Automated Travel Itinerary Generation and Booking

Adam Currie and Devon Govett
Dr. John Clements, Adviser
Computer Science Department
College of Engineering

June 2015

Abstract

Currently, planning travel online is a com- plicated process involving many websites, choices, and annoyances. For our senior project, we built a website where users can enter a list of all destinations they'd like to visit on their trip, and an optimal itinerary is automatically produced for them using the best available flights, rental cars, and hotels.

Contents

1 Introduction

Currently, planning travel online is a complicated process involving many websites, choices, and annoyances. Users need to find flights, book rental cars, book hotel rooms, and more, usually all individually, and they are given a multitude of choices for each step. There is room for a service that brings all of these things together to determine the best itinerary automatically, without all of these complications.

For our senior project, we built a website where users can enter a list of all destinations they'd like to visit on their trip, and an optimal itinerary is automatically produced for them using the best available flights, rental cars, and hotels. The user then has an opportunity to adjust parameters such as travel budget, and book their travel on site via third party providers, using a streamlined booking flow.

2 Problems with existing travel sites

There are many options for users looking to search and book travel online, including online travel agencies (OTAs) such as Expedia and Orbitz, metasearch sites like Kayak and Hipmunk that search across multiple providers, and directly on airline/hotel/rental car websites. All of these require the user to do searches for each leg of their itinerary individually, and the user is presented with an often unfiltered list of results that they must sort through manually to choose an option for each leg. Then, once users have decided on each leg, they must book them all manually by entering their personal information many times across different websites. All of this takes hours of human effort that could be better spent on other things.

3 Features

Our website searches for the best combination of flight, hotel, and car rental for trips of all sizes, all at once. One way, roundtrip, and multi-destination trips are supported. An itinerary is created based on the user's choice of destinations and dates. We search across several different providers for flights, hotels, and rental cars and then choose the best option for each leg algorithmically to build the best overall itinerary. After the user sees an initial itinerary, they can adjust their "budget" to prioritize either price, quality of trip, or something in the middle, using a slider.

Once the user has tweaked the itinerary to their liking, they can book the itinerary entirely on-site, without being redirected to third party websites like some other travel sites. However, we are not an online travel agency ourselves, as we do not work directly with suppliers (e.g. airlines, hotels, and rental car companies). Instead, we integrate with other existing providers to perform the booking on the user's behalf. This is both more convenient for the user and more feasible for us since the providers do the credit card processing, provide customer service, etc. but the user does not need to leave our website to book each leg individually. They enter their personal information once on our website, and we book each leg for them on their behalf. This is especially convenient for long and complex trips with many legs.

3.1 Flight Search

Flights are searched for when destinations are greater than 5 hours apart by driving. When flights are needed, we use Google/ITA Software's QPX Express API to perform the flight search. Flights are chosen based on factors such as price, layover time, number of stops, and overall duration. The budget parameter affects how flights are chosen by adjusting the importance of the previously mentioned factors.

3.2 Hotel Search

Hotels are placed on the itinerary for each destination when the trip is more than just a one way flight. We currently use the Expedia Affiliate Networks API to search for hotels. Hotels are chosen based on price and ratings, and the budget adjusts the importance of these factors.

3.3 Car Rental Search

Rental cars are placed on the itinerary between each non one-way flight. The car pick up location is as close to the airport as possible to help the user once the user arrives at their destination. We use the Cartrawler API to search for rental cars currently. Cars are chosen based on price and car size class, and the budget adjusts the importance of these factors.

4 Design and Implementation

This project involved building three systems: a database of travel information acquired from third parties, a backend API for querying the database and building itineraries, and a frontend user interface.

4.1 Database

We used the MongoDB database for our project. MongoDB is a NoSQL, document-oriented database, which worked well for us since we have numerous nested data structures that are harder to represent using traditional relational databases. Mongo also supports easy-to-use indexed geographic queries, which we use heavily to search by location. Our database comprises of a collection of places, and also serves as a cache for flight, hotel, and car rental data received from third party providers.

4.1.1 Places

We store several types of location data in the database, including airports, cities, hotels, and rental car locations. By storing this information locally we can optimize our queries to third party providers, and find places in proximity to other locations easily.

