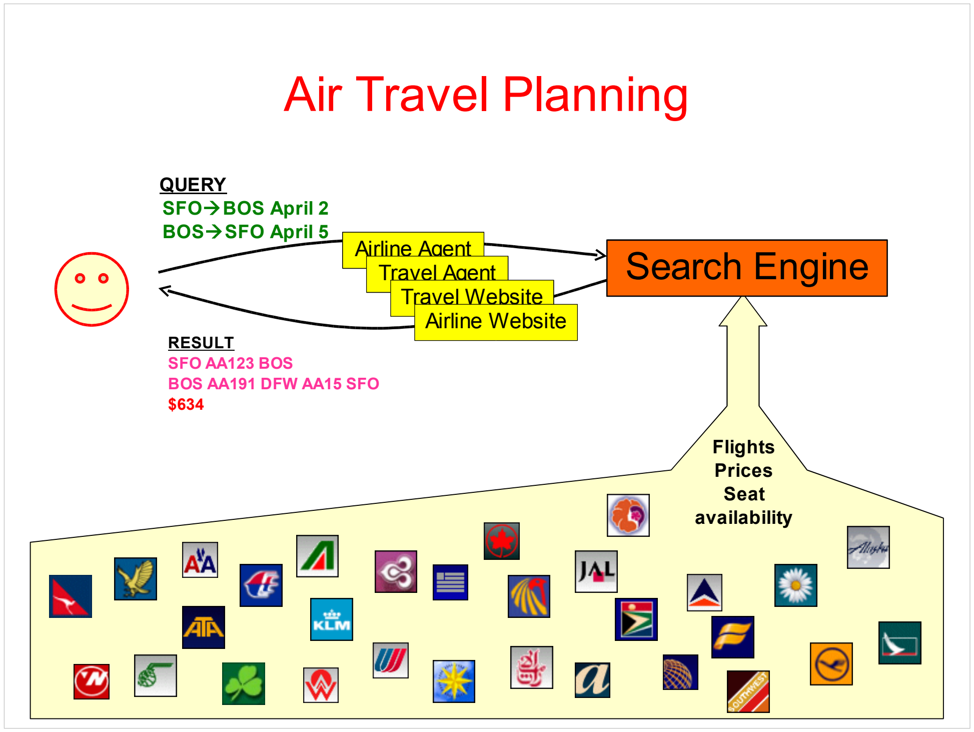
Computational Complexity of Air Travel Planning

Carl de Marcken
ITA Software
carl@itasoftware.com

This document is an annotated set of slides on the computational complexity of air travel planning. The goal is to give somebody with an undergraduate level computer science background enough information to understand why air travel planning is an interesting and especially difficult problem. It provides a basic introduction to the air travel planning problem and then presents a variety of original computational complexity results as well as some related demos. The complexity slides assume a basic familiarity with formal languages, computational complexity and computability, but the introduction to the problem should be accessible without this.

ITA Software produces search and optimization software for the travel industry. Our search engines power popular web sites such as Orbitz and Cheap Tickets; airline web sites such as America West, Continental Airlines and Alaska Airlines; computer reservation systems (CRS/GDSes) used by travel agencies, such as Galileo; and various travel agencies. ITA was founded by MIT computer scientists and is located adjacent to MIT in Cambridge, MA.



Air Travel Planning

QUERY
SFO→BOS April 2
BOS→SFO April 5

Airline Agent
Travel Agent
Travel Website
Airline Website

Search Engine

RESULT
SFO AA123 BOS
BOS AA191 DFW AA15 SFO
$634

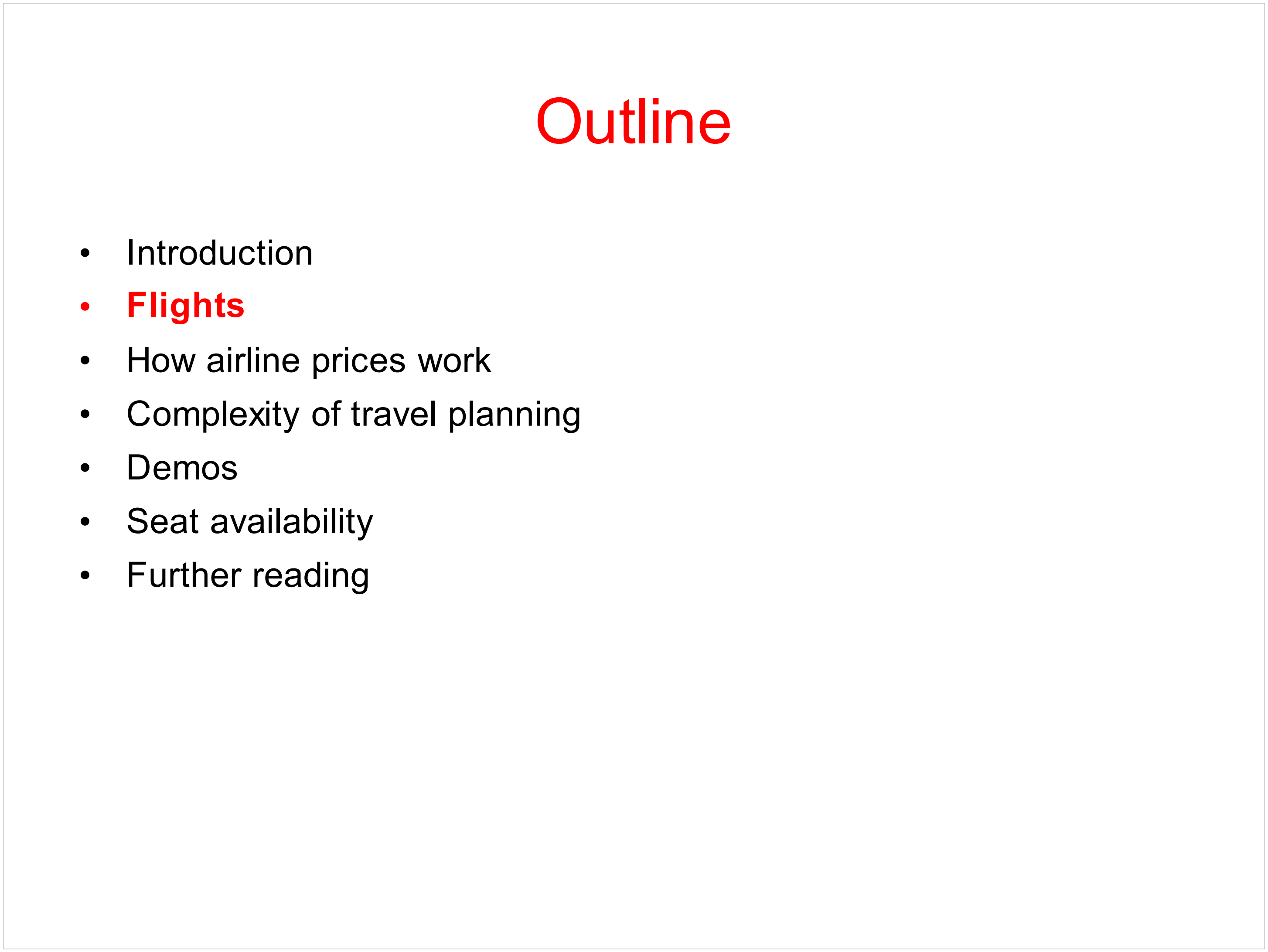
Flights
Prices
Seat
availability

Suppose a traveler is planning a round trip from San Francisco to Boston and back.

Most likely they’ll contact an airline reservation agent or a travel agent, or perhaps visit an airline's website or a general travel website like Orbitz, providing a query made up of airport and travel time requirements. For the most part these agents are middlemen, and will pass the query off to one of a handful of companies, including ITA Software, that provide search engines for the airlines and the traveling public. Hopefully the search engine will return one or more answers to the query. Each answer consists of a specific set of flights for each part of the trip, and a price. The rest of this talk is about the difficulties search engines face answering such questions.

The search engines run on databases of flights, prices, and seat availability, provided electronically over private networks by the 800 or so airlines of the world. The data is not directly available to the general public and access often must be negotiated with individual airlines. Flight data is updated daily or occasionally more frequently in the case of unexpected cancellations. Prices are updated about ten times a day, and seat availability continuously. A large portion of the flight, price and seat availability data, called **published** data, is used by all the major search engines, but a significant amount of **private** data is restricted.

Copyright 2003 ITA Software

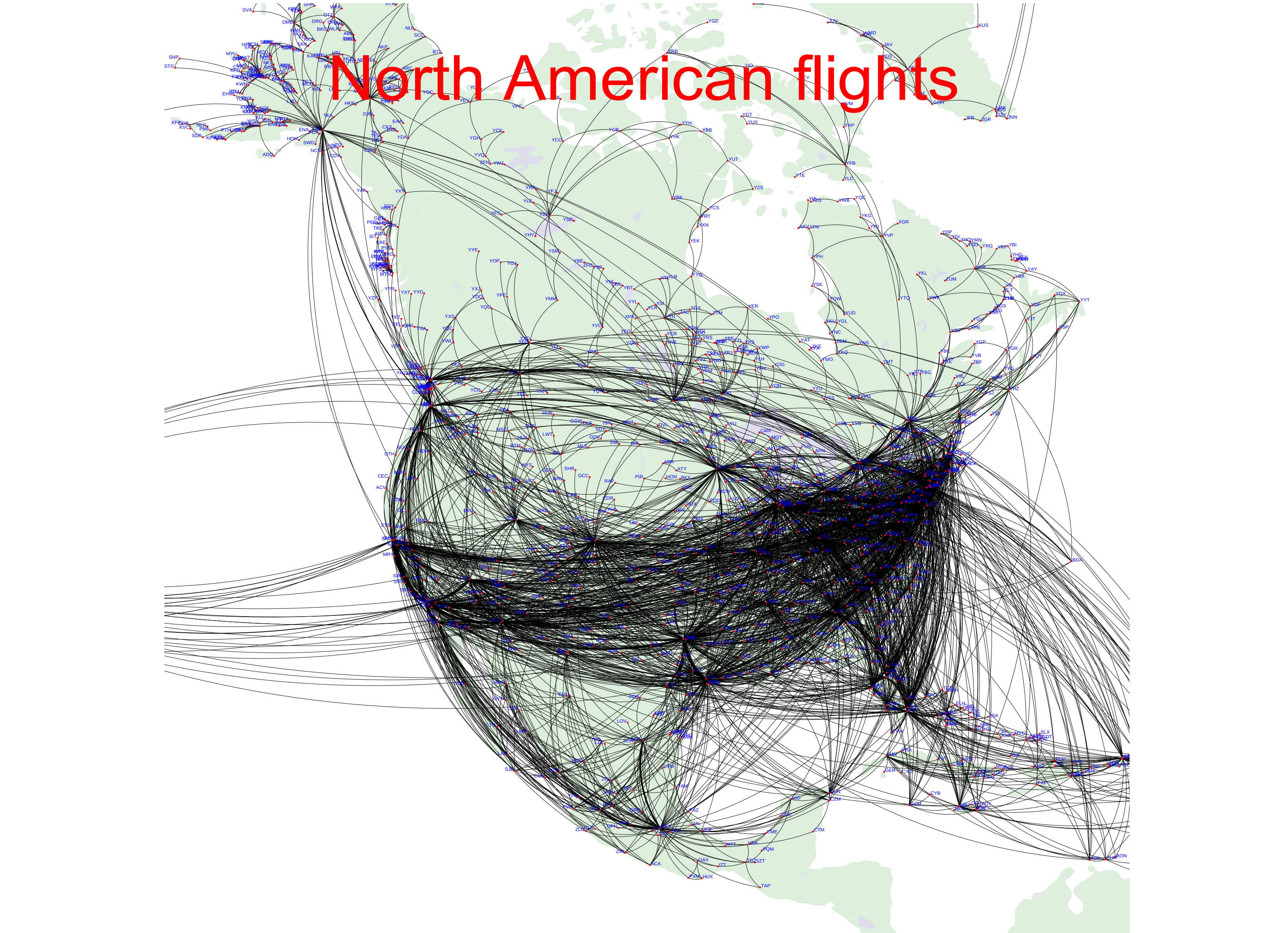


Outline

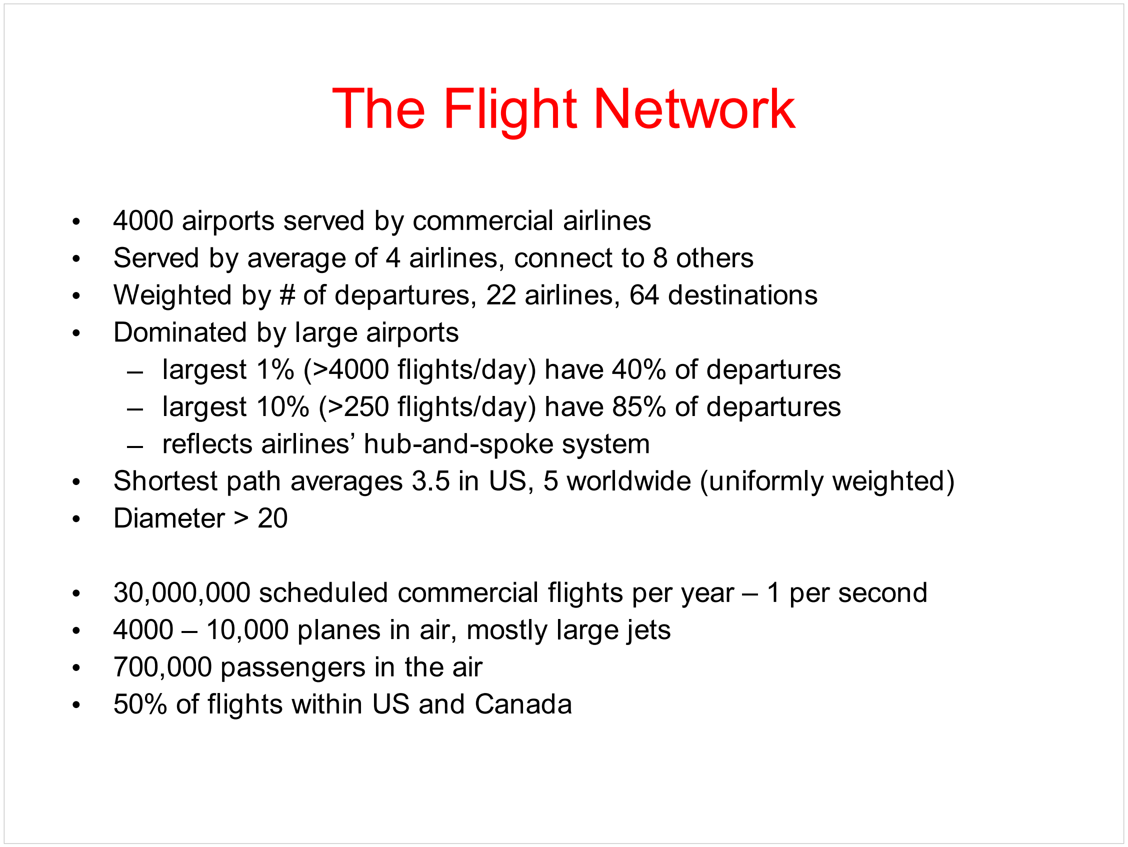
- Introduction
- **Flights**
- How airline prices work
- Complexity of travel planning
- Demos
- Seat availability
- Further reading

The talk starts by listing some fundamental properties of the flight network, but flights are simple relative to prices, and after the flight discussion will be a long but necessary introduction to airline prices. Then some basic computational complexity results about the difficulty of air travel planning are presented, with demos that illustrate the complexity results. The talk concludes with an introduction to seat availability processing, since it is an important part of understanding how airline prices work, though this information isn't used in the rest of the talk.

North American flights



This is a map of all scheduled commercial flights in North America, with an arc drawn between two airports if there is at least one flight between them over the next year. This and most other data presented in this talk dates from 2001.



The Flight Network

- 4000 airports served by commercial airlines
- Served by average of 4 airlines, connect to 8 others
- Weighted by # of departures, 22 airlines, 64 destinations
- Dominated by large airports
  - largest 1% (>4000 flights/day) have 40% of departures
  - largest 10% (>250 flights/day) have 85% of departures
  - reflects airlines' hub-and-spoke system
- Shortest path averages 3.5 in US, 5 worldwide (uniformly weighted)
- Diameter > 20

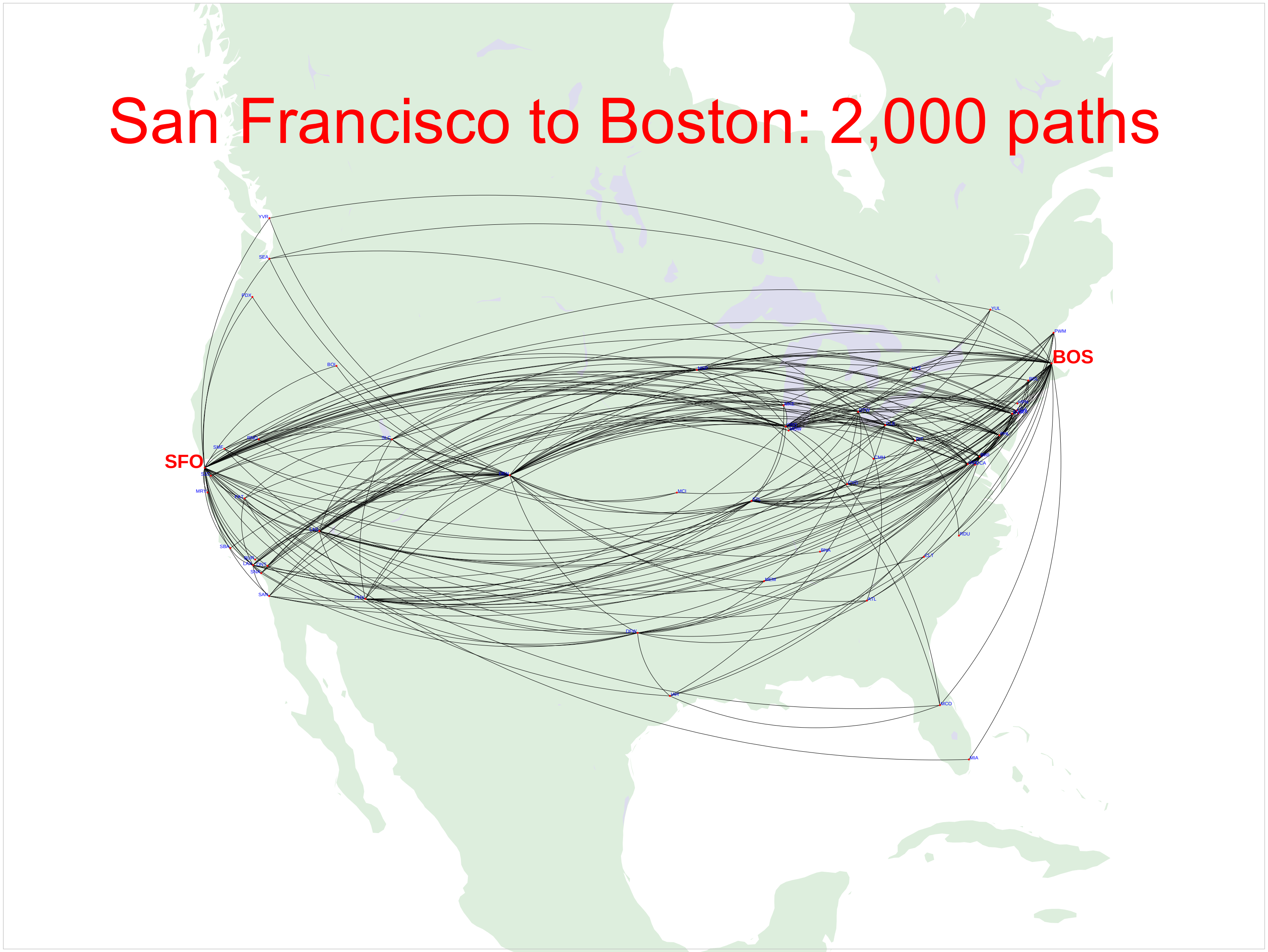
- 30,000,000 scheduled commercial flights per year – 1 per second
- 4000 – 10,000 planes in air, mostly large jets
- 700,000 passengers in the air
- 50% of flights within US and Canada

There are more than 4000 airports served by commercial airlines worldwide. Averaged uniformly, each airport has an outgoing degree of 8 (it has flights to 8 other airports), and is served by 4 airlines. However large airports dominate the system: re-weighted by their number of departures, airports average degree 64. This reflects the hub-and-spoke system used by many airlines, wherein for a given airline one to four airports account for half of their departures. Despite the high connectivity amongst the major airports, the shortest path between two airports chosen uniformly averages 3.5 flights in the United States and 5 worldwide. Amazingly, the graph diameter is often as high as 20: there are airports that can take 20 flights minimum to get between, over 4 days (typically this will be a small airport in Alaska or Canada to another small airport in Africa or Indonesia).

