BAB II
KAJIAN TEORI

2.1 Kajian Penelitian Sebelumnya

Traveling Salesman Problem (TSP) merupakan permasalahan optimasi yang cukup sulit untuk dipecahkan. Penyelesaian permasalahan ini membutuhkan algoritma yang mengharuskan mencari semua kemungkinan solusi. Permasalahan ini diibaratkan seorang *sales* yang ditugaskan untuk menawarkan produk yang dijualnya ke beberapa kota tujuan. *Sales* mengunjungi setiap kota tujuan tepat satu kali dan kemudian kembali ke kota asal, sehingga menghasilkan suatu jalur. Penelitian TSP sebelumnya telah dilakukan dengan studi kasus masing-masing, berikut pemaparannya.

Penelitian yang pertama adalah kasus penyelesaian *Traveling Salesman Problem* menggunakan algoritma g*reedy*. TSP mengharuskan melakukan perhitungan terhadap semua kemungkinan rute terpendek. Pada algoritma *brute-force*, untuk *n* kota yang harus dikunjungi maka menghasilakn *n!* kombinasi kota yang kemudian dibandingkan jarak rute setiap kombinasi. Dengan memakai algoritma ini, waktu komputasi yang dibutuhkan akan bertambah seiring jumlah kota yang harus dikunjungi juga bertambah. Penelitian ini menawarkan solusi penyelesaian TSP menggunakan algoritma *greedy* yang menggunakan prinsip pencarian jalur terpendek dengan fungsi seleksi. Algoritma *greedy* digunakan untuk menentukan lintasan linier, yang menghasilkan kemampuan menemukan solusi dengan jumlah node yang banyak dapat dilakukan dengan cepat. Dalam menyelesaikan TSP algoritma *brute-force* dan *dynamic programming* diyakini selalu mendapatkan solusi yang paling optimal, namun memiliki kelemahan dalam hal kecepatan komputasi ketika menemui banyak node yang harus dilalui. Sementara algoritma *greedy* menjanjikan solusi dan kecepatan komputasi yang jauh lebih cepat dibanding dua algortima di atas, meskipun tidak selalu menghasilkan solusi yang paling optimal.

Penelitian kedua mengenai penerapan Algoritma Genetika pada studi kasus antar jemput *laundry*. Usaha *laundry* sudah menjadi kebutuhan hiduo bagi masyarakat. Usaha ini makin berkembang sehingga menjadikan persaingan usaha semakin tinggi yang membuat penyedia layanan *laundry* memberikan inovasi layanan, seperti antar jemput cucian. Namun dalam pelanggan memiliki waktu tertentu dalam menyerahkan dan menerima cucian mereka, sehingga ada parameter waktu yang dihabiskan dan perkiraan rute mana yang akan diambil oleh sopir antar jemput *laundry*. Studi kasus ini mengambil pendekatan *Vahicle Routing*

Problem with Time Windows (VRPTW). VRPTW adalah masalah pencarian jalur yang akan dilalui dengan tujuan mencari rute tercepat atau terpendek dengan kendala tambahan berupa adanya *time windows* pada setiap pelanggan. Ketersediaan waktu pada masing-masing pelanggan dapat berbeda dan dinyatakan dalam selang waktu berupa batas waktu awal sampai akhir pelayanan pelanggan tersebut (Gambardella, 1999). Penyelesaian masalah ini menggunakan Algoritma Genetika yang pertama kali dikembangkan oleh John Holland dari Universitas Michigan tahun 1975 (Priambodo, 2016) merupakan algoritma yang memanfaatkan proses evolusi yang dikemukakan Charles Darwin. Algoritma Genetika memungkinkan tidak selalu mencapai hasil terbaik tetapi seringkali memecahkan masalah dengan cukup baik (Suprayogi, 2014). Penulis penelitian ini menguji lewat beberapa uji coba menggunakan beberapa metode. Metode yang digunakan menggunakan Metode Seleksi *Rouleter wheel* dan Elitis, Uji Coba Ukuran Generasi, Uji Coba Ukuran Populasi, dan Uji Coba Kombinasi Probabilitas *Crossover* dan Mutasi. Dengan beberapa metode ujicoba dihasilkan sebuah kesimpulan bahwa Algoritma Genetika dapat menyelesaikan VRPTW dengan kasus antar jemput laundry dengan menggunakan metode Seleksi *Rouleter wheel* dan Elitis karena metode ini lebih baik dan stabil.