4.1.2 Caching

Many travel providers are very slow to respond to requests sent to their APIs. This can be seen in general on most travel websites where doing a search can normally take upwards of 15 to 30 seconds, especially for flights. This is caused by a combination of factors such as the old mainframe backends that still power the underlying travel providers' databases, as well as the algorithmic complexity involved in flight route planning (see `http://www.demarcken.org/carl/papers/`

ITA-software-travel-complexity/ITA-software-travel-complexity.pdf).

In order to make queries faster, we utilize caching to improve the performance of subsequent requests for overlapping data. This is important in general, but especially so for subsequent requests when the user is adjusting their budget, where waiting around every time a change is made would be annoying and far too slow. We are able to make this fast by caching so we don't need to go back to third party providers every time. We use various techniques to widen our queries to providers to improve cache hit rate, and do local filtering of the returned data instead of letting the provider do that. This way, we get more data back from the provider than we need immediately, but we can cache that data for later.

4.2 Backend

We used Node.js to build the backend API for our website. Node.js is a JavaScript-based server platform that has been popular recently, and because of the JavaScript language's support for higher order functions and closures, it is well suited for building asynchronous and highly I/O-parallel applications. This made it a good fit for this project since we need to query many third party providers simultaneously to build itineraries efficiently. It also integrates really well with our database, MongoDB, which uses a JSON-like object structure for its documents, as well as our API which also uses a JSON interface. JSON (JavaScript Object Notation) is a lightweight data interchange format based on the JavaScript object syntax. Many languages have JSON support, but nothing can beat its namesake: JavaScript.

4.3 Frontend

We built our website using React; an open source JavaScript library from Facebook for building user interfaces. The website consists of a search page, an itinerary page, and a booking page.

The search page consists of a form where the user inputs destinations they'd like to travel to, the dates they'd like to visit each place, and the number of travelers going on the trip. We use the Google Places Autocomplete API to select each destination city, and the jQuery UI Datepicker library for date selection. When the user clicks the search button, the website queries the backend API and displays the generated itinerary to the user.

The itinerary page shows the user the generated itinerary for their trip, and allows them to adjust it. We use Mapbox to show maps for each leg, which allows the user to visualize the locations in their itinerary. Various information is displayed for each leg, including each flight segment for flight legs, an image gallery and description for hotels, and a picture and vehicle information for rental cars. We also display a pricing overview alongside the legs in the itinerary, which shows a breakdown of the prices for flights, hotels, and cars, and a grand total.

There is also a budget slider available to allow the user to choose their desired trade-off between price and quality of trip. Changing the budget slider re-queries the backend API and re-displays the adjusted itinerary. This is generally very fast since all results are cached during the initial query.

Once the user is ready to book their trip,

they can click the Book Itinerary button to go to the booking page. Before they book, we re-check the pricing and availability with our third party providers to ensure that all legs are still available at the prices the user saw on the itinerary screen. This is necessary since the user may have seen cached data, and prices may have changed in the meantime. On the booking page, there is a form that accepts contact information, payment information, and information about each traveler. This information is sent to the backend API which performs the booking for each leg using our third party providers on behalf of the user.

Once booked, the user can view their itinerary using the same interface from the search results page, but with their booking information along with each leg, including confirmation numbers, etc. We currently do not support canceling or making modifications to itineraries after they are booked, but this is something that we would like to support in the future.

5 Problems Encountered and Lessons Learned

Building this project was a significant effort, and we encountered several problems along the way. These included working with various slow and unresponsive providers, making search performance acceptable, dealing with timezones, rate limiting, and more.

5.1 Finding and working with travel providers

Finding publicly available APIs to access travel availability and pricing information was a challenge. There are many travel websites out there, but many of them either don't have APIs available, or only allow access to a small number of partners. Contacting these providers was often futile and frequently resulted in no response. We were lucky to find several providers that we were able to integrate with, but this was a challenge.