Commercial airliners take off about once per second worldwide, and most are large jets more than half full, resulting in close to a million people in the air at times. The United States and Canada heavily dominate air travel in terms of numbers of flights, though deregulation in Europe has led to a great increase in air travel there over recent years.

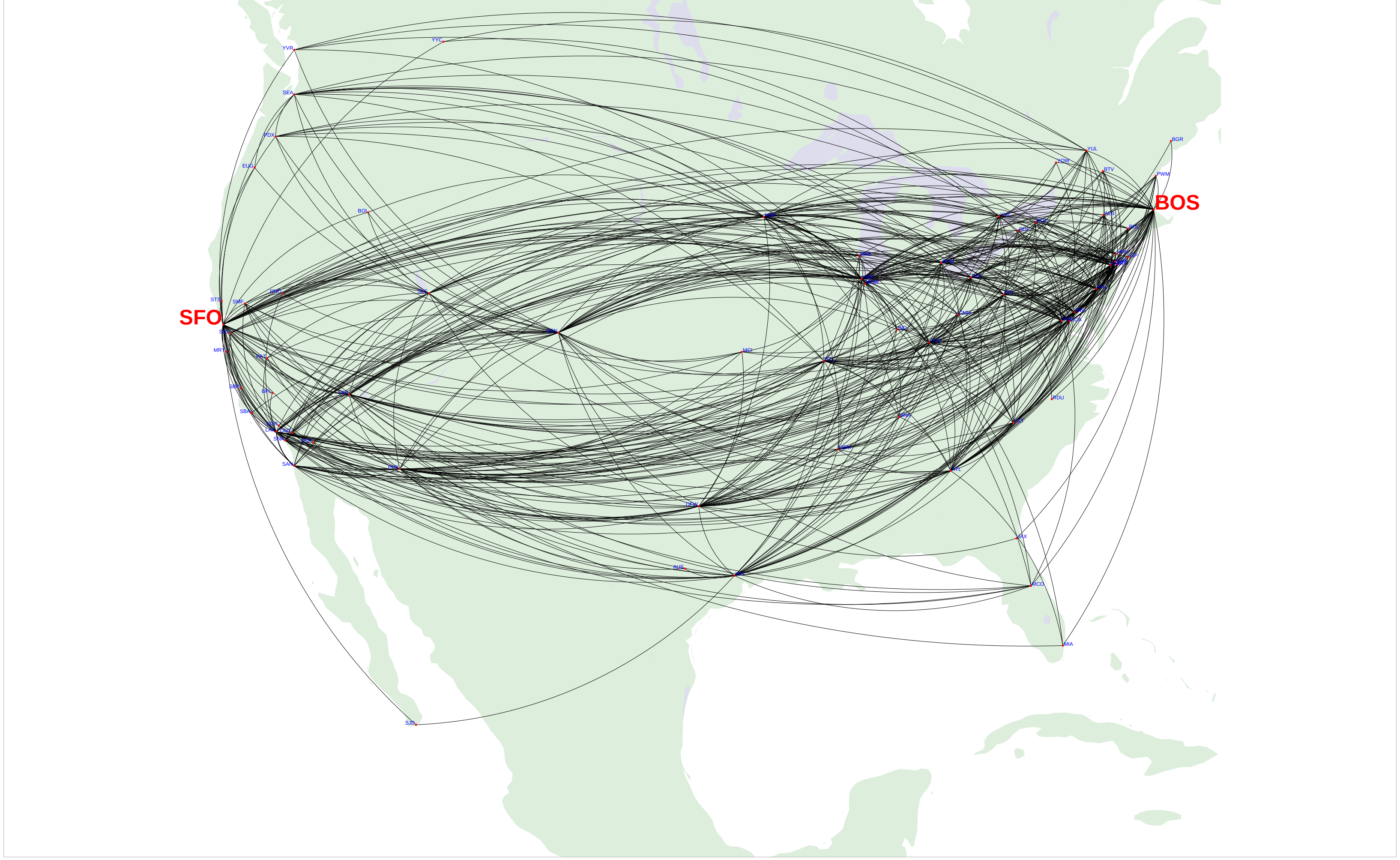
Copyright 2003 ITA Software

San Francisco to Boston: 2,000 paths



Switching to path planning, this map shows the routes involved in the 2,000 quickest paths (flight combinations) from San Francisco to Boston on a certain day. Specific flights are not shown: some arcs represent multiple flights between the same airports at different times of day. Notice that all of these paths look reasonable: they don’t leave the United States and southern Canada, all are length 3 or less, and all arrive the same day. An impecunious traveler might be willing to consider any of them. But 2,000 paths doesn’t begin to exhaust the possibilities.

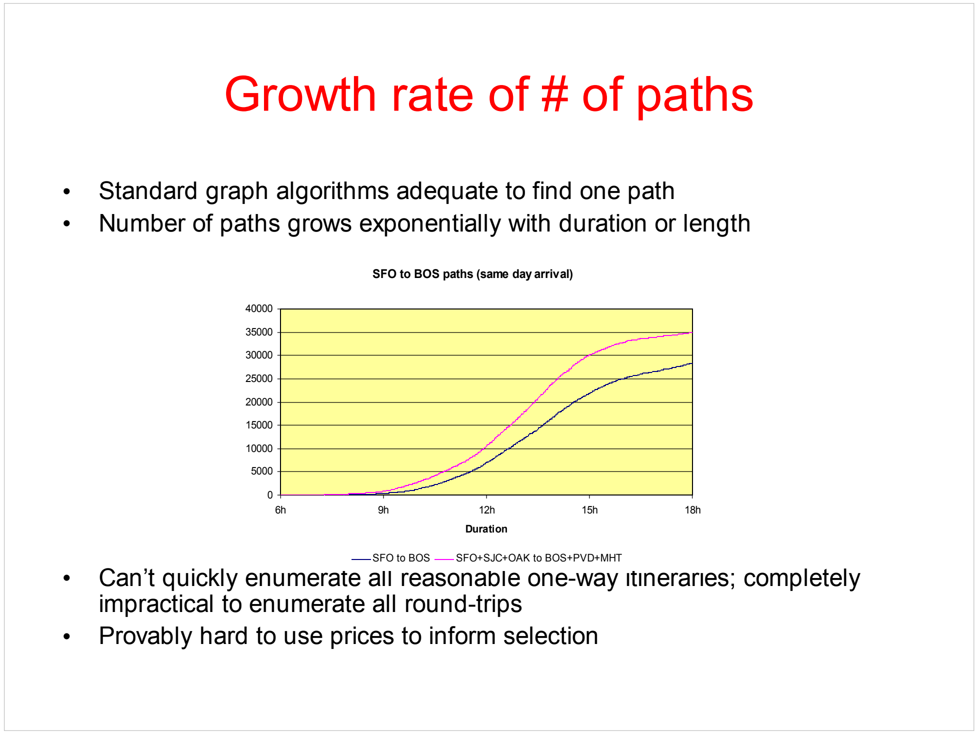
San Francisco to Boston: 10,000 paths



This map plots the arcs in the 10,000 quickest paths, all still arriving on the same day. A few more airports come into consideration, but again no path is out of the realm of possibility.

Copyright 2003 ITA Software

At 30,000,000 flights per year, standard algorithms like Dijkstra's are perfectly capable of finding the shortest path. However, as with any well-connected graph, the number of possible paths grows exponentially with the duration or length one considers. Just for San Francisco to Boston, arriving the same day, there are close to 30,000 flight combinations, more flying from east to west (because of the longer day) or if one considers neighboring airports. Most of these paths are of length 2 or 3 (the ten or so 6-hour non-stops don't visually register on the chart to the right). For a traveler willing to arrive the next day the number of possibilities more than squares, to more than 1 billion one-way paths. And that's for two airports that are relatively close. Considering international airport pairs where the shortest route may be 5 or 6 flights there may be more than 10^{15} options within a small factor of the optimal.



Growth rate of # of paths

- Standard graph algorithms adequate to find one path
- Number of paths grows exponentially with duration or length

SFO to BOS paths (same day arrival)
40000
35000
30000
25000
20000
15000
10000
5000
0
6h
9h
12h
15h
18h
Duration
SFO to BOS
SFO+SJC+OAK to BOS+PVD+MHT

- Can't quickly enumerate all reasonable one-way itineraries; completely impractical to enumerate all round-trips
- Provably hard to use prices to inform selection

One important consequence of these numbers is that there is no way to enumerate all the plausible one-way flight combinations for many queries, and the (approximately squared) number of round-trip flight combinations makes it impossible to explicitly consider, or present, all options that a traveler might be interested in for almost all queries.

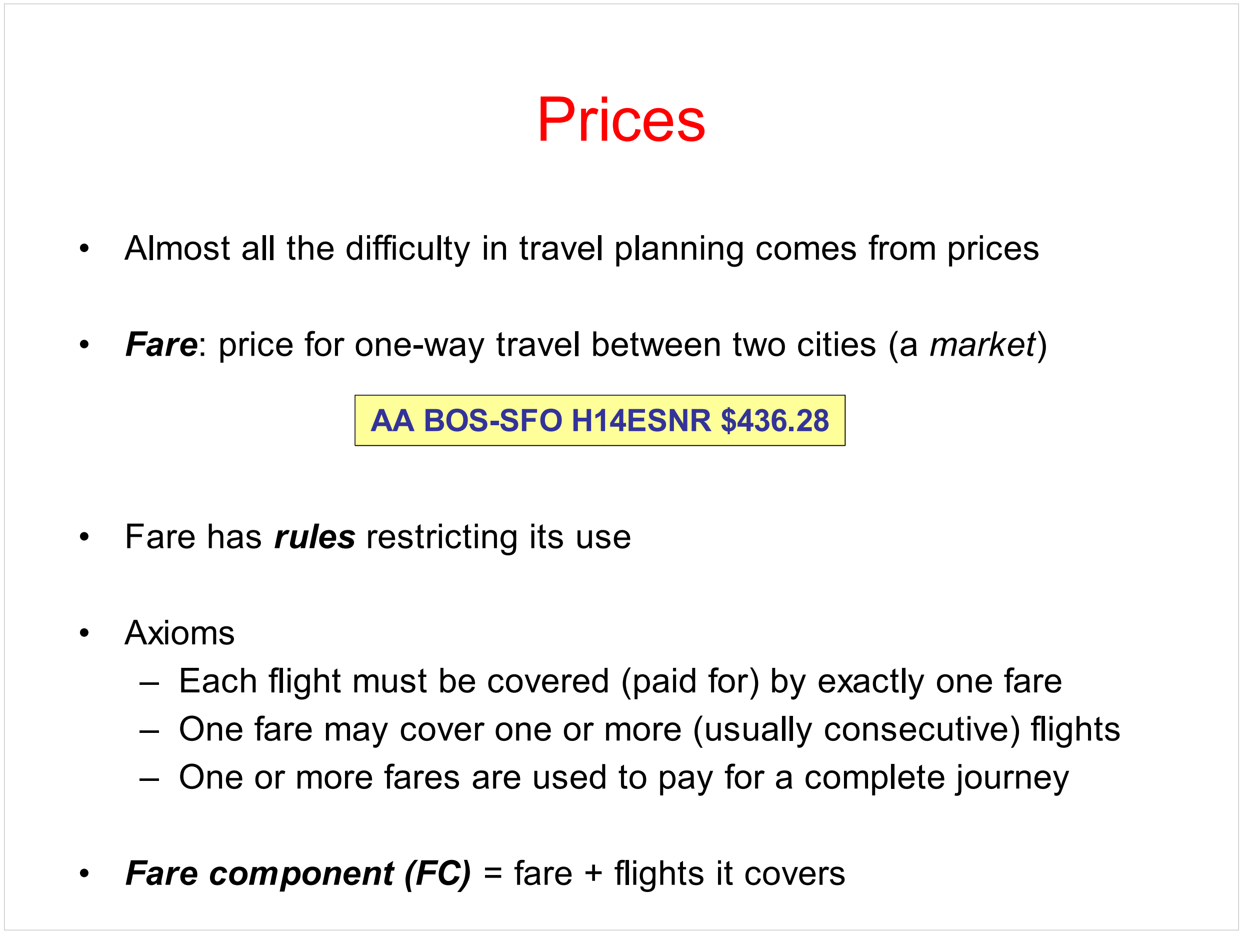
Air travel prices and paths have a very complex relationship with each other. As will be shown, it's provably hard to use prices to inform flight selection if one is searching for the cheapest route. This gives air travel planning a very different character from many other easier route planning problems, such as car route search. Dynamic programming techniques like those in Dijkstra's algorithm can not be used to reduce the exponential number of paths to a polynomial algorithm, because when prices are considered the state of a search can not be summarized by the current position: prices depend on the entire flight history.

One interesting note is that while standard algorithms can be applied to the flight graph to generate shortest paths (or the *k* shortest paths), it is a considerable challenge to develop algorithms that can enumerate the best paths fast enough for use in a planning system that may need to consider thousands to millions of routes within tens of seconds.

Outline

- Introduction
- Flights
- **How airline prices work**
- Complexity of travel planning
- Demos
- Seat availability
- Further reading

Airline prices are much, much more complex than prices for almost any other good or service, and are not intuitive. It is the prices, not route planning, that make air travel planning a difficult problem.



Prices

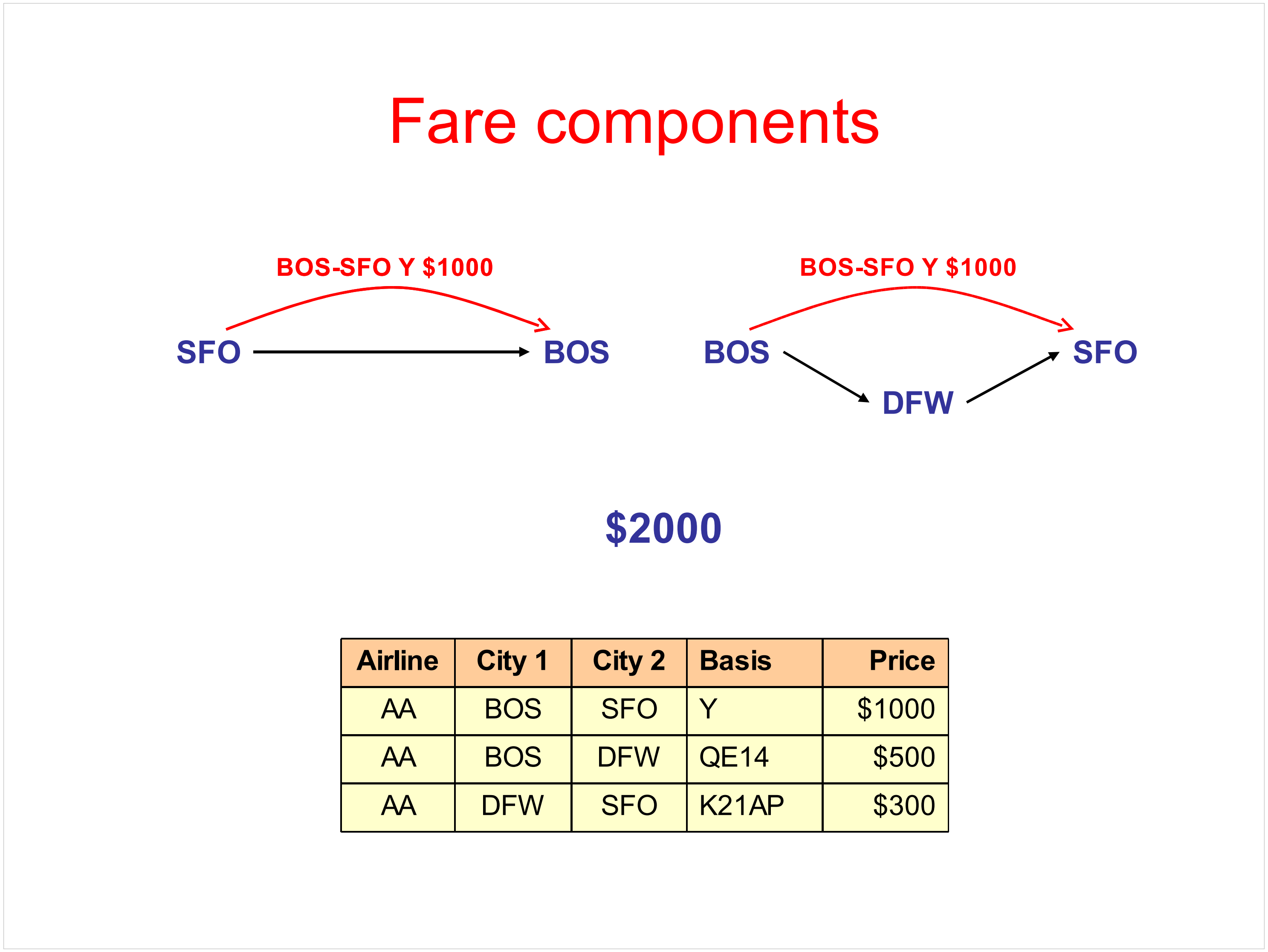
- Almost all the difficulty in travel planning comes from prices
- ***Fare***: price for one-way travel between two cities (a *market*)

AA BOS-SFO H14ESNR $436.28

- Fare has ***rules*** restricting its use
- Axioms
  - Each flight must be covered (paid for) by exactly one fare
  - One fare may cover one or more (usually consecutive) flights
  - One or more fares are used to pay for a complete journey
- ***Fare component (FC)*** = fare + flights it covers

The atomic unit of price in the airline industry is called a **fare**. A fare is a price an airline offers for one-way travel between two cities, usually good for travel in either direction. Such a city pair is known as a **market**. Each fare is given an alphanumeric identifier called a basis or fare basis code, H14ESNR in the example. Each fare is published with a set of rules restricting its use.

Fundamentally, each flight must be paid for by exactly one fare, but a single fare may pay for more than one flight. Multiple fares may be combined to pay for all the flights in a journey. The airline industry uses the term **fare component (FC)** to refer to a fare and the flights it pays for (covers).