Kajian selanjutnya tentang penentuan rute pengiriman truk minyak di Bangkok, Thailand. Perusahaan logistik SME yang menyediakan transportasi untuk mengirimkan minyak dari PTT *Oil warehouse* ke stasiun pengisian bahan bakar di penjuru Thailand. Truk memulai perjalanannya setiap pukul 9 malam hingga pukul 4 pagi dan mengirimkan minyak ke beberapa stasiun pengisian. Tujuan penelitian ini adalah untuk mengurangi biaya transportasi dalam pengiriman minyak dengan cara mengurangi jarak tempuh rute truk pengirim. Penelitian ini dilakukan untuk menyelesaikan masalah rute truk dengan pendekatan *Multiple Traveling Salesman Problem* (MTSP) yang dikemukakan oleh Lawler, Lenstra, Kan dan Shmyos pada tahun 1995 yang dimodifikasi untuk studi kasus ini (Lawler, 1995). Penyelesaian masalah ini digambarkan dengan membuat *matrix* yang berisi jarak setiap stasiun pengisian yang harus di kirim oleh *m* truk yang dapat memberikan jawaban lebih baik dibanding sebelumnya, namun tidak menjanjikan solusi paling optimal setiap truk.

Penelitian keempat tentang pembuatan *itinerary* wisata di Yogyakarta. Studi kasus ini diambil untuk membantu wisatawan dalam merancang perjalanan wisatanya di Yogyakarta. Implementasi penelitian ini disampaikan dengan sebuah aplikasi berbasis *website.* Seorang wisatawan yang berwisata di Yogyakarta diasumsikan memasukan jumlah harinya untuk berlibur, setelah itu wisatawan memasukan tempat menginap (titik awal perjalanan) dan

daftar destinasi yang akan ia kunjungi. *K-means* digunakan untuk membagi jumlah destinasi setiap harinya, kemudian untuk mementukan urutan destinasi dilakukan lewat pendekatan solusi *Traveling Salesman Problem* dengan algoritma *Brute Force.*

Berikut rangkuman kajian yang pernah dilakukan terhadap penelitian sebelumnya:

Tabel 2.1 Rangkuman penelitian sebelumnya

Peneliti	**Judul**	**Metode**	**Studi Kasus**
Lukman, Andi, AR, Rubinah, Nuhayati, 2011	Penyelesaian Traveling Salesman Problem dengan Algoritma Greedy	Algoritma Greedy	Sales
Suprayogi, D.A Mahmudy, W.F. 2014	Penerapan Algoritma Genetika Traveling salesman problem with Time Window: Studi Kasus Rute Antar Jemput Laundry	Algoritma Genetika	Layanan antar jemput *laundry*
Khamyat, Chatput 2015	*Solving the Oil Delivery Trucks Problem with Modify Multi-Traveling Salesman Problem Approach*	*Modify Multi-Traveling Salesman Problem*	*The SME's Oil Logistic Company in Bangkok, Thailand*
Kholidah, Kartika Nur 2017	Aplikasi Pembuat Itinerary Wisata di Provinsi Daerah Istimewa Yogyakarta dengan Pendekatan Solusi *Traveling Salesman Problem* dan *K-Means Clustering*	*Brute Force* dan *K-Means Clustering*	Pembuatan Itinerary Wisata di Provinsi Daerah Istimewa Yogyakarta

Penelitian ini menggunakan pendekatan *Traveling Salesman Problem* di karenakan parameter yang hendak digunakan untuk membuat rencana wisata adalah jarak antar titik wisata, dan penggunaan algoritma *Held-Karp* karena memiliki *runtime* yang lebih cepat

dibandingkan *Brute Force* untuk menyelesaikan permasalahan TSP. Metode *Balanced K-Means* digunakan untuk menyeimbangkan jumlah anggota setiap *cluster* nya.

