahun, serta sudut *assist* -60, -30, 0, 30, dan 60 derajat.

B. Studi Kasus Ordered Grand Tour

Pada studi kasus utama, A* dengan *gravity assist* menghasilkan rute Earth → Jupiter → Saturn → Uranus → Neptune. Total cost yang diperoleh adalah 365.797, waktu perjalanan 12.391 tahun, total delta-v proxy 35.218, total assist gain 1.643, *expanded states* 6006, dan runtime 0.320 detik pada output final. Rute dinyatakan *complete* dan *feasible under the simplified mission constraints*.

Tabel 3. Hasil Percobaan

Leg	Waktu	Jarak	Cost
Earth-Jupiter	2.455	4.961	37.980
Jupiter-Saturn	2.490	12.863	71.702
Saturn-Uranus	3.618	28.284	100.131
Uranus-Neptune	3.828	48.349	155.984

C. Perbandingan Solver

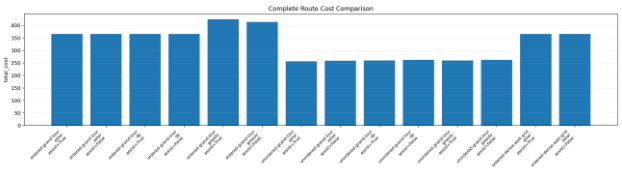
Tabel 4 merangkum sebagian hasil eksperimen. Pada ordered Grand Tour, A* dan DP menghasilkan rute yang sama sesuai *required sequence*. DP memiliki *expanded states* lebih kecil karena pada mode ordered masalah menjadi struktur progres linear dan DP dapat langsung memperbarui *frontier* menuju *required visit* berikutnya. Greedy menyelesaikan misi, tetapi memilih rute Earth → Jupiter → Mars → Earth → Saturn → Uranus → Neptune. Rute ini tetap *feasible* dalam batasan sederhana, tetapi cost dan waktu tempuh lebih besar karena Greedy memilih keputusan lokal yang menambah *flyby*.

Tabel 4. Perbandingan Solver

Mode/Solver	Assist	Cost	States
Ord.-A*	Ya	365.80	6006
Ord.-A*	Tidak	365.69	8237
Ord.-DP	Ya	365.80	96
Ord.-GREEDY	Ya	424.88	57
Unord.-A*	Ya	256.53	4160
Unord.-DP	Ya	260.18	465
Unord.-GREEDY	Ya	260.18	63

Pada unordered Grand Tour, solver bebas menentukan urutan required visit. A* dengan gravity assist memperoleh rute Earth -> Jupiter -> Uranus -> Saturn -> Neptune dengan cost 256.532 dan waktu 16.071 tahun. Cost ini lebih rendah daripada ordered mode, tetapi urutan tersebut tidak mengikuti narasi Grand Tour Voyager-like. Hal ini menunjukkan bahwa constraint urutan memiliki pengaruh besar terhadap rute yang dianggap optimal.

Gambar 5. Perbandingan Cost Solver



D. Dampak gravity Assist

Hasil eksperimen menunjukkan bahwa pengaruh *gravity assist* bergantung pada mode dan model biaya. Pada *unordered mode*, A* dengan *assist* memperoleh *cost* 256.532, sedangkan tanpa assist memperoleh *cost* 258.955. Dalam kasus ini, *assist* menurunkan *cost* dan juga mengurangi waktu perjalanan dari 16.571 menjadi 16.071 tahun. Pada *ordered mode*, *cost* dengan *assist* sedikit lebih tinggi daripada tanpa *assist*, yaitu 365.797 dibandingkan 365.687. Hal ini terjadi karena fungsi biaya tidak hanya memperhitungkan *assist gain*, tetapi juga perubahan delta-v proxy dan komponen transfer lain setelah vektor kecepatan berubah.

Temuan tersebut penting karena menunjukkan bahwa *approximation model* tidak menjamin setiap *gravity assist* menurunkan *cost* total. *Gravity assist* mengubah kondisi kecepatan yang kemudian memengaruhi leg berikutnya. Dalam model sederhana ini, assist berguna sebagai komponen eksperimen algoritmik, bukan klaim fisik bahwa semua *flyby* pasti lebih baik.