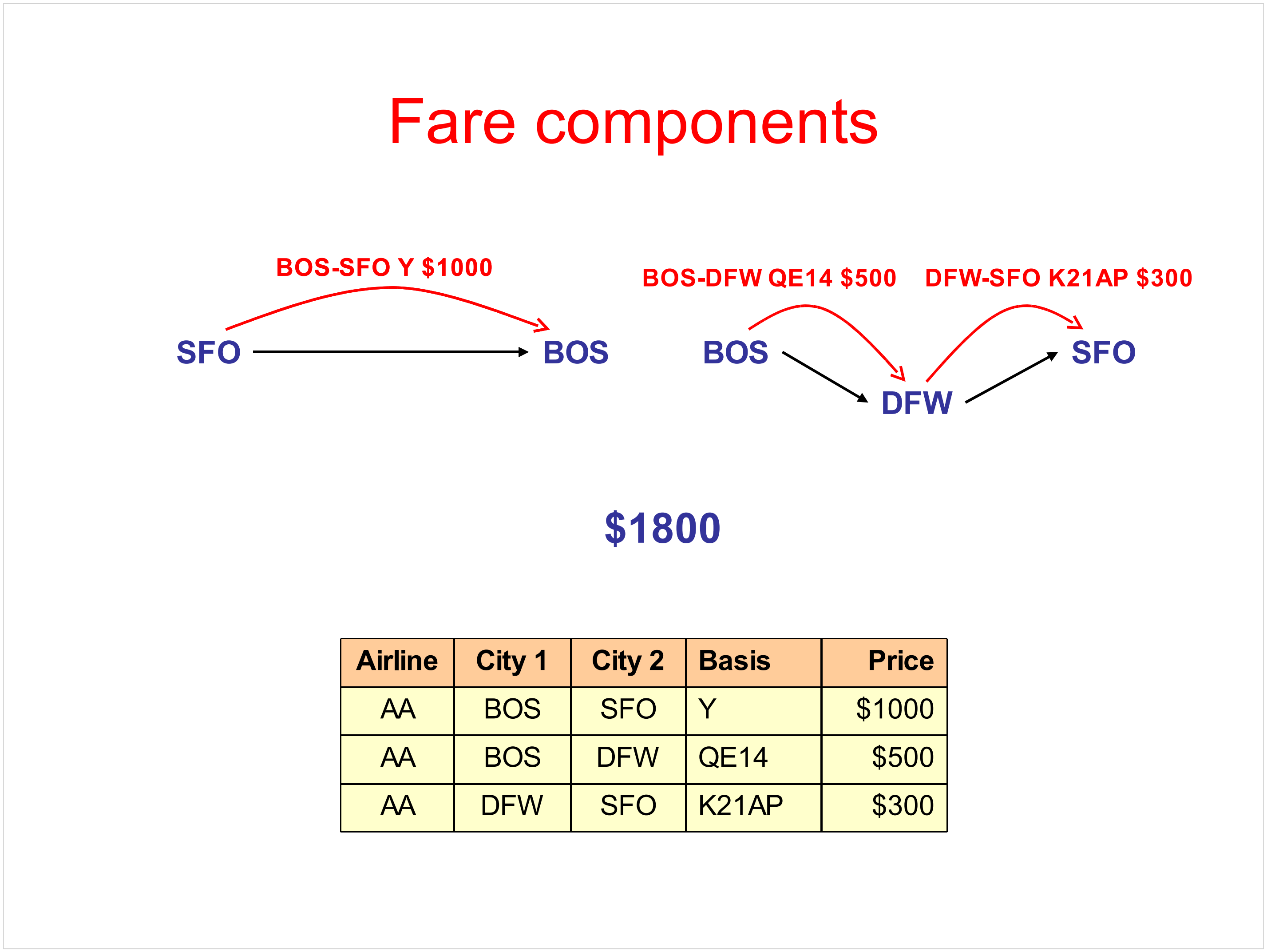
Fare components

$2000

Airline	City 1	City 2	Basis	Price
AA	BOS	SFO	Y	$1000
AA	BOS	DFW	QE14	$500
AA	DFW	SFO	K21AP	$300

As an example, suppose that a round-trip journey on American Airlines consists of the three flights shown, nonstop from San Francisco to Boston, then back to San Francisco through Dallas, and that American Airlines publishes the 3 fares listed in the table. For example, the $1000 "Y" fare can be used to pay for travel from BOS to SFO or from SFO to BOS. Then, assuming the Y fare's rules are satisfied, one way to pay for this journey is by using the Y to pay for the single outbound flight, and the Y again to pay for both return flights, for a total price of $2000.

Copyright 2003 ITA Software



Fare components

$1800

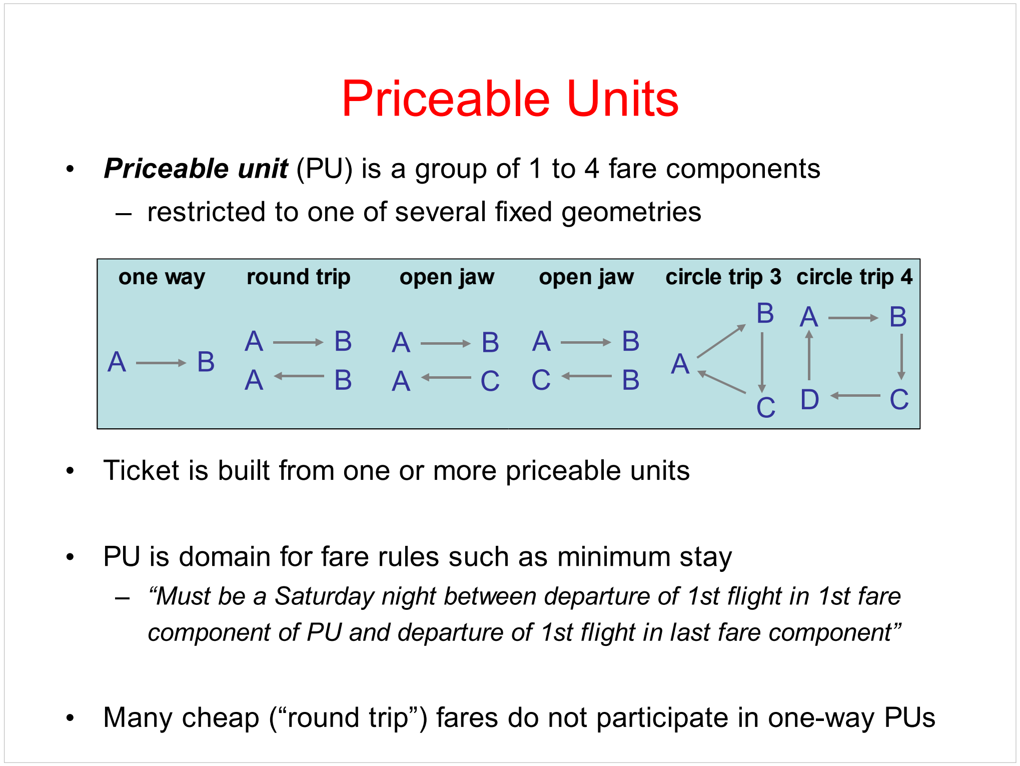
Airline	City 1	City 2	Basis	Price
AA	BOS	SFO	Y	$1000
AA	BOS	DFW	QE14	$500
AA	DFW	SFO	K21AP	$300

Assuming the QE14 and K21AP fares' rules are met, there is another valid way to pay for the same flights, using a BOS-DFW fare to pay for the first flight of the return and a DFW-SFO fare to pay for the second flight. In this case the total is only $1800.

In general, a traveler may choose whatever fare combination for a ticket they wish so long as all fare rules are satisfied: usually they'll want to choose the cheapest combination, but (for example) they may choose more expensive fares that permit refunds or first-class travel if they so desire.

Fares are not the end of the story. There is another unit of representation between a fare component and a complete ticket, called a **priceable unit (PU)**. A priceable unit is a collection of 1 to 4 fare components in one of a small number of fixed geometric shapes. An analogy is that if fares are atoms, priceable units are the molecules used to build complete tickets. It's not entirely correct, but a good working intuition is that a priceable unit is the smallest group of flights and fares that could be sold on their own.

The simplest priceable unit is the **one way PU** built from one fare component. Others include **round trip PUs** and **circle trip PUs** built from 2, 3 or 4 FCs that form a loop. **Open jaw PUs** are like circle trips with one FC missing – usually the open gap must be shorter than the distance flown in any of the flown FCs.



Priceable Units

- ***Priceable unit*** (PU) is a group of 1 to 4 fare components
  – restricted to one of several fixed geometries

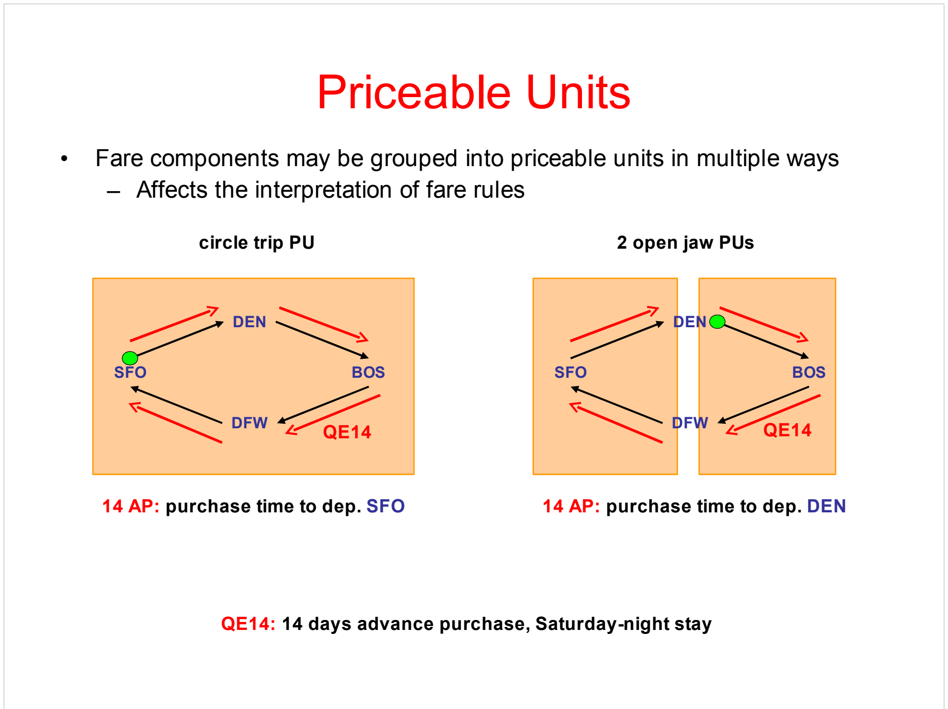
one way
round trip
open jaw
open jaw
circle trip 3
circle trip 4
A → B
A → B
A ← B
A → B
A ← C
A → B
C ← B
A → B → C → A
A → B → C → D → A

- Ticket is built from one or more priceable units
- PU is domain for fare rules such as minimum stay
  – *"Must be a Saturday night between departure of 1st flight in 1st fare component of PU and departure of 1st flight in last fare component"*
- Many cheap ("round trip") fares do not participate in one-way PUs

The airlines use priceable units as a domain in which many fare rules apply. For example, many of the least expensive fares have a **Saturday night stay** restriction. The airlines often define such a restriction by reference to the priceable unit containing the fare in question: "there must be a Saturday night between the departure of the 1st flight in the 1st FC of the PU and the departure of the 1st flight in the last FC of the PU". It is common for fares to require that other fares in the same PU be on the same airline.

Importantly, many of the cheapest fares, known as **round trip fares** (to be distinguished from round trip PUs and round trip journeys) have rules that prohibit their use in one way PUs. That means that for a round trip fare to be used, it must be combined in a priceable unit with at least one other fare; it is round trip fares that usually impose Saturday night stay restrictions. The net effect is that to use a (cheap) round trip fare one must fly a journey with at least two flights separated by a Saturday night, and must use at least two fares to pay for the journey.

As will be discussed, priceable units are responsible for much of the theoretical difficulty in pricing air travel.



Priceable Units

- Fare components may be grouped into priceable units in multiple ways
  - Affects the interpretation of fare rules

14 AP: purchase time to dep. SFO

14 AP: purchase time to dep. DEN

QE14: 14 days advance purchase, Saturday-night stay

A specific set of flights may be partitioned into fares and priceable units in many ways. The above diagram shows two ways to partition the four flights of a SFO to BOS round-trip journey. In both cases, four fares are used, one per flight. On the left, the four fare components are grouped into a single circle-trip priceable unit. On the right, they are grouped into two open-jaw priceable units. Each black line represents a flight and each red a fare. The boxes represent priceable units.

Because rules like the Saturday night stay restriction and advance purchase restrictions depend on how flights and fares are divided into priceable units, it is important how fare components are grouped into priceable units, and this non-determinism adds to the search space. Suppose the third fare, a BOS-DFW QE14 fare, has a 14-day advance purchase restriction and a Saturday night stay restriction, and the three other fares do not have important restrictions.

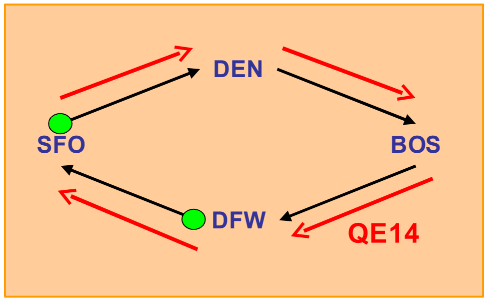
The 14-day advance purchase restriction will likely be defined as requiring 14 days to pass between the time of reservation and the departure of the first flight in the priceable unit of the fare with the restriction. In the case of the single circle-trip priceable unit that flight is the initial departure from SFO, marked in green. In the case of two open-jaw PUs, that flight is the later departure from DEN. Thus, from the perspective of the 14-day restriction, it is more likely the restriction will be met in the two-PU choice.

Copyright 2003 ITA Software

Priceable Units

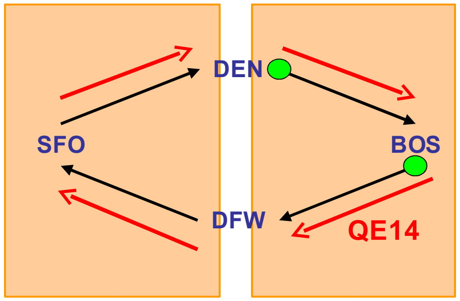
- Fare components may be grouped into priceable units in multiple ways
  - Affects the interpretation of fare rules

circle trip PU



14 AP: **purchase time to dep. SFO**
SAT: **dep. SFO to dep. DFW**

2 open jaw PUs

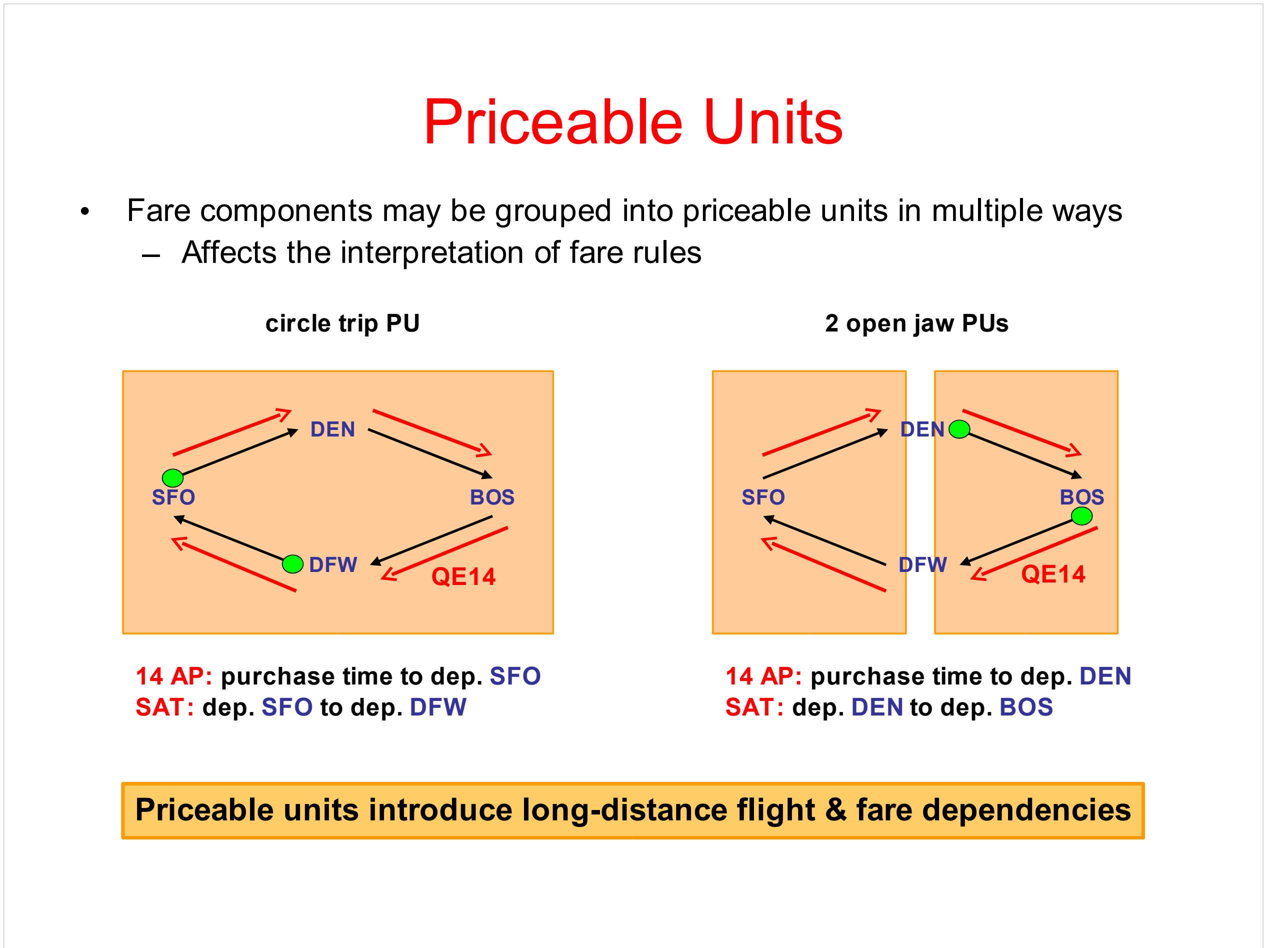


14 AP: **purchase time to dep. DEN**
SAT: **dep. DEN to dep. BOS**

QE14: **14 days advance purchase, Saturday-night stay**

The QE14 fare's Saturday night stay restriction is likely defined as requiring that the first flight of the last fare component in the priceable unit be on or after the first Sunday following the departure of the first flight of the first fare component in the PU. In the single circle-trip PU case, the relevant times will be the departure from SFO and the departure from DFW. In the two open-jaw PUs case, the relevant times will be the departure from DEN and the departure from BOS. Therefore from the perspective of the Saturday night stay restriction, the circle-trip choice is advantageous (more time passes between the two measurements).