On the booking end, this was an even larger challenge as there is an even smaller set of providers that allow booking via their APIs. Many APIs include only a search component, allowing partners only deep links to the API provider's website to do the actual booking. This works fine for many metasearch sites, such as Kayak and Hipmunk, which do not generally do booking on their own websites. However, since we combine many providers together to book an itinerary all at once, it would be very inconvenient for our users to have to click off to each provider individually to perform their booking, entering their personal information on each site separately. We did end up finding enough partners to make this work, but in order to actually bring our site into production, we would need to make official business deals with them.

5.2 Performance

Unfortunately, many travel search APIs and websites are very slow. Performing flight searches is generally quite slow across most websites due to the algorithmic complexity of flight route planning. Compounding the problem, we need to perform many queries in order to build a single itinerary, which makes things even slower. To combat this, we perform caching for all of the content we

receive from third party providers, as described above. Designing this caching to maximize cache hit rate was a significant challenge.

We generally try to request more data than we need at once from third party providers so that it can be cached for future queries. We also do as much work as we can ourselves rather than relying on the filtering provided by these APIs, and recompute pricing information using a subset of cached data from previous wider queries where possible. This works fairly well. However, queries are still slow the first time a user performs them, and since availability and pricing data changes relatively quickly, it may happen more often than desired. Generally, the more users are using the service, the better the caching will perform.

6 Future Work

Building this website for our senior project was a rewarding experience, and we managed to get many more features done than originally planned. However, there is a great deal more we would like to do, and need to do, before it would be ready to launch publicly. Some of these were mentioned in this paper previously, but a few more are listed below.

6.1 Itinerary Watch Feature

We would like to add a feature to allow the user to watch an itinerary for pricing changes. The user would receive an email once a day detailing any price changes. Many travel websites offer this feature for individual flights, hotels, or cars, but no one offers a feature to watch an entire itinerary for overall changes all at once.

6.2 Suggest Alternate Travel Dates

It would be useful if we could analyze the travel pricing and availability of surrounding dates to what the user entered and suggest alternate dates where travel is more affordable or where better options are available. Many sites support features like this, but for only one leg at a time. An analysis over an entire trip all at once would be very useful.

6.3 More Search and Booking Providers

We currently integrate with one partner for each type of travel leg, but ideally we would integrate with as many providers as possible to truly build the best possible itineraries. Users generally search across several different websites before booking their travel, and doing this all at once for them would be very valuable. There are many so-called metasearch websites out there that do this, such as Kayak and Hipmunk, but they still only search one leg at a time.

6.4 Manual Itinerary Editing

While automatically generating the best itinerary by default is very convenient and saves lots of time, we recognize that users may not trust the system at first since they are used to filtering through lists of results themselves. We would like to eventually add a feature to allow users to view the full list of options for each leg, sorted using the same algorithms we use to choose the best option.

This would allow the user to manually override a leg with their own selection from our results if they wanted to. It would also allow the user to see for themselves why we chose the option we did, and would perhaps lead to more trust in the automated choices. Additionally, if users did replace an automatically selected leg with a different option, we could learn from this and improve our algorithms.

In addition to overriding legs, we also need to allow removing of legs from a trip. For example, we always include rental cars following flights currently, but users may decide they don't want a car and will take public transit instead. Also, users may have found a different option on another website that they want to get instead of one of our results, but they still want to book some of the itinerary we generated. Allowing users to remove legs from an itinerary before booking would alleviate these issues.

6.5 Mobile Apps and In-Transit Experience

For this project, we built a website, which is great because it works across all devices and platforms. However, mobile optimized applications are very popular these days and can offer additional features and better offline experiences than web applications. We have the beginnings of an iOS app built (for the iOS development course), but it needs to be improved to have feature parity with the website.

Additionally, mobile apps can offer a great in-transit experience, for when the user has already booked their itinerary, and they are on their trip. The user may need to make changes to their itinerary while they are traveling, cancel portions of their itinerary, or add additional legs. Additionally, we can offer features to check flight statuses, get airport maps, get directions to locations in their itinerary, suggest things to do in their destinations, etc.

7 Conclusion

This project improves the user experience of planning and booking travel. Letting computers do the hard work of searching and filtering the available flights, hotels, and rental cars to build the best possible itinerary saves users a large amount of time. Integrating booking into the website to allow users to book their entire trip all at once improves the user experience as well. We are excited to continue working on this project in the future, and possibly launch it to the public.[**?**]