2.2 *Traveling Salesman Problem*

Traveling Salesman Problem (TSP) dikemukakan oleh matematikawan Irlandia, Sir William Rowam Hamilton dan matematikawan Inggris Thomas Penyngton Kirkman pada abad ke-18 (Megariza, 2011). Definisi TSP sendiri sebuah permasalahan optimasi yang dapat diibaratkan terdapat beberapa kota (simpul) dan jarak antar kota (sisi) sehingga membuat satu kota dengan kota lain terhubung. TSP untuk mencari waktu terbaik yang mungkin untuk mengunjungi seluruh kota tepat satu kali dan kembali pada kota keberangkatan, hal ini dikenal dengan Sirkuit Hamilton. TSP salah satu permasalahan yang terkenal dalam teori graf.

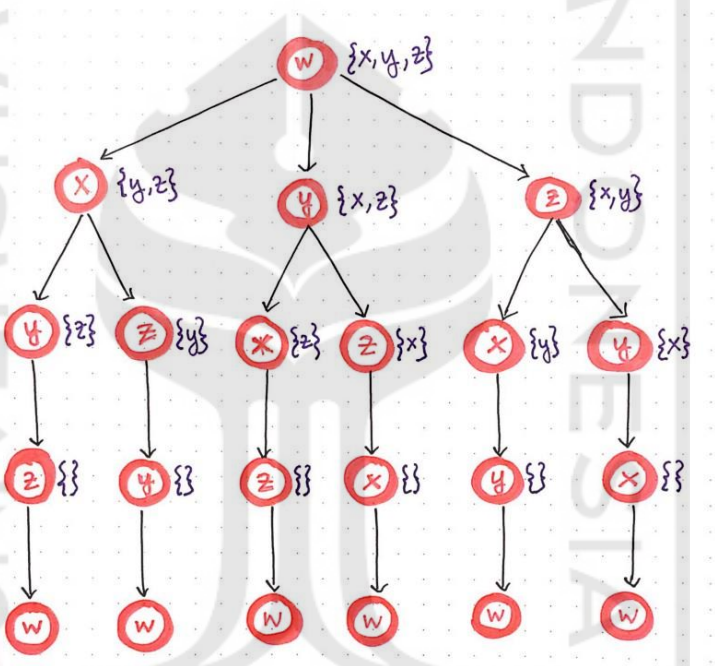
Dalam hal kompleksivitas, TSP memiliki nilai yang cukup tinggi. Total rute yang mungkin apabila diberikan *n* kota adalah (*n*-1)!/2 (Rajesh Matai, 2010). Hal ini didapatkan pada graf lengkap dengan $n>2$, sebagai contoh diberikan 7 kota, maka akan menghasilkan 360 rute atau sirkuit Hamilton yang berbeda dan yang dibutuhkan adalah 1 buah rute yang memiliki *cost* paling minimum. Meski begitu terdapat penelitian telah menyelesaikan permasalahan ini menggunakan algoritma genetika yang merupakan salah satu metode heuristic.