Copyright 2003 ITA Software



Priceable Units

- Fare components may be grouped into priceable units in multiple ways
  - Affects the interpretation of fare rules

14 AP: purchase time to dep. SFO
SAT: dep. SFO to dep. DFW

14 AP: purchase time to dep. DEN
SAT: dep. DEN to dep. BOS

Priceable units introduce long-distance flight & fare dependencies

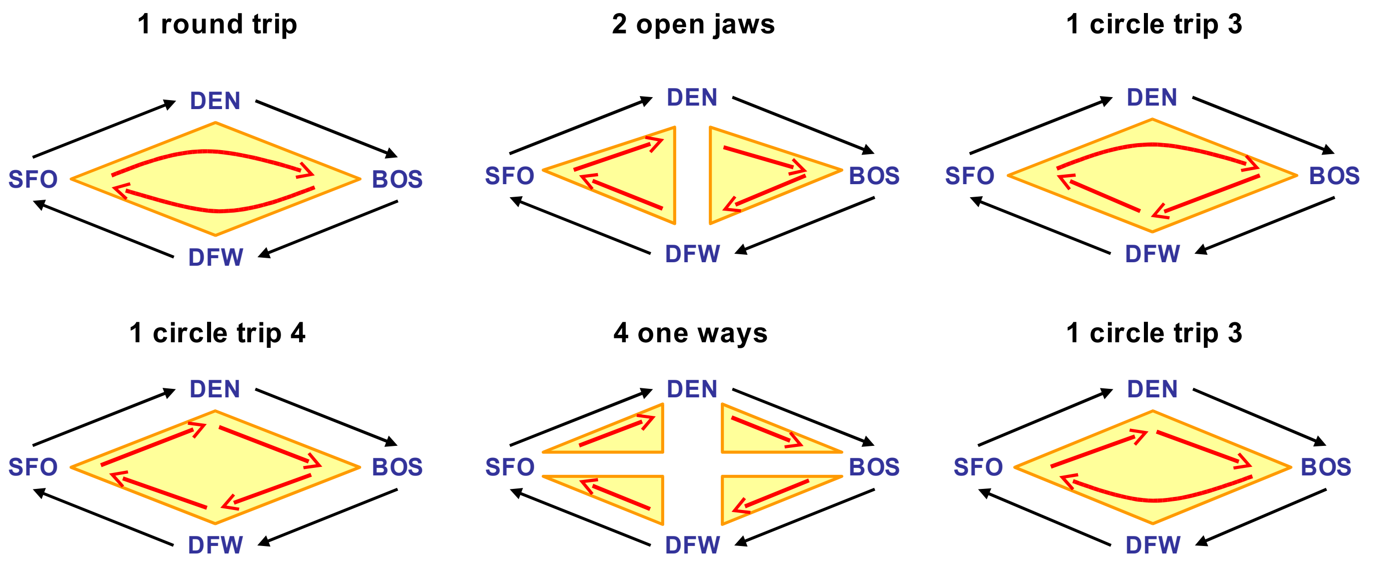
Priceable units add to the complexity of ticket pricing in two ways. First, the choice of different partitions of fare components into priceable units increases the search space. Second, and more importantly, priceable units introduce long-distance dependencies between different parts of a trip. Notice in the case of the circle-trip priceable unit how the Saturday night stay restriction tests the combination of the departure times of two flights widely separated in both time and on the ticket. Nothing limits the distance between different fare components of a priceable-unit, which has huge ramifications for search algorithms: most efficient search techniques demand that constraints be local.

Fare rules can and usually do restrict fare combinations within a priceable unit as well as flight combinations. For example, a fare may require that other fares in the same PU be published by the same airline and have similar basis codes.

Copyright 2003 ITA Software

Priceable Units

- Flights can be broken into fare components and priceable units in many ways



1 round trip

2 open jaws

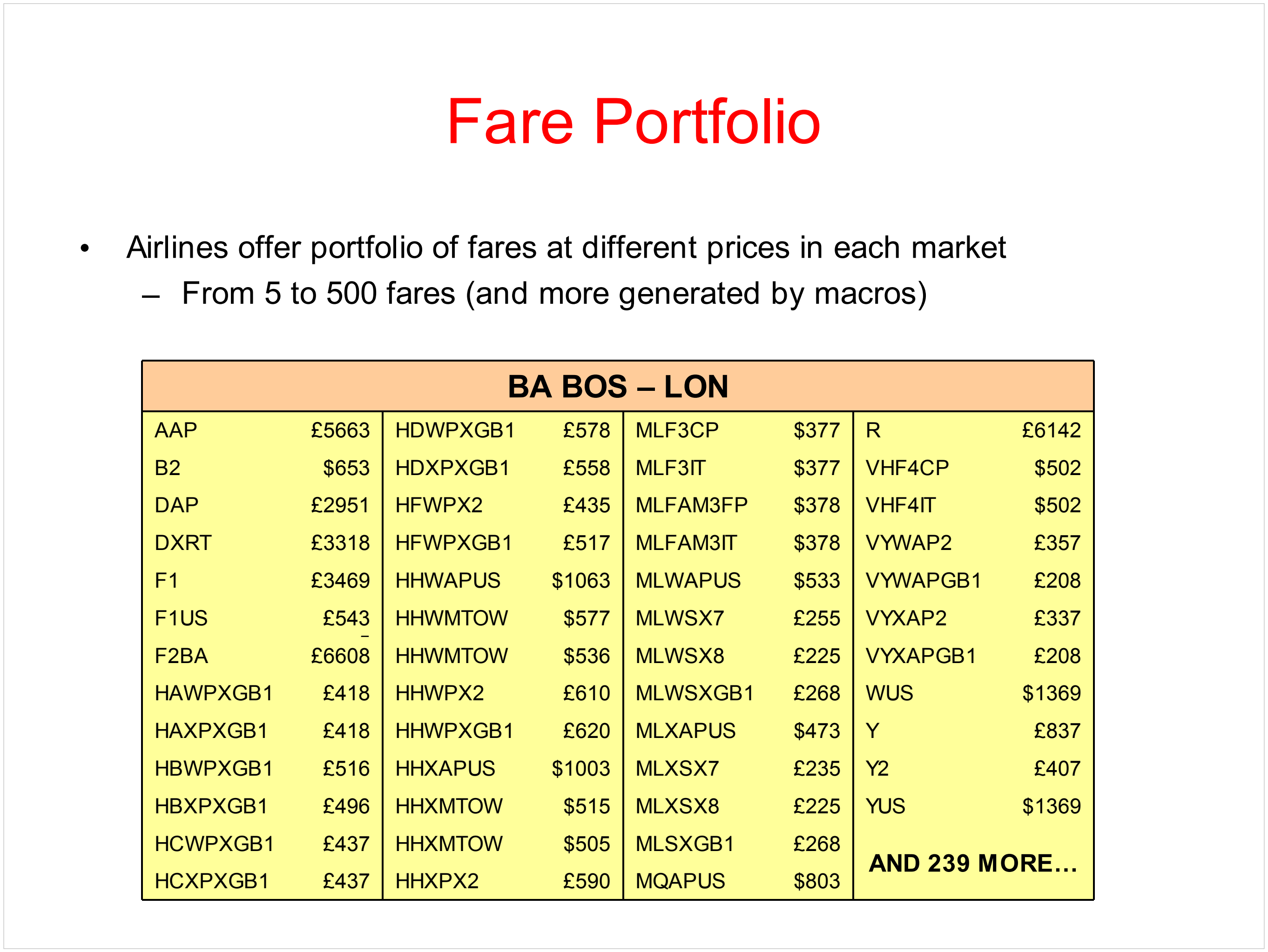
1 circle trip 3

1 circle trip 4

4 one ways

1 circle trip 3

A specific set of flights may be partitioned into fares and priceable units in many, many ways. For the four flights shown above 6 possibilities are shown (there are more). Each red line represents a fare component and each yellow polygon a priceable unit. For example, a round trip PU may be used with one fare paying for both outbound flights and one for both return flights. Alternatively two open jaw priceable units can be used, each containing two fares, each fare paying for one flight.



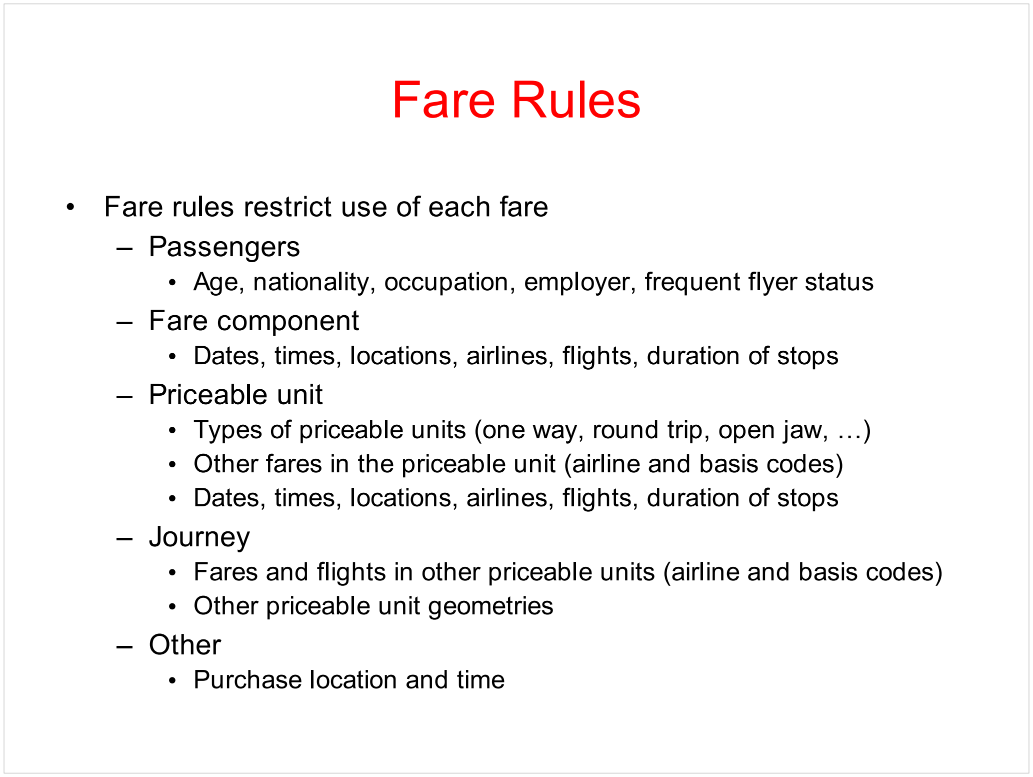
Fare Portfolio

- Airlines offer portfolio of fares at different prices in each market
  - From 5 to 500 fares (and more generated by macros)

BA BOS – LON							
AAP	£5663	HDWPXGB1	£578	MLF3CP	$377	R	£6142
B2	$653	HDXPXGB1	£558	MLF3IT	$377	VHF4CP	$502
DAP	£2951	HFWPX2	£435	MLFAM3FP	$378	VHF4IT	$502
DXRT	£3318	HFWPXGB1	£517	MLFAM3IT	$378	VYWAP2	£357
F1	£3469	HHWAPUS	$1063	MLWAPUS	$533	VYWAPGB1	£208
F1US	£543	HHWMTOW	$577	MLWSX7	£255	VYXAP2	£337
F2BA	£6608	HHWMTOW	$536	MLWSX8	£225	VYXAPGB1	£208
HAWPXGB1	£418	HHWPX2	£610	MLWSXGB1	£268	WUS	$1369
HAXPXGB1	£418	HHWPXGB1	£620	MLXAPUS	$473	Y	£837
HBWPXGB1	£516	HHXAPUS	$1003	MLXSX7	£235	Y2	£407
HBXPXGB1	£496	HHXMTOW	$515	MLXSX8	£225	YUS	$1369
HCWPXGB1	£437	HHXMTOW	$505	MLSXGB1	£268	**AND 239 MORE…**	
HCXPXGB1	£437	HHXPX2	£590	MQAPUS	$803		

Within every market airlines publish not just one, but many fares. For example, British Airways offers more than 280 fares between Boston and London. Some airlines publish as many as 1000 fares in international markets, and there are various “macro” systems in the industry that can double or triple that number.

Each fare has a different price and a different fare basis code. A reasonable question is why the airlines publish so many fares, if the traveler is free to choose the cheapest. The answer is that each fare has its own rules restricting use, so that in any particular circumstances only a small subset of the fares may be usable.



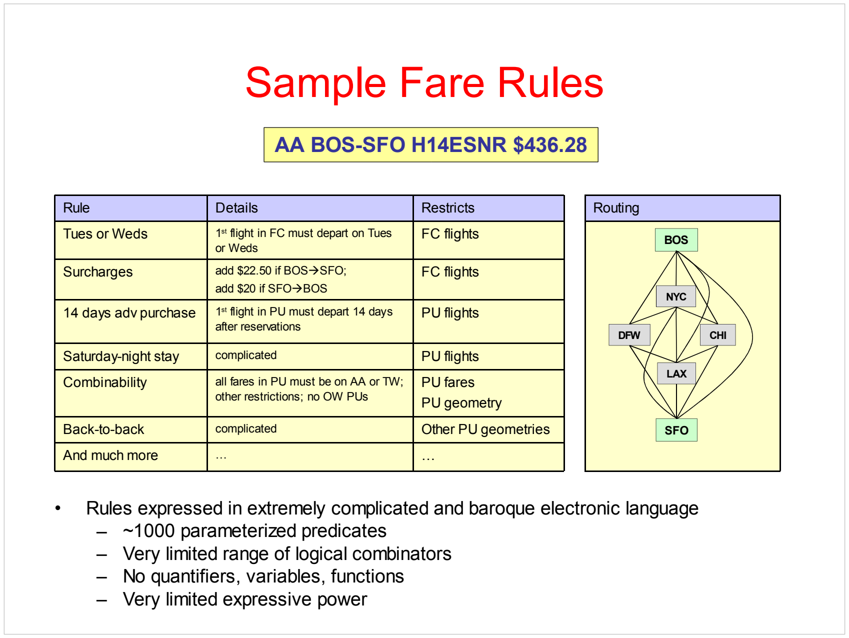
Fare Rules

- Fare rules restrict use of each fare
  - Passengers
    - Age, nationality, occupation, employer, frequent flyer status
  - Fare component
    - Dates, times, locations, airlines, flights, duration of stops
  - Priceable unit
    - Types of priceable units (one way, round trip, open jaw, …)
    - Other fares in the priceable unit (airline and basis codes)
    - Dates, times, locations, airlines, flights, duration of stops
  - Journey
    - Fares and flights in other priceable units (airline and basis codes)
    - Other priceable unit geometries
  - Other
    - Purchase location and time

Fare rules can restrict most any aspect of a journey. Often they restrict the passengers who may use the fare – limiting special discounts to children, for example -- and the travel agents who may sell the fare. Many fares include restrictions on the flight numbers, locations and departure times of flights within the fare's fare component. Typically fares within the United States prohibit stops of longer than 4 hours within a FC. Fare rules may also impose restrictions at the priceable unit domain, such as the Saturday night stay restriction that depends on the times of flights from the first and last FC in the fare's PU simultaneously. Rules very often restrict the fares that can combine in a priceable unit, such as requiring them to be on the same airline or have similar fare basis codes.

It is even possible for a fare to restrict parts of the journey outside the fare's priceable unit. As will be shown, this greatly increases the difficulty of the search problem.

One passenger's fares may restrict another's, such as cheap **companion** fares that force a second or third passenger to accompany the first, and that restrict the fares those passengers may use to pay for common flights. This can cause an exponential increase in the complexity of search with the number of passengers on the trip.



Sample Fare Rules

AA BOS-SFO H14ESNR $436.28

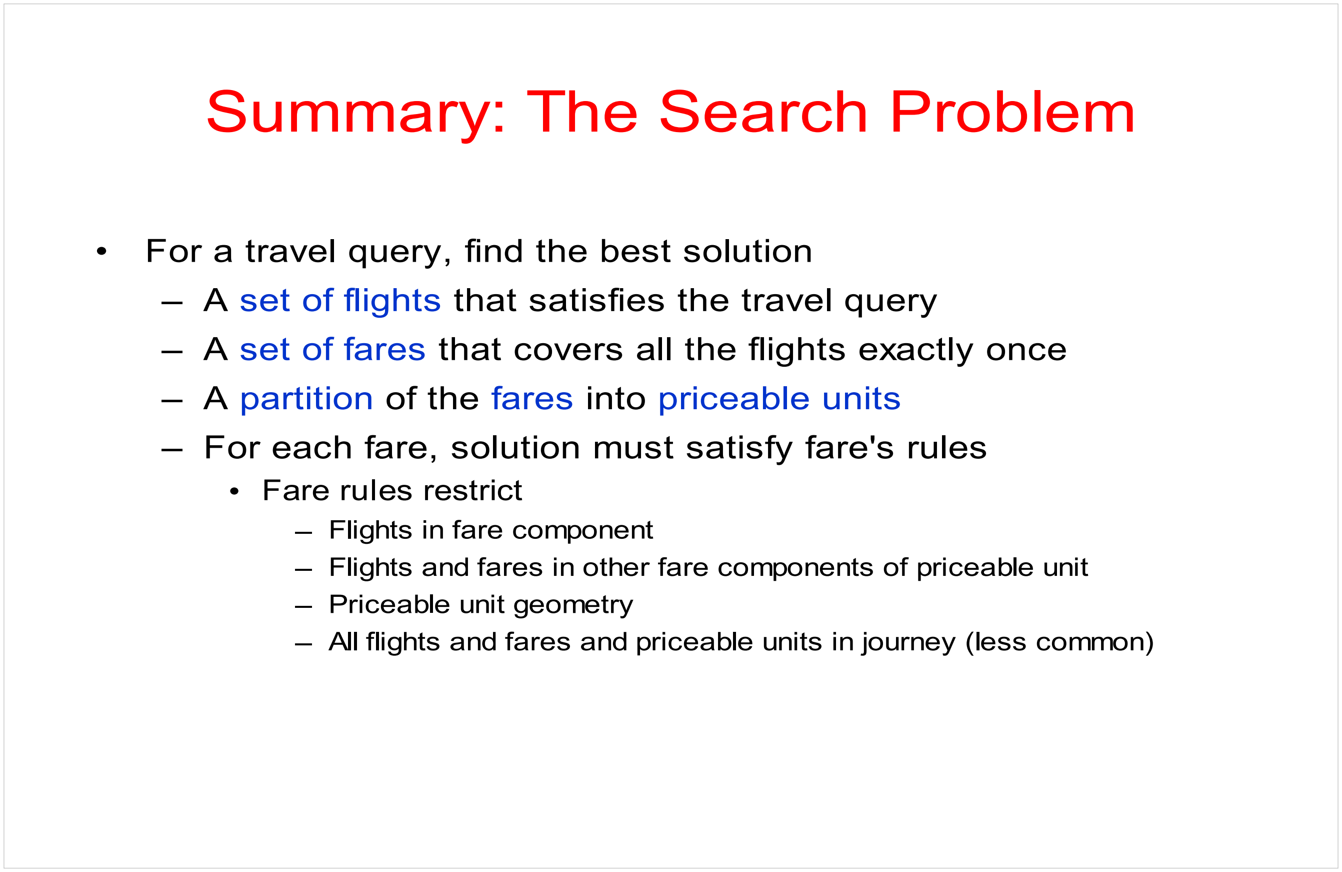
Rule	Details	Restricts
Tues or Weds	1st flight in FC must depart on Tues or Weds	FC flights
Surcharges	add $22.50 if BOS→SFO; add $20 if SFO→BOS	FC flights
14 days adv purchase	1st flight in PU must depart 14 days after reservations	PU flights
Saturday-night stay	complicated	PU flights
Combinability	all fares in PU must be on AA or TW; other restrictions; no OW PUs	PU fares PU geometry
Back-to-back	complicated	Other PU geometries
And much more	…	…

- Rules expressed in extremely complicated and baroque electronic language
  - ~1000 parameterized predicates
  - Very limited range of logical combinators
  - No quantifiers, variables, functions
  - Very limited expressive power

Returning to the AA BOS-SFO H14ESNR fare, here is a subset of the fare's rules. The fare restricts the first flight in its fare component to be on Tuesday or Wednesday. It has a 14 day advance purchase restriction defined on the entire priceable unit. That means that if the fare is used to pay for the return portion of a trip, then even if the flights the fare is paying for leave 21 days after reservations the fare may not be usable, if the outbound portion of the trip took place only 7 days after reservation. The fare, a round trip fare, prohibits use in one way PUs. The fare includes a complicated **back to back** restriction that limits the geometries of other priceable units in the journey.