E. Pengaruh Grid Waktu Tunggu

Eksperimen ordered-dense-wait-grid menunjukkan bahwa penambahan kandidat waktu tunggu meningkatkan jumlah *state* yang diekspansi. Pada A* dengan *assist*, *expanded states* meningkat dari 6006 menjadi 13834. Tanpa *assist*, *expanded states* meningkat dari 8237 menjadi 18554. Rute dan *cost* terbaik tidak berubah pada konfigurasi ini, tetapi runtime meningkat. Hasil ini menggambarkan *trade-off* diskretisasi: semakin banyak kandidat waktu tunggu, ruang pencarian semakin luas, tetapi peluang menemukan rute alternatif juga meningkat.

VI. KESIMPULAN

Pemodelan optimasi rute Grand Tour tata surya menunjukkan bahwa persoalan perjalanan antarplanet dapat direpresentasikan sebagai pencarian pada ruang status yang memuat posisi, waktu, urutan kunjungan, jumlah *flyby*, dan kondisi kecepatan. Penyederhanaan orbit menjadi model dua dimensi, penggunaan fungsi biaya relatif, serta pendekatan *gravity assist* memungkinkan persoalan yang kompleks tersebut dianalisis dalam kerangka strategi algoritma.

Perbandingan beberapa pendekatan memperlihatkan bahwa setiap algoritma memiliki karakteristik yang berbeda. A* Search mampu mengarahkan pencarian menggunakan informasi heuristik dan batasan ruang status, Dynamic Programming efektif dalam memanfaatkan struktur submasalah dan progres kunjungan, sedangkan Greedy menghasilkan keputusan dengan cepat tetapi dapat terjebak pada pilihan lokal yang kurang menguntungkan secara keseluruhan. Hasil tersebut menegaskan bahwa pemilihan strategi algoritma sangat dipengaruhi oleh bentuk state, batasan misi, serta tujuan optimasi yang digunakan.

Pendekatan *gravity assist* berbasis rotasi vektor kecepatan relatif memberikan abstraksi yang cukup untuk mengamati pengaruh flyby terhadap biaya perjalanan dalam model komputasional. Meskipun demikian, hasil yang diperoleh tetap bersifat relatif dan tidak dapat disamakan dengan perhitungan astrodinamika nyata. Pengembangan lebih lanjut dapat dilakukan melalui penggunaan data *ephemeris*, orbit *elips*, model transfer yang lebih akurat, perhitungan *flyby* berdasarkan parameter gravitasi planet, serta pengujian pada konfigurasi misi dan fungsi objektif yang lebih beragam.

VII. LAMPIRAN

Github program: An-Dafa/grand-tour-optimizer: A simplified interplanetary Grand Tour route optimizer using A* Search, Dynamic Programming, and gravity assist approximation

VII. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih sebesar-besanya kepada:

1. Tuhan Yang Maha Esa,
2. Orang tua penulis,
3. Dosen pengampu mata kuliah Strategi Algoritma
4. Changli

Yang senantiasa mendukung penulis dalam penyusunan makalah dan pembuatan program ini.

REFERENSI

[1] NASA Science, "The Grand Tour," National Aeronautics and Space Administration, Apr. 8, 2024. [Online]. Accessed: Jun. 15, 2026.

[2] NASA Science, "Basics of Spaceflight: A Gravity Assist Primer," National Aeronautics and Space Administration, Nov. 4, 2024. [Online]. Accessed: Jun. 15, 2026.

[3] NASA Science, "Voyager 2," National Aeronautics and Space Administration. [Online]. Accessed: Jun. 15, 2026.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[5] R. Bellman, Dynamic Programming. Princeton, NJ, USA: Princeton University Press, 1957.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[7] R. R. Bate, D. D. Mueller, J. E. White, and W. W. Saylor, Fundamentals of Astrodynamics, 2nd ed. Mineola, NY, USA: Dover Publications, 2020.

[8] NASA Science, "Orbits and Kepler's Laws," National Aeronautics and Space Administration, May 2, 2024. [Online]. Accessed: Jun. 18, 2026.

[9] Jet Propulsion Laboratory, "Planetary Physical Parameters," Solar System Dynamics, California Institute of Technology. [Online]. Accessed: Jun. 18, 2026.

[10] S. Irnich and G. Desaulniers, "Shortest path problems with resource constraints," in Column Generation, G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. New York, NY, USA: Springer, 2005, pp. 33–65, doi: 10.1007/0-387-25486-2_2.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

An-Dafa Anza Avansyah
13524038