2.3 *Held-Karp Algorithm*

Algoritma *Held-Karp* merupakan salah satu algoritma *dynamic programming* yang dikemukakan oleh Bellman, Held dan Karp (Rahayuda, 2017). Algoritma ini juga disebut dengan *Bellman-Held-Karp* karena ketiga peneliti tersebut menciptakan sebuah algoritma untuk memecahkan *Traveling Salesman Problem* pada tahun 1962 (Rahayuda, 2017). Untuk menyelesaikan permasalahan TSP dapat menggunakan algoritma *brute-force* atau algoritma *held-karp* untuk mencari urutan kunjungan dengan jarak paling minimum (Rani, 2017). Kompleksitas waktu terburuk dari algoritma ini adalah $O(2^n n^2)$, dan meskipun masih bersifat eksponensial, namun algoritma ini lebih baik dibanding *brute-force* yang memiliki kompleksitas O(n!). Pada kasus aplikasi pembuat *itinerary* ini, penulis menggunakan pendekatan *bottom up,* yaitu memecah permasalahan menjadi sub masalah yang lebih kecil kemudian menggabungkan solusi dari setiap sub masalah untuk menyelesaikan masalah

utama. Langkah-langkah yang perlu diperhatikan dalam menyelesaikan permasalahan TSP ini sebagai berikut:

1. Menentukan titik keberangkatan, dan jarak setiap titik yang harus dikunjungi. Jarak setiap titik dimasukan kedalam *adjacency matrix.*
2. Membuat dua buah *matrix* bantuan, untuk menyimpan data setiap rute yang akan dikunjungi.
3. Mencari jarak setiap titik rute kunjungan dengan catatan, apabila titik belum pernah dikunjungi, maka akan menghitung jarak antara titik tersebut, namun apabila titik tersebut sudah dikunjungi maka akan mengambil nilai dari data yang disimpan didalam *matrix* bantuan.



Gambar 2.1 Contoh *tree* dalam permasalahan TSP (Joshi, 2017)

Perhatikan pada gambar di atas pada rute dari *x* ke *y*, dipanggil sebanyak dua kali, hal ini bisa membuat penggunaan memory lebih besar dalam melakukan komputasi perhitungan pada nilai yang sudah dihitung sebelumnya. Maka dari itu, data yang pernah dihitung akan dimasukkan kedalam *matrix* bantuan dan akan dipakai ketika data yang tersebut dipanggil lagi.

4. Setiap titik tersambung akan dihitung untuk mendapatkan jarak kunjungan dan apabila terdapat kemungkinan jarak kunjungan baru yang lebih kecil, rute kunjungan akan dipindahkan dengan rute kunjungan yang baru.

5. Iterasi ini dilakukan hingga semua titik sudah dikunjungi dan ditemukan rute kunjungan dengan jarak paling minimum.

2.4 *Balanced K-Means Clustering*

Clustering merupakan sebuah proses untuk mengelompokan data dalam beberapa kelompok sehingga data pada satu kelompok memiliki tingkat kemiripan yang maksimum dan data antar kelompok yang memiliki kemiripan minimum (Tan, 2006). Clustering adalah salah satu topic riset data mining dan digunakan dibanyak sektor dalam *artificial intelligence*, statistic dan *social sciences*. Metode yang sering digunakan dalam *clustering* adalah *K-means*. Metode ini menggunakan penghitungan *euclidean distance* untuk menentukan anggota *cluster* terhadap jarak *centroid* (titik tengah), sehingga anggota akan masuk kedalam salah satu *cluster* dimana memiliki jarak paling minimum terhadap salah satu *centroid*. Namun, menggunakan metode ini akan menghasilkan jumlah anggota *cluster* yang timpang pada kasus pembuatan aplikasi *itinerary* wisata ini, maka untuk mengurangi resiko ini, penggunaan metode *Balanced K-Means* dinilai menghasilkan hasil akhir lebih optimal dan seimbang. Konsep dasar metode *Balanced K-Means* mirip dengan *K-means*, namun membuat jumlah anggota *cluster* menjadi lebih seimbang antar *cluster*. Sebelum memproses input yang diberikan, ditentukan *pre-allocated size* setiap *cluster* sehingga jumlah minimal dan maksimal anggota pada *cluster* sudah diketahui sebelumnya.