One part of the fare's rules is known as the **routing**. The routing is a directed graph of permitted routes within the fare component. For example, the H14ESNR fare permits non-stop travel between BOS and SFO, but also permits stops in NYC, CHI, DFW and LAX (but not in both DFW and CHI). Thus, the price is the same, $436.28, whether one takes one flight or four, despite the wildly different cost to the airline of providing the service. In fact, since many of the cheapest fares on popular business routes prohibit non-stop travel, it is commonly the case that airlines' prices and expenses are anti-correlated, something to think about when you read about airline bankruptcy filings!

Fare rules are expressed in an extremely complicated and baroque electronic language, built from hundreds of parameterized predicates joined by sometimes bizarre logical combinators. Although the language has to be very big to express all of the many airlines' restrictions, the language is not nearly as expressive as a general purpose programming language. There are no functions, variables, quantifiers, scoping operators, iterators, etc. A non-profit company, **ATP** (Airline Tariff Publishing), owned jointly by many airlines, manages and distributes fares and rules electronically and working jointly with airlines and search companies defines the electronic representations.

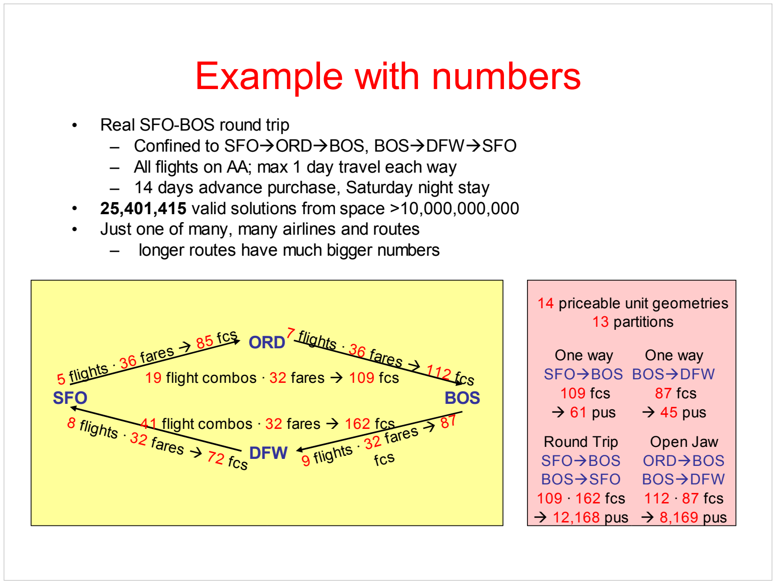


Summary: The Search Problem

- For a travel query, find the best solution
  - A set of flights that satisfies the travel query
  - A set of fares that covers all the flights exactly once
  - A partition of the fares into priceable units
  - For each fare, solution must satisfy fare's rules
    - Fare rules restrict
      - Flights in fare component
      - Flights and fares in other fare components of priceable unit
      - Priceable unit geometry
      - All flights and fares and priceable units in journey (less common)

Although clearly not a practical algorithm, in the tradition of non-deterministic search one might summarize the air travel search problem as: 1. Guess a set of flights that satisfies the travel query; 2. Guess a set of fares and a mapping from flights to fares that covers all flights exactly once; 3. Guess a partitioning of the fares into priceable units; and 4. Verify that all fares' rules are met, where the rules may conceivably test all flights and fares in the journey but take a very restricted form.

In fact this isn't the entire truth. For international travel some additional checks (so-called **IATA checks**) require comparing one answer against another, so that one answer can't be used if another more expensive one is close by in a technical sense. While practically this adds greatly to the difficulty of international search, IATA checks are too complex to explore here. Fortunately, they only compare potential fare variations for the same set of flights, which limits their effect on the theoretical computational complexity of the air travel planning problem.



Example with numbers

- Real SFO-BOS round trip
  - Confined to SFO→ORD→BOS, BOS→DFW→SFO
  - All flights on AA; max 1 day travel each way
  - 14 days advance purchase, Saturday night stay
- **25,401,415** valid solutions from space >10,000,000,000
- Just one of many, many airlines and routes
  - longer routes have much bigger numbers

ORD
SFO
BOS
DFW
5 flights · 36 fares → 85 fcs
7 flights · 36 fares → 112 fcs
19 flight combos · 32 fares → 109 fcs
41 flight combos · 32 fares → 162 fcs
8 flights · 32 fares → 72 fcs
9 flights · 32 fares → 87 fcs

14 priceable unit geometries
13 partitions

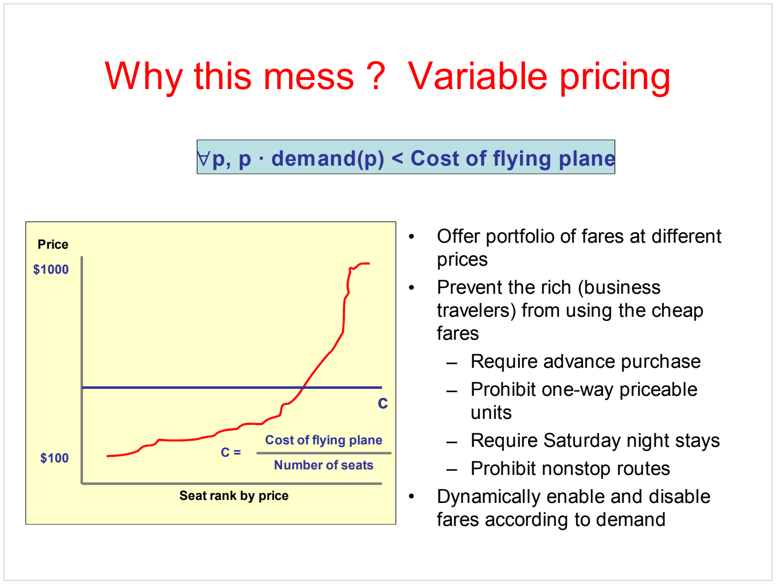
One way SFO→BOS 109 fcs → 61 pus	One way BOS→DFW 87 fcs → 45 pus
Round Trip SFO→BOS BOS→SFO 109 · 162 fcs → 12,168 pus	Open Jaw ORD→BOS BOS→DFW 112 · 87 fcs → 8,169 pus

This is a real SFO-BOS round trip example with numbers, constrained to a particular American Airlines route (SFO to BOS through ORD, then after a Saturday night back from BOS to SFO through DFW) with travel each way limited to one calendar day. There are a total of 25,401,415 valid solutions from a space of more than 10,000,000,000 combinations of flights and fares and priceable units.

Exploring the diagram, from SFO to ORD on the travel date AA offers 5 flights and publishes 36 fares in the market. After testing those fare rules that restrict fare component flights and times, there are a total of 85 possible SFO to ORD fare components (from a space of 5 * 36 = 180). Similarly, from SFO to BOS the 5 SFO to ORD flights combine with the 7 ORD to BOS flights to produce 19 flight combinations (instead of 35, because of time constraints). AA publishes 32 SFO-BOS fares, but the 19 flight combinations * 32 fares only produce 109 fare components after checking appropriate rules.

When constructing priceable units, the 112 outbound ORD to BOS fare components combine with the 87 return BOS to DFW fare components to produce 8,169 open jaw priceable units. Again this is less than 112 * 87 because fare rules limit combinations. There are 14 possible priceable unit geometries, and 13 ways to use the geometries to cover all four flights.

Putting together all the possible ways to combine all the possible priceable units into a complete ticket, there are 25,401,415 solutions. This is just for this particular airline and route – it represents a very small portion of the search space that an engine would need to consider for an unrestricted SFO to BOS round trip journey.



Why this mess ? Variable pricing

$\forall p, p \cdot \text{demand}(p) < \text{Cost of flying plane}$

- Offer portfolio of fares at different prices
- Prevent the rich (business travelers) from using the cheap fares
  - Require advance purchase
  - Prohibit one-way priceable units
  - Require Saturday night stays
  - Prohibit nonstop routes
- Dynamically enable and disable fares according to demand

Why this mess? Why so many fares, such complicated rules, the logic of priceable units, and so on? The answer is often called **variable pricing**. Various airline economists make the following claim: there is no price such that the price times the demand at the price equals the cost of flying a large jet. There are a lot of technical issues that can be raised with their argument, but leaving those aside the argument is that if the airline charges $1 per ticket of course the plane will fill, but the total revenue of $150 barely pays for an hour of a pilot's salary. If they charge $1000 a ticket then if they could fill the plane they'd make a fortune, but only a small number of people are willing to fly at that price, so again they can't equal the fixed costs of flying a plane. But if the airline can make those who are willing to pay it pay $1000, and others pay $800, and others $500, maybe down to $100 or so, then the sum total over all passengers is sufficient to pay for the fixed costs. In fact, some estimates put the incremental cost of flying a single passenger as low as $30 (for the meal and baggage and ticket handling), so that *once the airline has committed to flying the plane* it is in their interest to sell a ticket for $30 rather than let a seat go empty. But they must keep those who can pay more from buying their ticket at low prices, a tough balancing act.

The airlines solve this problem in two ways, collectively called **revenue management.** The first is to use fare prices and fare rules to construct a system wherein the cheapest fares have restrictions that increase their perceived cost for a business traveler to the point where the business traveler will choose to buy more expensive fares. For example, cheap fares require round trip travel, prohibit non stop flights and ticket refunds, et cetera. But the cheap fares remain available for leisure travelers with more flexibility, for whom the extra restrictions are not so onerous. The second way, discussed later, consists of dynamically deciding whether to sell a given fare for a flight based on how much demand there is for the flight. For example, if a flight is not filling, lower priced fares are made available (on the grounds that it's better to get some money than none) but on high-demand flights only the most expensive fares are available.

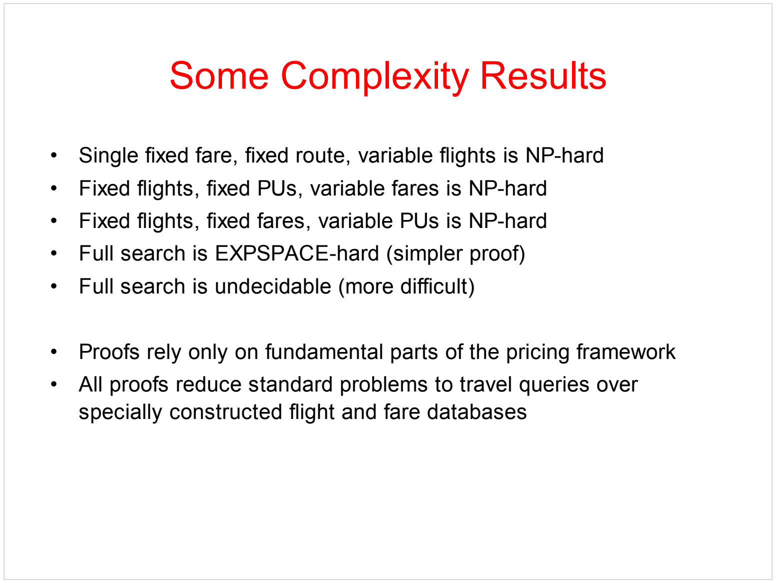
Outline

- Introduction
- Flights
- How airline prices work
- **Complexity of travel planning**
- Demos
- Seat availability
- Further reading

Copyright 2003 ITA Software

Next are 4 proof sketches of the complexity of different aspects of the air travel planning and pricing problem.

It would in fact be easy to show that air travel planning is hard if airlines could publish any type of rule with a fare, as opposed to the restricted set they commonly use and that can be encoded in the industry's electronic formats. Except for the last one, the following proofs will rely only on the most fundamental parts of the airlines' pricing framework, used routinely. And except the last one, all the proofs are fairly simple and reduce standard computer science problems known to be difficult to the question of whether there is a valid ticket for a query over specially constructed flight and fare databases.



Some Complexity Results

- Single fixed fare, fixed route, variable flights is NP-hard
- Fixed flights, fixed PUs, variable fares is NP-hard
- Fixed flights, fixed fares, variable PUs is NP-hard
- Full search is EXPSPACE-hard (simpler proof)
- Full search is undecidable (more difficult)

- Proofs rely only on fundamental parts of the pricing framework
- All proofs reduce standard problems to travel queries over specially constructed flight and fare databases

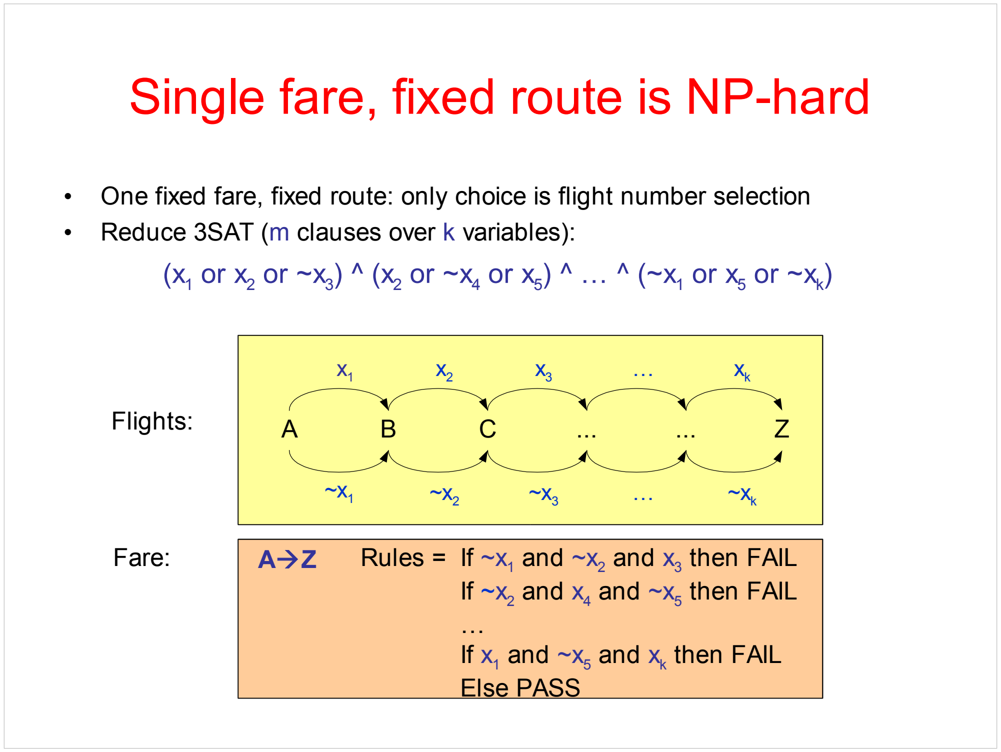
1. Even if the airlines publish only a single fare (with every ticket a single one way PU), and all the airports in a passenger's itinerary are fixed, so that the only remaining choice is what flights to use between the consecutive airports, deciding whether there is a valid ticket is NP-hard.

2. Fixing the flights and priceable units, but leaving open the choice of fares to pay for each flight, deciding whether there is a valid ticket is NP-hard.

3. Removing bounds on the size of solutions, the general question of whether there is a ticket for a query is EXPSPACE-hard. That is, air travel planning is *at least as hard* (it might be harder) as deciding whether a computer program that can use space exponentially bigger than the input halts. EXPSPACE-hard problems are (thought to be) much harder than NP-complete problems like the traveling salesman problem, and even much harder than PSPACE-complete problems like playing games optimally. There is no practical hope that computers will ever be able to solve EXPSPACE-hard problems perfectly, even if quantum computing becomes a reality.

4. The final result shows that just finding out whether there is a valid solution for a query is actually harder then EXPSPACE-complete: it is unsolvable (undecidable). The question of whether a valid ticket exists can not be solved for all databases and all queries no matter how long a computer thinks. However the full proof of this result is considerably more complex than the EXPSPACE-hard proof without offering any greater understanding of the problem.

One interesting result not written up here is that even completely fixing the flights and fares of a ticket, so that the only remaining question is how to partition the fares into priceable units, is NP-complete. This is interesting because only flight and fare information makes its way onto printed tickets, not the grouping of fares into PUs. Therefore the problem of just validating a printed ticket is worst-case NP-complete, though it is rarely difficult in practice.



Single fare, fixed route is NP-hard

- One fixed fare, fixed route: only choice is flight number selection
- Reduce 3SAT (m clauses over k variables):

$$(x_1 \text{ or } x_2 \text{ or } {\sim}x_3) \wedge (x_2 \text{ or } {\sim}x_4 \text{ or } x_5) \wedge \ldots \wedge ({\sim}x_1 \text{ or } x_5 \text{ or } {\sim}x_k)$$

Flights:

Fare:

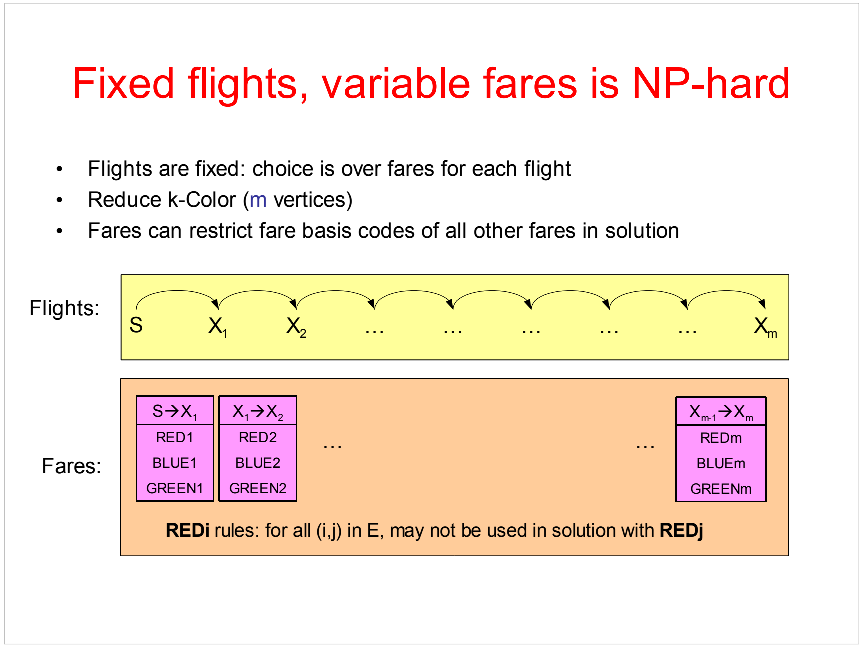
A→Z Rules = If ${\sim}x_1$ and ${\sim}x_2$ and x_3 then FAIL
If ${\sim}x_2$ and x_4 and ${\sim}x_5$ then FAIL
…
If x_1 and ${\sim}x_5$ and x_k then FAIL
Else PASS

It is possible to reduce the NP-complete 3SAT problem to the question of whether there is a combination of flights that satisfies a particular fare's rules. For a given 3SAT expression over k variables one can construct a sequence of k+1 airports with exactly two flights between the i^{th} and $(i+1)^{th}$ airport, one flight representing the assignment of true to the i^{th} variable, the other false. It is possible to construct a single fare from the first to the last airport such that the fare's rules enforce the 3SAT logical expression. This is not entirely trivial to show, in that the limited rule language forces one to apply DeMorgan's rule to the expression and to encode the restrictions on variables as a certain kind of restriction on the flight numbers that can be used between two specific airports, but the fare rule language is (just barely) expressive enough to do this.