Langkah-langkah dalam operasi *Balanced K-Means* sebagai berikut:

1. Menentukan jumlah *cluster*
2. Menentukan titik *centroid* awal setiap cluster secara acak
3. Melakukan perhitungan menggunakan *Hungarian algorithm* dalam menentukan data setiap *cluster*
4. Menghitung rata-rata dari data masing-masing *cluster* untuk mendapatkan calon *centroid* baru.
5. Membandingkan anggota *cluster* sebelumnya dengan anggota *cluster* baru, apakah ada data yang berpindah *cluster*, apabila hasil perbandingan berbeda, maka menghitung *centroid* baru dan kembali melakukan langkah nomor 3, namun apabila tidak ada data yang berpindah *cluster*, maka proses *clustering* selesai.

2.5 *Hungarian Algorithm*

Algoritma Hungarian pertama kali dikemukakan oleh Harold Khun tahun 1955, yang kemudian diperbaiki oleh James Munkres pada tahun 1957. Oleh karena itu, algoritma ini juga dikenal dengan Kuhn – Munkres (Nugraha, 2018). Algoritma Hungarian adalah salah satu algoritma yang digunakan untuk menyelesaikan masalah penugasan. Pemecahan masalah menggunakan algoritma ini sederhana dan mudah dipahami. Tujuan penyelesaian menggunakan algoritma ini adalah menugaskan pekerjaan kepada setiap individu untuk mendapatkan hasil yang optimal dengan biaya paling minimal. Pada penelitian ini, algoritma Hungarian digunakan pada proses *balanced k-means* untuk membagi setiap data kedalam cluster yang diharuskan memiliki jumlah anggota seimbang.

Langkah - langkah dalam menyelesaikan masalah penugasan menggunakan *hungarian algorithm* sebagai berikut:

1. Memodelkan data pekerja dan pekerjaan ke dalam matrix, dimana data pekerja sebagai baris dan data pekerjaan sebagai kolom. Menentukan angka *n* yang didapatkan dari nilai terbesar perbandingan antara jumlah baris dan jumlah kolom data. Jika ukuran kolom lebih kecil dibanding ukuran baris, maka ukuran kolom akan ditambahkan dengan *k*, dimana *k* didapatkan dari ukuran kolom awal, tetapi jika ukuran baris lebih kecil dibandingkan ukuran kolom, maka akan ditambahkan *l* baris. Angka *l* didapatkan dari selisih ukuran baris dan kolom. Apabila ukuran matrix sudah seimbang ($m \times m$), masukkan data pekerja dan pekerjaan kedalam matrix tersebut. Jika terdapat baris matrix yang belum memiliki nilai, masukkan nilai nol (0) ke dalam baris matrix tersebut dan apabila terdapat kolom matrix yang belum memiliki nilai, masukkan kembali nilai kolom yang sudah terisi kedalam kolom yang kosong.

Tabel 2.2 Tabel awal nilai penugasan

	a	**b**	**c**	**d**
1	15	20	18	20
2	14	13	21	23
3	25	20	23	19
4	17	18	12	20

2. Mencari nilai paling kecil setiap baris *matrix*, kemudian kurangi data pada baris tersebut dengan nilai paling kecil setiap baris *matrix*.

Tabel 2.3 Tabel mencari data paling minimum pada baris

	a	b	c	d	Minimum
1	15 - 15	20 - 15	18 - 15	20 - 15	15
2	16 - 13	13 - 13	21 - 13	23 - 13	13
3	25 - 19	20 - 19	23 - 19	19 - 19	19
4	17 - 12	18 - 12	12 - 12	20 - 12	12

Setelah dikurangi nilai paling kecil setiap baris, tabel akan berubah seperti di bawah ini.

Tabel 2.4 Tabel hasil pengurangan nilai pada baris

	a	b	c	d
1	0	5	3	5
2	3	0	8	10
3	6	1	4	0
4	5	6	0	8

3. Memastikan semua baris dan kolom memiliki nilai 0. Apabila masih terdapat kolom yang belum memiliki nilai 0, maka dilakukan langkah ke-4, namun apabila semua baris dan kolom memiliki nilai 0, maka langkah berlanjut ke-5.