This simple fact is interesting because the airlines often advertise fares to the public: "$100 special from Boston to San Francisco!" Even looking at that fare's rules, it may not necessarily be easy to find a sequence of flights that will satisfy them.

In point of fact, the particular rule mechanisms used in the complete proof are not usually problematic for search, but the combined set of all types of fare component flight and route and time restrictions can make it very difficult to find valid flight sequences for many fares between airports separated by 3 or more flights (or to prove that no valid sequence exists).

Fixing a sequence of flights but introducing a choice of fares to pay for each flight, it is again possible to reduce an NP-complete problem, k-coloring a graph (coloring graph vertices with k colors such that no two connected vertices share the same color). In this construction vertices are represented by flights and the color of a vertex by the choice of fare used to pay for the flight. A sequence of flights is constructed, one per vertex, and a query posed between the endpoints. The fare database is constructed to contain one fare per combination of vertex and color. In the figure, the fares for the flight representing vertex i are named REDi, BLUEi and GREENi (for 3-color).



Fixed flights, variable fares is NP-hard

- Flights are fixed: choice is over fares for each flight
- Reduce k-Color (m vertices)
- Fares can restrict fare basis codes of all other fares in solution

Flights: S X_1 X_2 … … … … … X_m

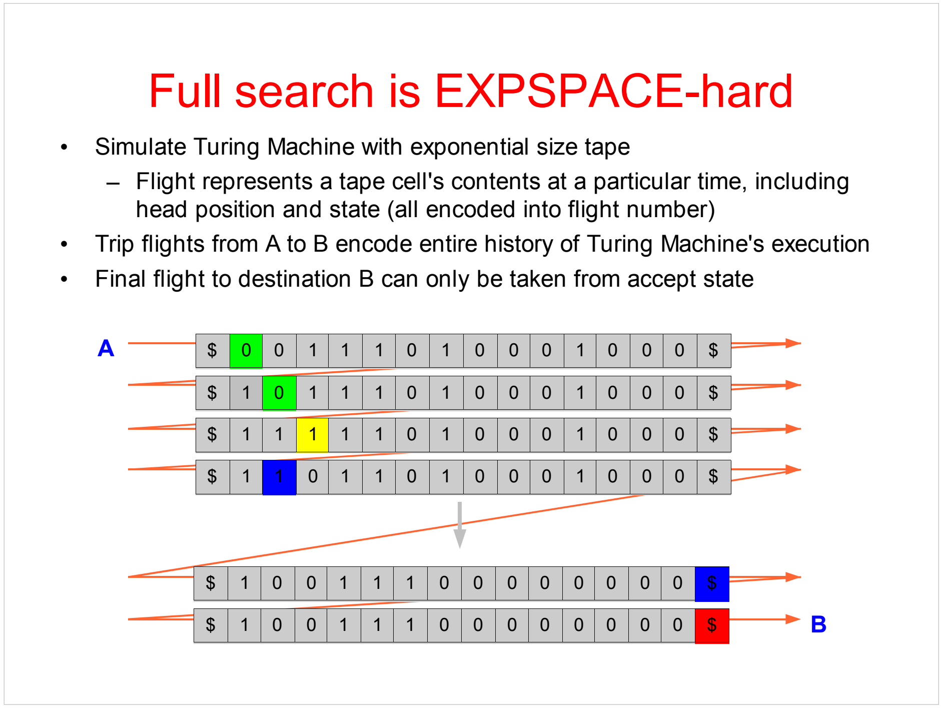
Fares: $S \rightarrow X_1$ RED1 BLUE1 GREEN1 | $X_1 \rightarrow X_2$ RED2 BLUE2 GREEN2 | … … | $X_{m-1} \rightarrow X_m$ REDm BLUEm GREENm

REDi rules: for all (i,j) in E, may not be used in solution with **REDj**

It is possible to encode in fare rules restrictions on the fare basis codes of other fares that appear on the same ticket, even if they are in other priceable units. Extra-priceable-unit restrictions on fare combinations are called **end-on-end fare combinability** restrictions, or **end-on-end** restrictions. To complete the translation of graph coloring into air travel pricing, if vertex i is connected to vertex j by an edge, then every vertex i fare prohibits the vertex j fare of the same color from appearing on the same ticket. For example, if vertex 3 is connected to vertex 6, then RED3 prohibits RED6, BLUE3 prohibits BLUE6 and GREEN3 prohibits GREEN6.

Again, the only difficulties to this proof consist of finding mechanisms in the airline industry's rule system sufficiently powerful to encode the original problem. The power of a fare to restrict, even independently, the fare basis codes of all other fares in a solution is simply too powerful for efficient search.

To the extent that one associates NP-hard (NP-complete) problems with exponential time search, this result is extremely important, because in many queries the base and exponent are sufficiently large for exhaustive search to be impossible. For example, for a round-trip query requiring three outbound and three return flights, it may be necessary to search over tickets that use 6 fares per passenger. For each market, there may be 1000 published fares. Finally, if multiple passengers travel there can be interactions between the passengers' fares. So, for a two person query the search space may be greater than 1000^{12}, or 10^{36}. For a completely fixed set of flights!!



Full search is EXPSPACE-hard

- Simulate Turing Machine with exponential size tape
  - Flight represents a tape cell's contents at a particular time, including head position and state (all encoded into flight number)
- Trip flights from A to B encode entire history of Turing Machine's execution
- Final flight to destination B can only be taken from accept state

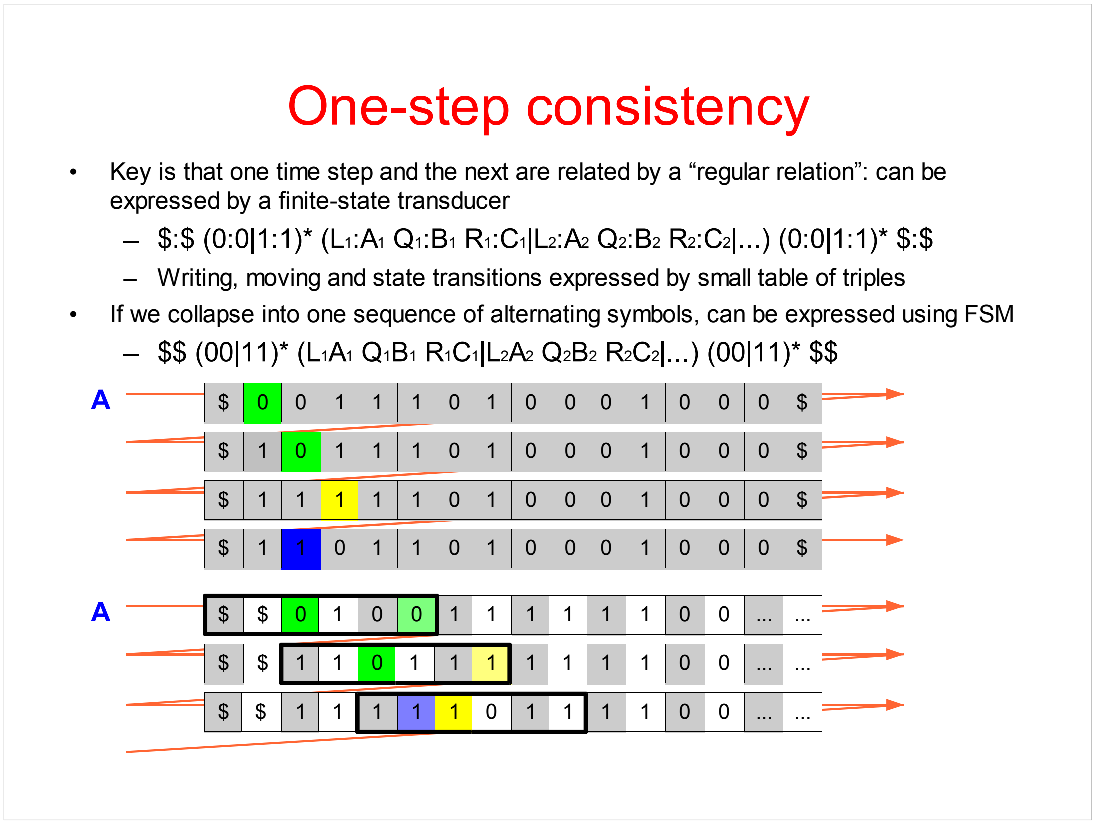
$	0	0	1	1	1	0	1	0	0	0	1	0	0	0	$
$	1	0	1	1	1	0	1	0	0	0	1	0	0	0	$
$	1	1	1	1	1	0	1	0	0	0	1	0	0	0	$
$	1	1	0	1	1	0	1	0	0	0	1	0	0	0	$
$	1	0	0	1	1	1	0	0	0	0	0	0	0	0	$
$	1	0	0	1	1	1	0	0	0	0	0	0	0	0	$

Both of the previous proof sketches assumed a bounded number of flights, related to the size of the input problem. If one allows that a traveler might take *any* finite number of flights to satisfy their query, the problem is much harder. It is not difficult to show the problem is at least EXPSPACE-hard using a proof similar to Cook's proof that SAT is NP-complete. The idea is, given a Turing Machine to simulate, to construct a query from A to B such that if there is an answer, flights of that answer encode the execution history of the Turing Machine tape from initial state to an accept state, and if the Turing Machine doesn't halt or accept the input, for there to be no valid solution to the travel query.

A network of flights is constructed such that each flight is covered by a single fare, with the combination representing the contents of one cell of a TM tape at a certain time, holding either 0, 1, $ (the tape end symbol) or the combination of 0, 1 or $ and a state symbol. The figure depicts a valid solution that reflects the execution of a TM. The solution is read from left to right, top to bottom, and each line represents the TM configuration at a particular time. Color is used to represent the TM state: the colored 0 in the second cell indicates that the TM starts in the green state with the read/write head over the second cell of the tape. Here red is used to indicate the accept state: when (and only when) the TM transitions to the red state, the flight graph permits a following flight to the destination B.

For further intuition, imagine that each flight is one day long and that the tape is of length 10, and that the trip starts on January 1st. Then the January 1st flight represents the initial contents of cell 1 ($). The January 24th flight represents the contents of cell 4 at time step 3 (1, state yellow). The flight number can be used to encode the cell contents (#1000 for $, #2000 for 0, #3000 for 1) and the airline the state (Tape Airlines if the head is in a different cell, or Green Airlines, Yellow Airlines, Blue Airlines, Red Airlines, etc). So the depicted solution would have flights TA1000, GA2000, TA2000, TA3000, etc). The next slide will complicate this representation slightly.

Copyright 2003 ITA Software



One-step consistency

- Key is that one time step and the next are related by a “regular relation”: can be expressed by a finite-state transducer
  - $\${:}\$\ (0{:}0|1{:}1)^*\ (L_1{:}A_1\ Q_1{:}B_1\ R_1{:}C_1|L_2{:}A_2\ Q_2{:}B_2\ R_2{:}C_2|\ldots)\ (0{:}0|1{:}1)^*\ \${:}\$$
  - Writing, moving and state transitions expressed by small table of triples
- If we collapse into one sequence of alternating symbols, can be expressed using FSM
  - $\$\$\ (00|11)^*\ (L_1A_1\ Q_1B_1\ R_1C_1|L_2A_2\ Q_2B_2\ R_2C_2|\ldots)\ (00|11)^*\ \$\$$

A	$	0	0	1	1	1	0	1	0	0	0	1	0	0	0	$
	$	1	0	1	1	1	0	1	0	0	0	1	0	0	0	$
	$	1	1	1	1	1	0	1	0	0	0	1	0	0	0	$
	$	1	1	0	1	1	0	1	0	0	0	1	0	0	0	$

A	$	$	0	1	0	0	1	1	1	1	1	1	0	0	...	...
	$	$	1	1	0	1	1	1	1	1	1	1	0	0	...	...
	$	$	1	1	1	1	1	0	1	1	1	1	0	0	...	...

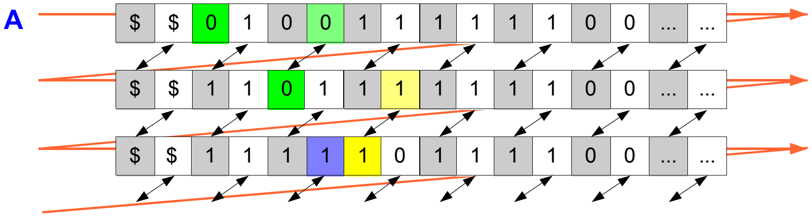
The important issue in the proof is how to enforce the standard logic for how a TM advances in time. The flight sequence must correctly simulate the TM.

Start by considering a single TM transition. The tape away from the head does not change. The cell the head is over may change contents, the state may change, and the head may move left or right one cell. Thus, all change takes place in a window of 3 cells centered on the head, and within these cells only a small finite number of before and after configurations are possible. Therefore the set of permitted transitions can be encoded using a *non-deterministic finite-state transducer* (FST) that uses (0:0|1:1)* to account for most of the tape and a small set of triples L:A Q:B R:C to encode the changes near the head. From the start of the diagram, we see that one triple is $:$ GREEN0:1 0:GREEN0.

We can change the representation so that the before-and-after contents of each cell are interleaved, as in the lower diagram. There the gray cells represent the tape at the current time step, and the white cells the tape at the next time step. Then a single row reflects one transition, and the consistency of that transition can be enforced by a *non-deterministic finite-state machine* (FSM): where the FST had X:Y, the FSM has XY. Therefore the initial flight sequence becomes TA1000 TA1000 GA2000 TA3000 TA2000 GA2000.

Multi-step consistency

- Now one-step transitions are encoded within a time step by FSM flight graph
- To ensure multi-step consistency, need to enforce equality between cells on a diagonal
  - Implemented using round-trip priceable-units that enforce same-flight-number restrictions on outbound flight and return flight
  - Key issue is ensuring that right cells are paired; implemented using minimum and maximum stay restrictions: min stay = max stay = TAPE-LENGTH * 2 - 1
  - EXP-SPACE limit comes from encoding of min/max stay: n bits encodes 2^n length



$	$	0	1	0	0	1	1	1	1	1	1	0	0	...	...
$	$	1	1	0	1	1	1	1	1	1	1	0	0	...	...
$	$	1	1	1	1	1	0	1	1	1	1	0	0	...	...

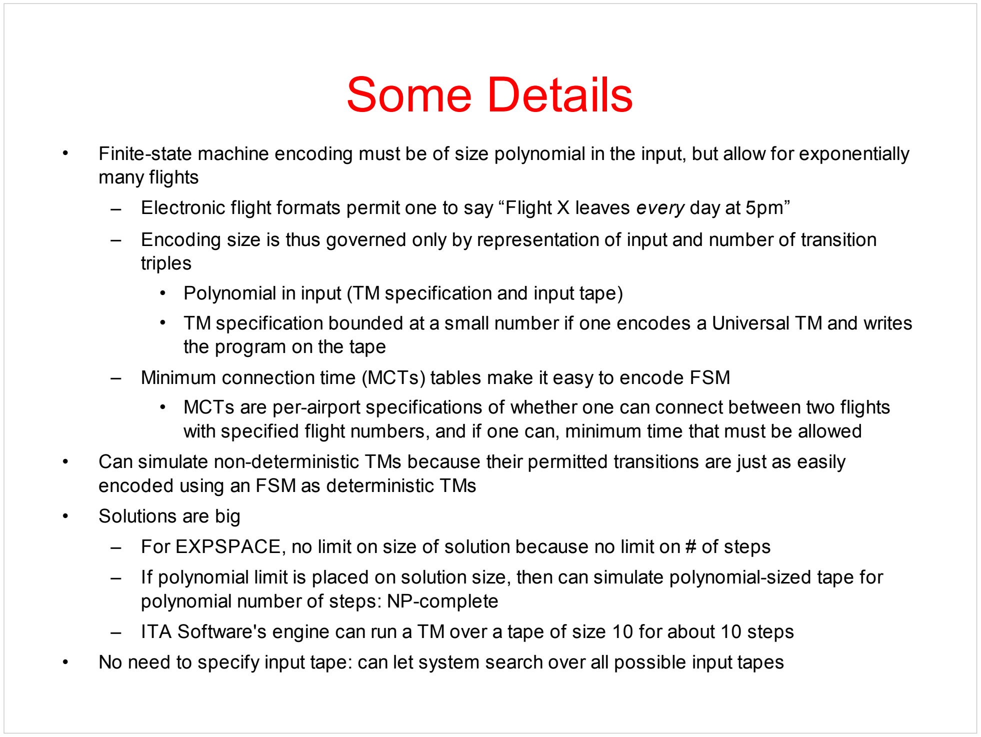
The advantage of encoding single-step transitions within a row is that multi-step consistency is ensured if the “after” portion of one row is identical to the “before” portion of the next, or in other words, if each white “after” cell is encoded using the same airline and flight number as the gray “before” cell that is one down and to the left of it. But this geometric relation is also a linear one: if the TM tape length is T, then each “before” flight must be the same as the “after” flight that occurs 2T-1 days later.

This flight equality restriction can be enforced using round-trip priceable units. The fare database is constructed so that each white cell (in a non-final row) can only be priced using a fare with rules that ensure it is paired in a round-trip priceable unit with another same-flight fare. The rules also have 2T-1 day minimum and maximum stay restrictions, ensuring that exactly the right white and gray cell are paired. Without this restriction the tape could become scrambled from one time step to the next.

These minimum and maximum stay restrictions are the limiting factor in the simulation. It takes n bits to encode a minimum stay of length 2^n, so this construction can “only” simulate a TM with a tape of length exponential in the length of the encoding of the TM. On the other hand, there is no limit to the number of steps the TM can run.

The flight and fare databases necessary for this query do not depend on how long execution takes, unlike Cook's SAT proof. This is because it is possible in the airlines' electronic flight distribution language to say that a flight leaves every day without specifying each departure separately.

For a fixed Turing Machine the only causes of variation in the size of the encoding are that some flights need to be dedicated to forcing the beginning of the first row of the simulation to match the TM input (thus, the flight network grows linearly with the size of the TM input) and the encoding of tape length using minimum and maximum stay restrictions on fares. As it is possible to encode a duration of 2^n using n bits, general travel planning is at least EXPSPACE-hard.