4. Mencari nilai paling kecil setiap kolom *matrix*, kemudian kurangi data pada baris tersebut dengan nilai paling kecil setiap baris *matrix*.

5. Memastikan bahwa dalam setiap baris dan kolom tepat memiliki satu angka 0

Tabel 2.5 Tabel dengan nilai 0 pada setiap baris dan kolom

	a	b	c	d
1	0	5	3	5
2	3	0	8	10
3	6	1	4	0
4	5	6	0	8

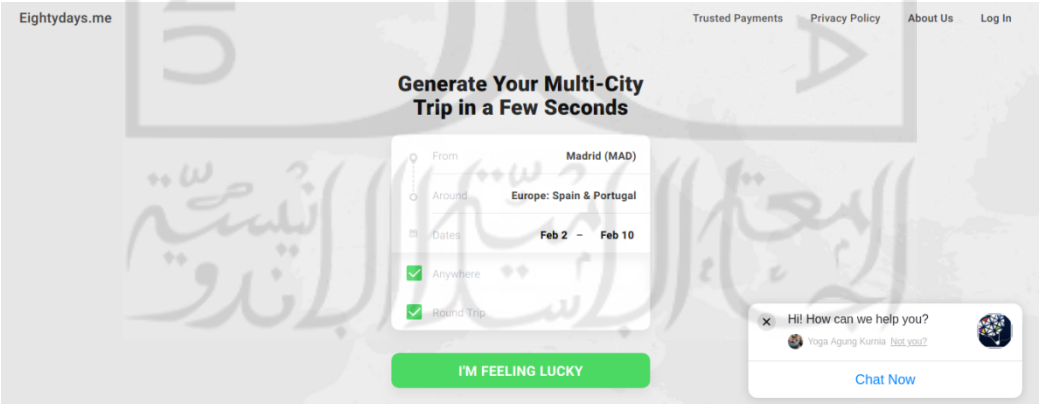
Dapat diperhatikan, kotak dengan latar abu-abu, merupakan angka 0 pada pada baris dan kolom yang berbeda. Apabila kondisi pada langkah ke-5 terpenuhi, maka solusi sudah ditemukan dengan hasil:

1. 1 mengerjakan pekerjaan a dengan bobot 15
2. 2 mengerjakan pekerjaan b dengan bobot 13
3. 3 mengerjakan pekerjaan d dengan bobot 19
4. 4 mengerjakan pekerjaan c dengan bobot 12

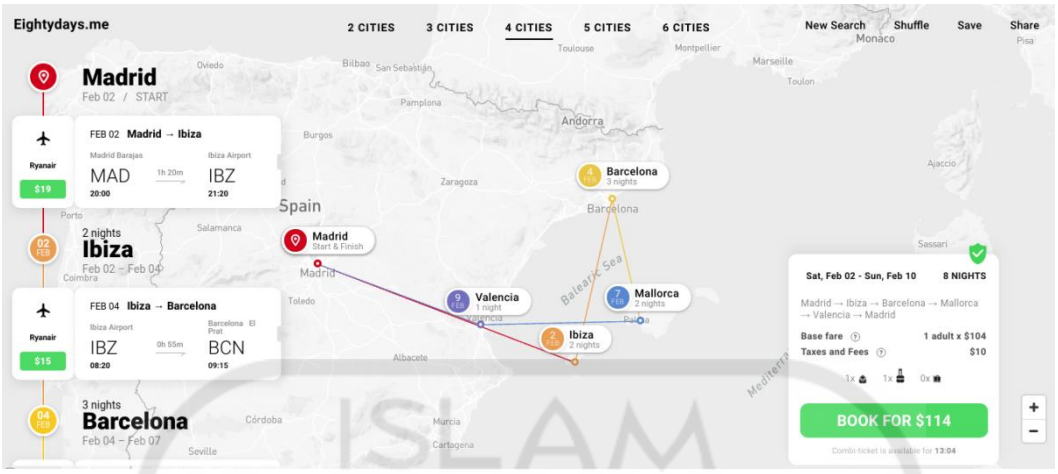
2.6 Aplikasi *Itinerary* Wisata

Perkembangan teknologi memiliki andil dalam perkembangan sektor wisata dalam beberapa tahun terakhir. Banyak aplikasi yang telah dikembangkan untuk membantu wisatawan dalam melakukan perjalanan wisata. Tripadvisor, dan Lonely Planet merupakan contoh aplikasi yang berisi *review* perjalanan atau destinasi, menyediakan layanan transportasi dan informasi penginapan disekitar destinasi tersebut. Contoh aplikasi lain adalah *Couchsurfing* yang ditujukan untuk *traveler* atau *host*. *Traveler* merupakan wisawatan yang sedang melakukan perjalanan wisata, sedangkan *host* merupakan orang yang menjamu dan menyediakan tempat untuk para *traveler*.

Hingga saat ini, penulis menemukan referensi dalam pembuatan aplikasi *itinerary* wisata yaitu Eightydays. Aplikasi ini membantu wisatawan untuk merancang *itinerary* atau rencana perjalananan beberapa kota di Benua Eropa (Rizzo, 2017). Untuk menggunakan aplikasi ini, wisatawan memasukan data berupa tanggal keberangkatan, durasi perjalanan, banyak kota yang ingin dikunjungi, kota keberangkatan, dan kota tujuan akhir perjalanan kemudian aplikasi akan bekerja untuk menentukan *itinerary* bagi wisatawan tersebut. Sebagai contoh Gambar 2.2, wisatawan memasukan lokasi di kota “Madrid”, dimulai pada tanggal 2 Februari hingga 10 Februari, dan tujuan kota berada di sekitar “Spanyol dan Portugal”. Hasil perancangan *itinerary* ini dapat dilihat pada Gambar 2.3, terdapat informasi lain berupa perkiraan biaya transportasi pada kanan bawah halaman aplikasi.



Gambar 2.2 Contoh tampilan Eightydays



Gambar 2.3 Contoh hasil pembuatan *itinerary* di aplikasi Eightydays

Aplikasi ini akan membantu pembuatan *itinerary* wisata, apabila wisatawan bertujuan untuk mengunjungi beberapa kota, namun dalam satu kota, terdapat beberapa tempat wisata yang dapat dikunjungi. Berawal dari referensi ini, muncul ide untuk menyediakan aplikasi untuk perancangan rencana perjalanan berdasarkan tempat atau daerah wisata yang ingin dikunjungi wisatawan. Pemilihan platform *Android* dipertimbangkan, karena untuk mendukung mobilitas wisatawan, sehingga dengan menggunakan *smartphone* dapat melakukan pembuatan rencana perjalanan.

Aplikasi kedua yang penulis jadikan referensi adalah aplikasi Generator Wisata. Aplikasi ini berasal dari mahasiswa Bandung yang berhasil meraih beberapa penghargaan dalam kompetisi di Indonesia. Aplikasi ini membantu pengguna untuk membuat rencana perjalanan pada suatu daerah di Indonesia. Pengguna diharuskan untuk memasukkan data kota yang ingin dikunjungi, tanggal mulai dan selesai berlibur, serta memasukan kategori tempat yang ingin dikunjungi. Kategori yang tersedia dalam aplikasi ini diantaranya museum, sejarah, gunung, pantai, hutan, budaya dan *hidden paradise*. Aplikasi ini akan menampilkan beberapa pilihan tempat wisata dan perkiraan waktu perjalanan yang dibutuhkan dari awal perjalanan wisata hingga akhir wisata. Aplikasi yang dapat di unduh di Google Play Store ini, memberikan informasi jalur dalam bentuk peta Google Maps dan destinasi wisata favorit para *traveler* di sekitar kota yang kita kunjungi.

Gambar 2.4 Aplikasi Generator Wisata