Some Details

- Finite-state machine encoding must be of size polynomial in the input, but allow for exponentially many flights
  - Electronic flight formats permit one to say "Flight X leaves *every* day at 5pm"
  - Encoding size is thus governed only by representation of input and number of transition triples
    - Polynomial in input (TM specification and input tape)
    - TM specification bounded at a small number if one encodes a Universal TM and writes the program on the tape
  - Minimum connection time (MCTs) tables make it easy to encode FSM
    - MCTs are per-airport specifications of whether one can connect between two flights with specified flight numbers, and if one can, minimum time that must be allowed
- Can simulate non-deterministic TMs because their permitted transitions are just as easily encoded using an FSM as deterministic TMs
- Solutions are big
  - For EXPSPACE, no limit on size of solution because no limit on # of steps
  - If polynomial limit is placed on solution size, then can simulate polynomial-sized tape for polynomial number of steps: NP-complete
  - ITA Software's engine can run a TM over a tape of size 10 for about 10 steps
- No need to specify input tape: can let system search over all possible input tapes

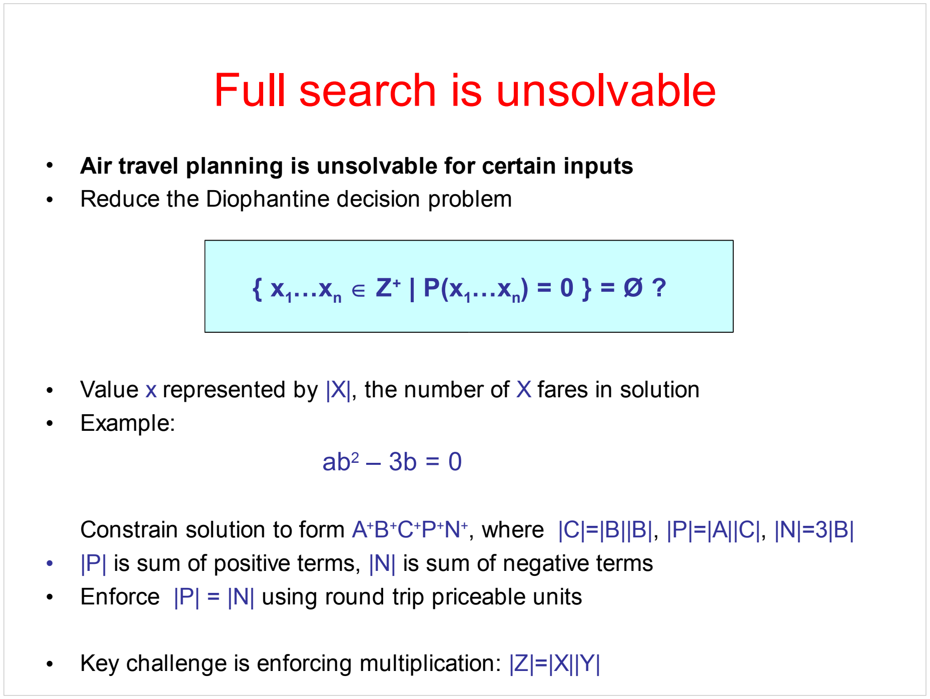
The proof goes through equally well for non-deterministic TMs. This means that even if solution size is bounded to a polynomial of the length of the TM input, the problem is NP-hard (in fact, NP-complete).

As will be understood by those familiar with the undecidability proofs of Post's Correspondence Problem and CFG intersection, the source of complexity here is the combination of finite state constraints (expressible in many ways: in this proof with the flight network) and long-distance "equality" checks (expressed using round-trip priceable units). Both of these are fundamental to the airline industry, though the trips constructed in this proof are very artificial.

An interesting question is whether there is a way to prevent tape permutations in some way other than with minimum and maximum stay restrictions, since this is the limiting factor in the proof's complexity bound. If one could ensure the same geometric equality conditions no matter how long the tape was, then the air travel planning could simulate a TM for an unbounded number of steps on tapes of unbounded length: the problem of finding a valid ticket for a trip would be unsolvable (undecidable) in the general case. For such a proof one needs a way to restrict the ways that priceable units can be laid out in an answer. The airlines do provide one such mechanism, called the **back to back** restriction. It is not nearly so fundamental to the industry, and not easy to coerce to a form useful for this proof, but it can be used by slightly restructuring the layout of priceable units in solutions, as is possible using a proof based on Diophantine equations. Unfortunately the details of that proof are considerably more complex and add little to the understanding of the problem.

The general travel planning problem is unsolvable, meaning that no computer, no matter how long it spends, can find an answer to every travel query (or determine that none exists) for every database of flights and fares that the airlines can publish.

The proof here is based on the Diophantine decision problem, and is substantially more complicated than the previous ones presented. It is not possible to explain all here, and this sketch omits the most difficult steps, but the general idea can be conveyed.



Full search is unsolvable

- **Air travel planning is unsolvable for certain inputs**
- Reduce the Diophantine decision problem

$$\{ x_1 \ldots x_n \in \mathbb{Z}^+ \mid P(x_1 \ldots x_n) = 0 \} = \varnothing \, ?$$

- Value x represented by $|X|$, the number of X fares in solution
- Example:

$$ab^2 - 3b = 0$$

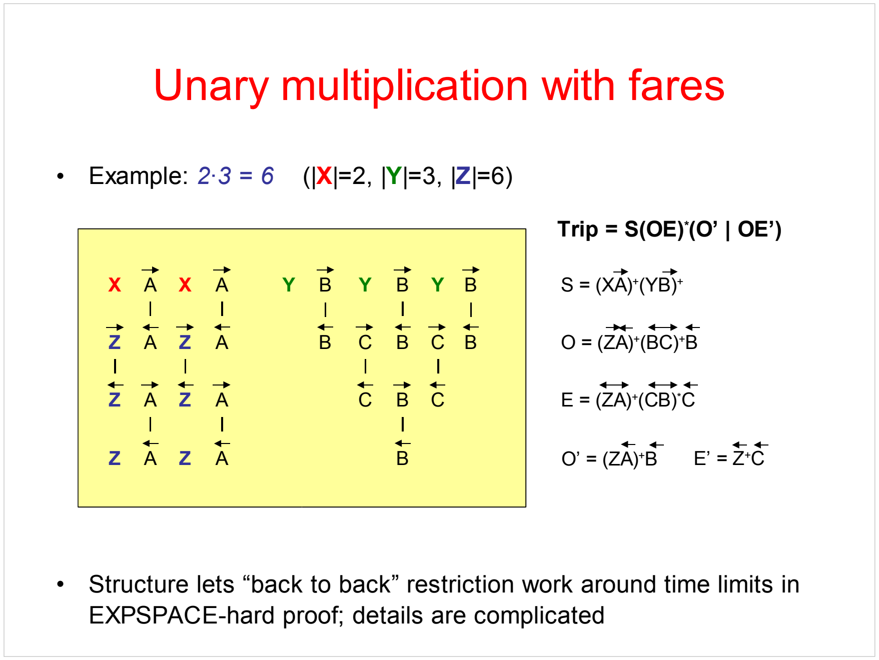
Constrain solution to form $A^+B^+C^+P^+N^+$, where $|C|=|B||B|$, $|P|=|A||C|$, $|N|=3|B|$

- $|P|$ is sum of positive terms, $|N|$ is sum of negative terms
- Enforce $|P| = |N|$ using round trip priceable units
- Key challenge is enforcing multiplication: $|Z|=|X||Y|$

The (unsolvable) Diophantine decision problem, also known as Hilbert's 10th problem, is that of determining whether a polynomial with integer coefficients has positive integer roots. This can be translated into a travel problem wherein the equation has roots if and only if there is a solution to the travel problem. If there is a solution, the number of fares in the solution in each of various different classes matches the roots of the polynomial. In general the count of fares in a class represents the numerical value of a variable or expression: the reduction does unary arithmetic with fares. The flight network is used to ensure that at least one fare in each input variable fare class must be used. The sum of all positive terms will be represented by the number of fares in class P, and the sum of all negative terms by the number of fares in class N. By giving P and N fares rules that force them to combine with each other in two-fare-component PUs, it is guaranteed that a solution only exists if the equation sums to zero.

Since addition can be expressed in the reduction definition (by expanding the class of fares that represents an expression), the key issue in this proof is whether it is possible to express multiplicative constraints on fare counts, such that any valid solution must have number of fares of type Z precisely equal to the number of fares of type X times the number of fares of type Y.

Multiplicative constraints (|Z| = |X||Y|) can be enforced for solutions of arbitrary size, though a full proof is substantial. Using unary multiplication, one can set up a geometric structure very similar to that used in the EXPSPACE proof. This time instead of copying forward the tape of a TM, priceable units are used to copy forward the number of fares of type X and the number of fares of type Y. The pattern of flights and fares is used to ensure that as time progresses, the number of unfinished priceable units involving B fares (a helper fare) steadily reduces, so that the number of steps (lines) in the construction is equal to the number of fares of type Y. During each time step, another helper fare (A) is used to ensure that a Z fare appears for every X. The end result is that the number of Z fares is precisely equal to |X||Y|.



Unary multiplication with fares

- Example: *2·3 = 6* (|X|=2, |Y|=3, |Z|=6)

Trip = S(OE)*(O’ | OE’)

S = $(X\overrightarrow{A})^+(Y\overrightarrow{B})^+$

O = $(\overrightarrow{Z}\overleftarrow{A})^+(\overleftrightarrow{BC})^+\overleftarrow{B}$

E = $(\overleftrightarrow{ZA})^+(\overleftrightarrow{CB})^*\overleftarrow{C}$

O’ = $(Z\overleftarrow{A})^+\overleftarrow{B}$ E’ = $\overleftarrow{Z}^+\overleftarrow{C}$

- Structure lets “back to back” restriction work around time limits in EXPSPACE-hard proof; details are complicated

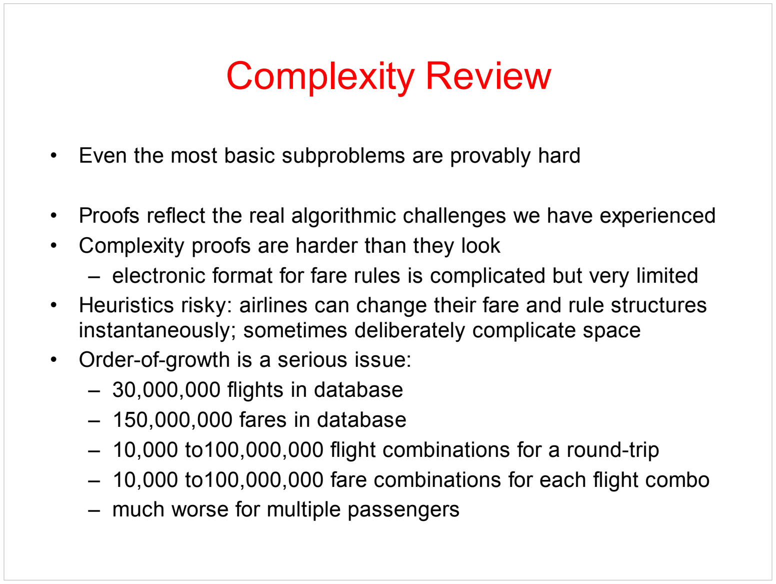
This is not a complete sketch: the difficult step is ensuring that exactly |X| fares of type Z appear in every row, a very similar problem to the tape permutation problem. However with suitable complications to this basic structure some details permit the airlines’ “back to back” restriction to be used to make the proof go through no matter how many fares are in the solution. The back to back restriction is a rule the airlines can selectively enforce that limits the manner in which priceable-units on the same airline can be embedded. It is designed to prevent people from circumventing Saturday night stay restrictions for a round-trip A to B, B to A (with insufficient layover in B) by buying a “double” ticket A to B (short stay) B to A (long stay) A to B (short stay) B to A, priced with the first A to B paired with the last B to A and the first B to A paired with the second A to B. (The price of two round-trip tickets built from cheap round-trip Saturday night stay fares is often much less than one built from expensive unrestricted fares.)

This unsolvability result is amusing, but doesn’t offer any greater insight into why the travel planning problem is hard than the EXPSPACE proof. Any system that permits long-distance constraints sufficient to copy arbitrary amounts of data forward (as PUs do) is liable to be undecidable.

It is important to understand that multiplication is extremely powerful, but only if the circuit is bi-directional, so that it can search for factors. It is easy to evaluate a polynomial. It is another matter entirely to be able to search for the roots of one. This multiplication circuit is really a multiplicative constraint on solutions, and a search engine that can find solutions satisfying the constraint can be used to multiply, divide, factor and find roots.

What do these proofs tell us? The first two proofs show that even if all but one dimension of the problem are fixed, the problem remains at least NP-hard. Each dimension of the search is hard by itself. And the proofs themselves reflect fairly well the algorithmic difficulties in solving the problem, especially the proof that even if flights are fixed, the search for fare combinations is NP-hard, since the types of restrictions used in that proof are quite commonly encoded by the airlines. The proofs are slightly more difficult than they look here, only because it is difficult to find the machinery in the airline industry’s complicated but inexpressive rule language to express the constraints necessary for reductions.

The proofs that these constrained problems are NP-hard really should read NP-complete, but to prove completeness one would have to show that there is a polynomial time evaluator, and the problem specification is way too complex to ever prove such a thing. To the best of my knowledge, polynomial time evaluation is possible.



Complexity Review

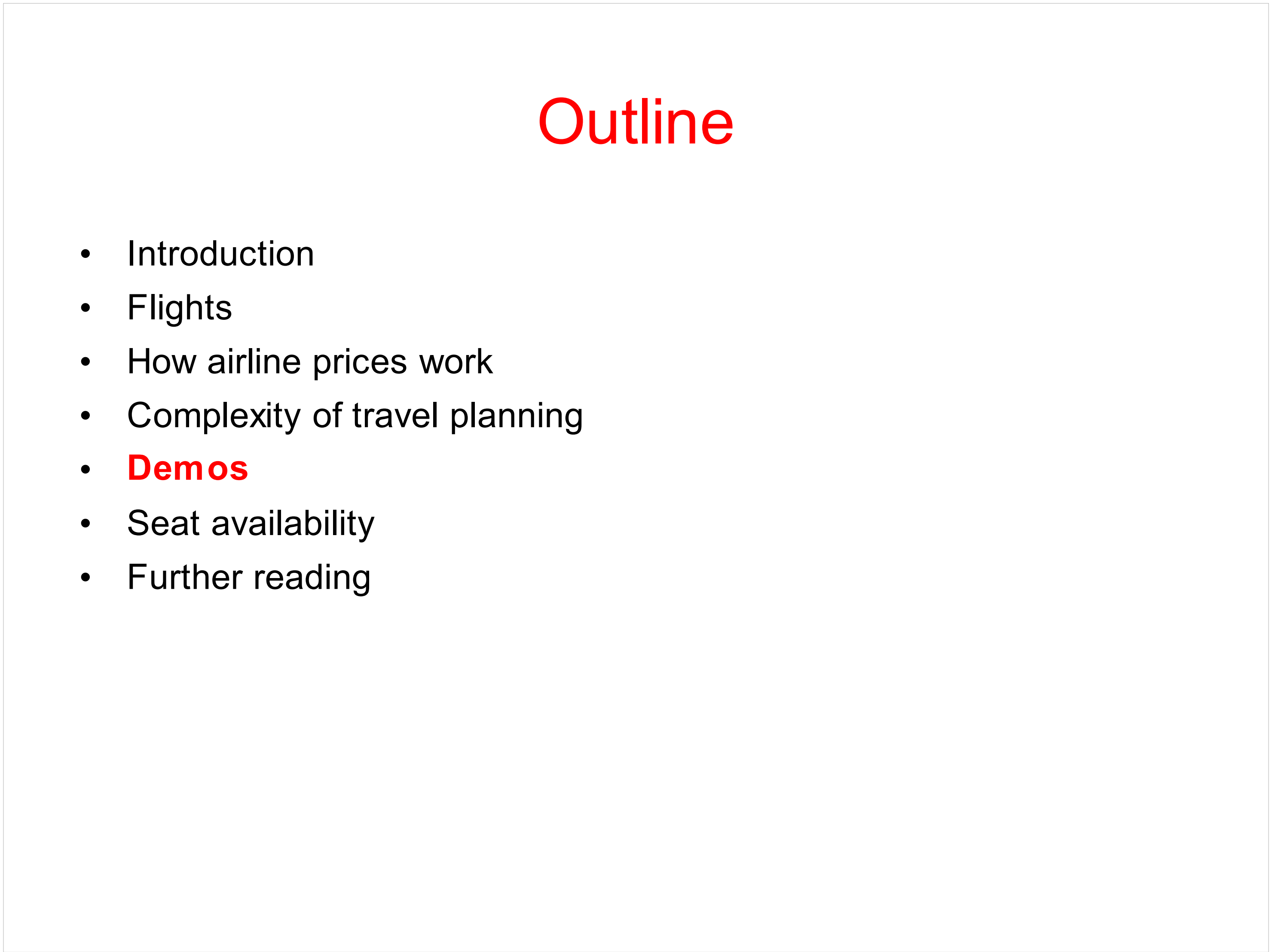
- Even the most basic subproblems are provably hard

- Proofs reflect the real algorithmic challenges we have experienced
- Complexity proofs are harder than they look
  - electronic format for fare rules is complicated but very limited
- Heuristics risky: airlines can change their fare and rule structures instantaneously; sometimes deliberately complicate space
- Order-of-growth is a serious issue:
  - 30,000,000 flights in database
  - 150,000,000 fares in database
  - 10,000 to100,000,000 flight combinations for a round-trip
  - 10,000 to100,000,000 fare combinations for each flight combo
  - much worse for multiple passengers

The EXPSPACE and unsolvability proofs are harder to interpret. They depend on very unusual constructions and lengthy tickets. They should perhaps be seen as supporting evidence for the power of the airlines' pricing system, that reinforces the simpler results. The fact that they depend only on very simple rule systems also suggests that complexity can arise from the combination of independently simple pricing rules.

A normal response to a problem that is theoretically hard is to search for heuristics that perform well in practice. One of the challenges to the travel planning problem is that the airlines can update their fares and rules 10 times a day, potentially changing the structure of the search problem in an instant: any carefully tuned system could be destroyed by a new vice president of marketing at an airline on his or her first day of work.

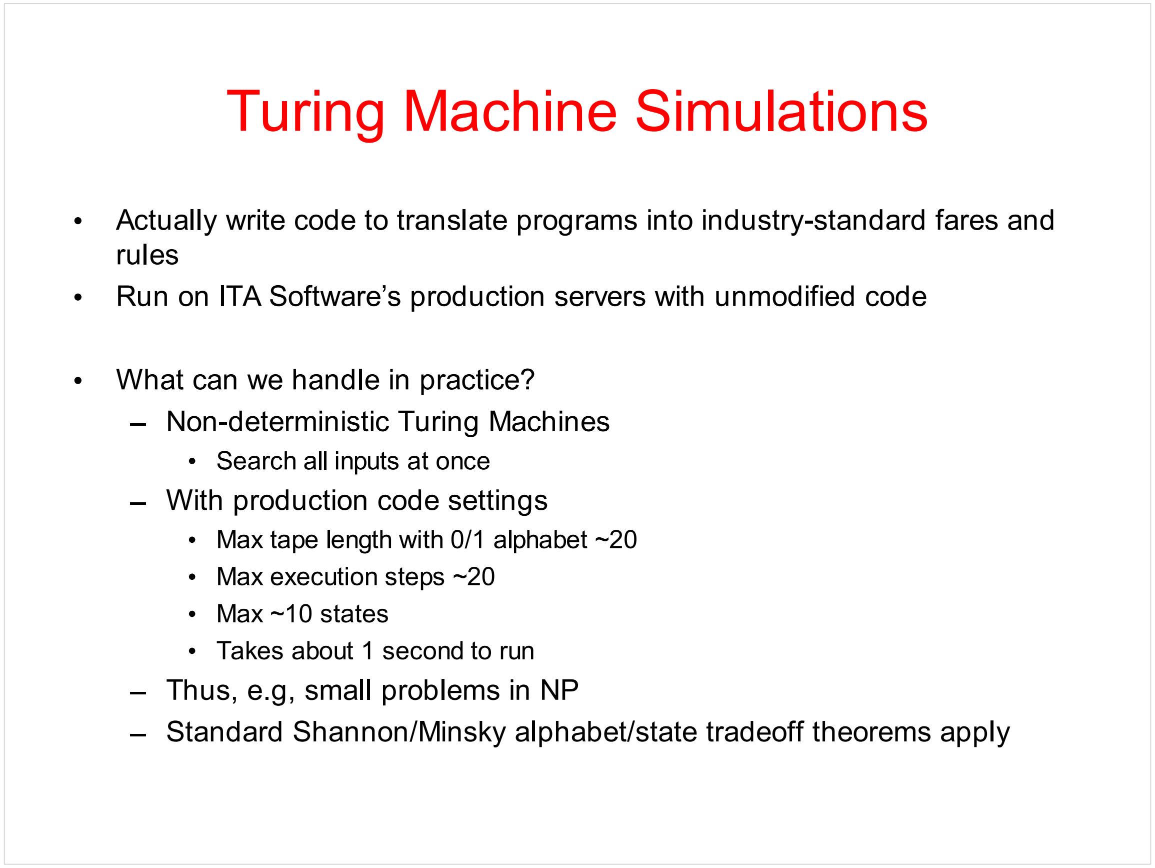
One might hope that problem sizes are small enough in practice to be solvable, but the N’s in algorithmic complexity can be big. There are 30 million flights a year, 150 million published fares, 10,000 to 100,000,000 or more flight combinations for a simple round trip journey, 10,000 to 100,000,000 or 100,000,000,000,000 or more fare combinations for a fixed set of flights, and exponentially worse for multiple passengers.



Outline

- Introduction
- Flights
- How airline prices work
- Complexity of travel planning
- **Demos**
- Seat availability
- Further reading

As a demonstration of the techniques used in the preceding proofs, ITA Software has written some tools to translate programs and circuits into industry-standard databases, and run ITA Software's stock search engine on them. All solutions shown are the result of running the search engine as normal, but on this specially constructed input.



Turing Machine Simulations

- Actually write code to translate programs into industry-standard fares and rules
- Run on ITA Software’s production servers with unmodified code

- What can we handle in practice?
  - Non-deterministic Turing Machines
    - Search all inputs at once
  - With production code settings
    - Max tape length with 0/1 alphabet ~20
    - Max execution steps ~20
    - Max ~10 states
    - Takes about 1 second to run
  - Thus, e.g, small problems in NP
  - Standard Shannon/Minsky alphabet/state tradeoff theorems apply

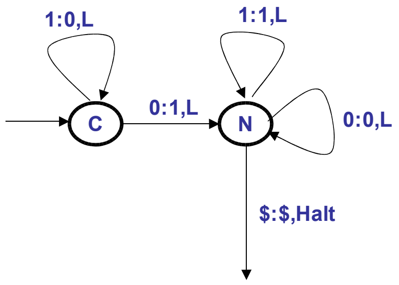
Turing Machines can be encoded as in the EXPSPACE proof, but that doesn't mean a travel planning search engine will be able to run them! Every search engine has limitations, and given the computational complexity of the planning problem it is difficult to imagine very large simulations succeeding. In the case of ITA Software's engine as of early 2003, it is possible to simulate TMs with small numbers of states for between 10 and 20 steps on tapes of length from 10 to 20 (depending on the specific TM and tape alphabet). Fundamentally, ITA Software's engine will not generally duplicate airports for a user-requested portion of a trip, which limits the size of the tape and number of time steps to a polynomial in the size of the databases. The net effect is that ITA Software's engine can execute non-deterministic TMs over small tapes for small numbers of steps, or in other words, that it can solve small instances of NP problems. As one would expect.

After the databases have been constructed, the machine is run by posing a query with one trip segment per time step, between made-up airports. For example, to run for 3 time steps one poses a query “find solutions from XXA to XXB, then from XXB to XXC, then from XXC back to XXA”. The flights in each trip segment encode the tape at that time step as well as the transition to the next time step.

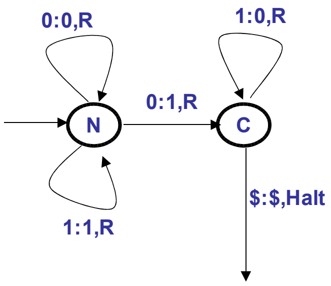
Right to Left Increment

- Right-to-left boolean increment-by-1 machine is 2-state DTM (DFST)
- Left-to-right boolean increment-by-1 machine is 2-state NTM (NFST)
- Set prices of “1” fares to reflect bit position
  - 1.00$*$2^i$ input tape, 0.01$*$2^i$ output tape

Deterministic R-to-L



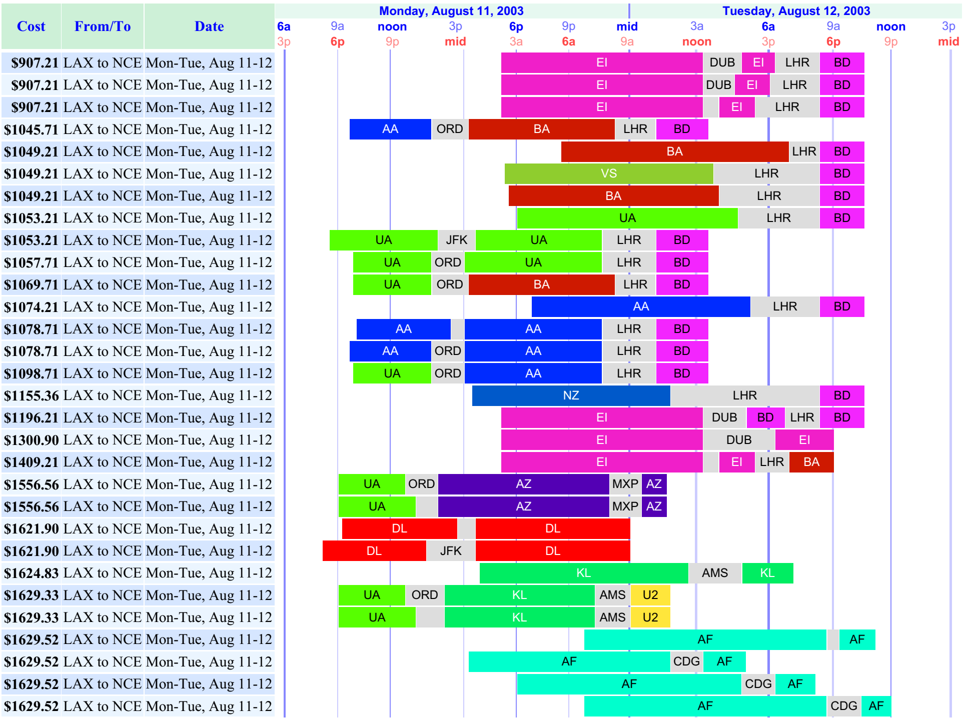
Non-Deterministic L-to-R



As a first example, a left-to-right binary increment program is encoded. Incrementing a binary number from right (least significant bit) to left (most significant bit) is a simple deterministic operation that only involves moving the head to the left and maintaining the carry bit in the state. Working from left to right essentially reverses the machine's transitions, which requires non-determinism. The machine must guess whether the remaining low bits will carry to know whether to flip the bit under the head.

To simplify interpretation of results, the encoding of the TM assigns values to fares in such a way that the input and output values represented in binary appear as the dollar and cents portion of the trip cost. This is done by giving a price to the “1” fares used for the first segment of the trip a value in dollars proportional to 2^i where i is the distance from the right end of the tape, and similarly in cents to the “1” fares used for the last segment of the trip. Thus, solutions should have values $0.01, $1.02, $2.03, et cetera.

Graphical Presentation



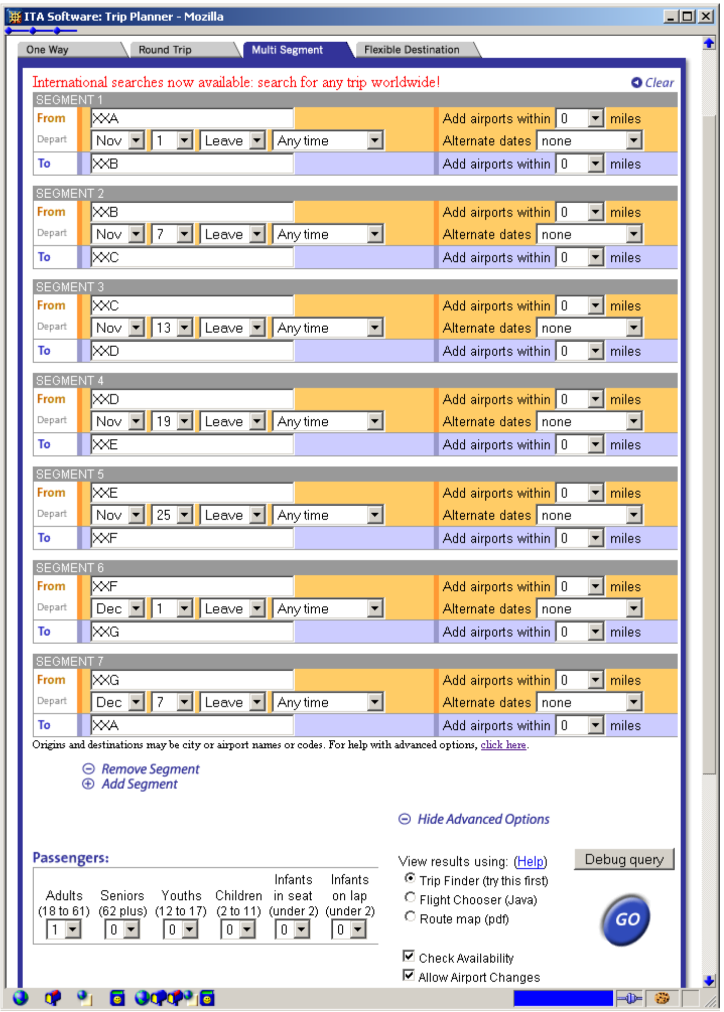
Cost	From/To	Date	
$907.21	LAX to NCE	Mon-Tue, Aug 11-12	EI DUB EI LHR BD
$907.21	LAX to NCE	Mon-Tue, Aug 11-12	EI DUB EI LHR BD
$907.21	LAX to NCE	Mon-Tue, Aug 11-12	EI EI LHR BD
$1045.71	LAX to NCE	Mon-Tue, Aug 11-12	AA ORD BA LHR BD
$1049.21	LAX to NCE	Mon-Tue, Aug 11-12	BA LHR BD
$1049.21	LAX to NCE	Mon-Tue, Aug 11-12	VS LHR BD
$1049.21	LAX to NCE	Mon-Tue, Aug 11-12	BA LHR BD
$1053.21	LAX to NCE	Mon-Tue, Aug 11-12	UA LHR BD
$1053.21	LAX to NCE	Mon-Tue, Aug 11-12	UA JFK UA LHR BD
$1057.71	LAX to NCE	Mon-Tue, Aug 11-12	UA ORD UA LHR BD
$1069.71	LAX to NCE	Mon-Tue, Aug 11-12	UA ORD BA LHR BD
$1074.21	LAX to NCE	Mon-Tue, Aug 11-12	AA LHR BD
$1078.71	LAX to NCE	Mon-Tue, Aug 11-12	AA AA LHR BD
$1078.71	LAX to NCE	Mon-Tue, Aug 11-12	AA ORD AA LHR BD
$1098.71	LAX to NCE	Mon-Tue, Aug 11-12	UA ORD AA LHR BD
$1155.36	LAX to NCE	Mon-Tue, Aug 11-12	NZ LHR BD
$1196.21	LAX to NCE	Mon-Tue, Aug 11-12	EI DUB BD LHR BD
$1300.90	LAX to NCE	Mon-Tue, Aug 11-12	EI DUB EI
$1409.21	LAX to NCE	Mon-Tue, Aug 11-12	EI EI LHR BA
$1556.56	LAX to NCE	Mon-Tue, Aug 11-12	UA ORD AZ MXP AZ
$1556.56	LAX to NCE	Mon-Tue, Aug 11-12	UA AZ MXP AZ
$1621.90	LAX to NCE	Mon-Tue, Aug 11-12	DL DL
$1621.90	LAX to NCE	Mon-Tue, Aug 11-12	DL JFK DL
$1624.83	LAX to NCE	Mon-Tue, Aug 11-12	KL AMS KL
$1629.33	LAX to NCE	Mon-Tue, Aug 11-12	UA ORD KL AMS U2
$1629.33	LAX to NCE	Mon-Tue, Aug 11-12	UA KL AMS U2
$1629.52	LAX to NCE	Mon-Tue, Aug 11-12	AF AF
$1629.52	LAX to NCE	Mon-Tue, Aug 11-12	AF CDG AF
$1629.52	LAX to NCE	Mon-Tue, Aug 11-12	AF CDG AF
$1629.52	LAX to NCE	Mon-Tue, Aug 11-12	AF CDG AF

To help understand the upcoming slides, this figure shows a graphical presentation of a small set of solutions to a one-way Los Angeles (LAX) to Nice (NCE) query. Each row represents one possible itinerary from LAX to NCE. Time is laid out horizontally, with colored bars representing flights. A flight's airline determines the color of its bar. Layovers are represented by gray bars. When there is space airline and airport codes are written. This general format will be used to present solutions to queries, though in subsequent slides each trip will have several parts and thus be represented by several lines.

In the TM encoding used for these demos, a tape cell's contents, and the TM state if the head is over the cell, are represented by the airline of a flight. Thus, the configuration of the tape can be read from the sequence of bar colors.

Copyright 2003 ITA Software

Query



International searches now available: search for any trip worldwide!

SEGMENT 1						
From	XXA				Add airports within 0 miles	
Depart	Nov	1	Leave	Any time	Alternate dates none	
To	XXB				Add airports within 0 miles	
SEGMENT 2						
From	XXB				Add airports within 0 miles	
Depart	Nov	7	Leave	Any time	Alternate dates none	
To	XXC				Add airports within 0 miles	
SEGMENT 3						
From	XXC				Add airports within 0 miles	
Depart	Nov	13	Leave	Any time	Alternate dates none	
To	XXD				Add airports within 0 miles	
SEGMENT 4						
From	XXD				Add airports within 0 miles	
Depart	Nov	19	Leave	Any time	Alternate dates none	
To	XXE				Add airports within 0 miles	
SEGMENT 5						
From	XXE				Add airports within 0 miles	
Depart	Nov	25	Leave	Any time	Alternate dates none	
To	XXF				Add airports within 0 miles	
SEGMENT 6						
From	XXF				Add airports within 0 miles	
Depart	Dec	1	Leave	Any time	Alternate dates none	
To	XXG				Add airports within 0 miles	
SEGMENT 7						
From	XXG				Add airports within 0 miles	
Depart	Dec	7	Leave	Any time	Alternate dates none	
To	XXA				Add airports within 0 miles	

Origins and destinations may be city or airport names or codes. For help with advanced options, click here.

⊖ Remove Segment
⊕ Add Segment

⊖ Hide Advanced Options

Passengers:

Adults (18 to 61)	Seniors (62 plus)	Youths (12 to 17)	Children (2 to 11)	Infants in seat (under 2)	Infants on lap (under 2)
1	0	0	0	0	0

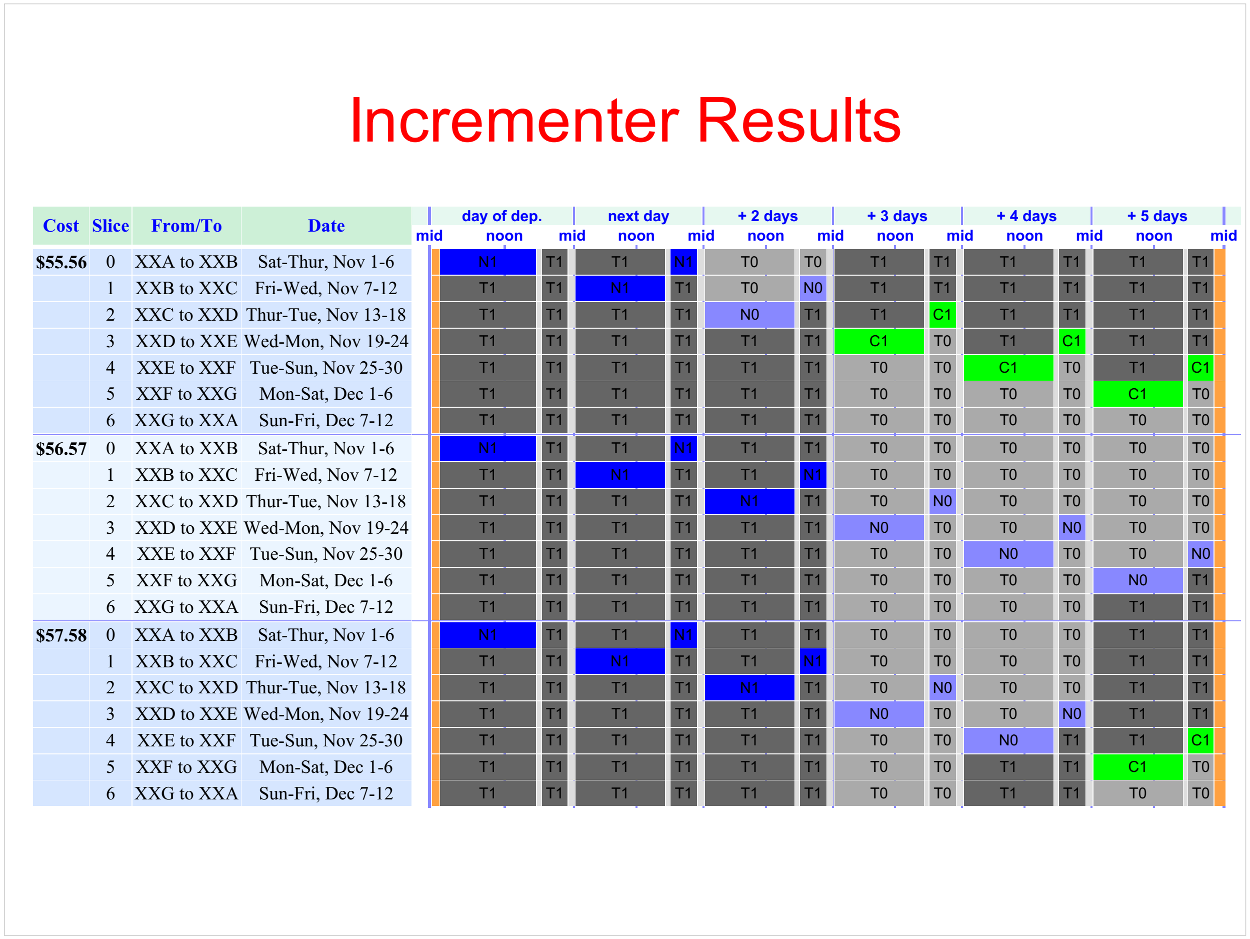
View results using: (Help)
● Trip Finder (try this first)
○ Flight Chooser (Java)
○ Route map (pdf)

☑ Check Availability
☑ Allow Airport Changes

The query is posed...

Copyright 2003 ITA Software

Notes for MIT course 6.034, Fall, 2003



Incrementer Results

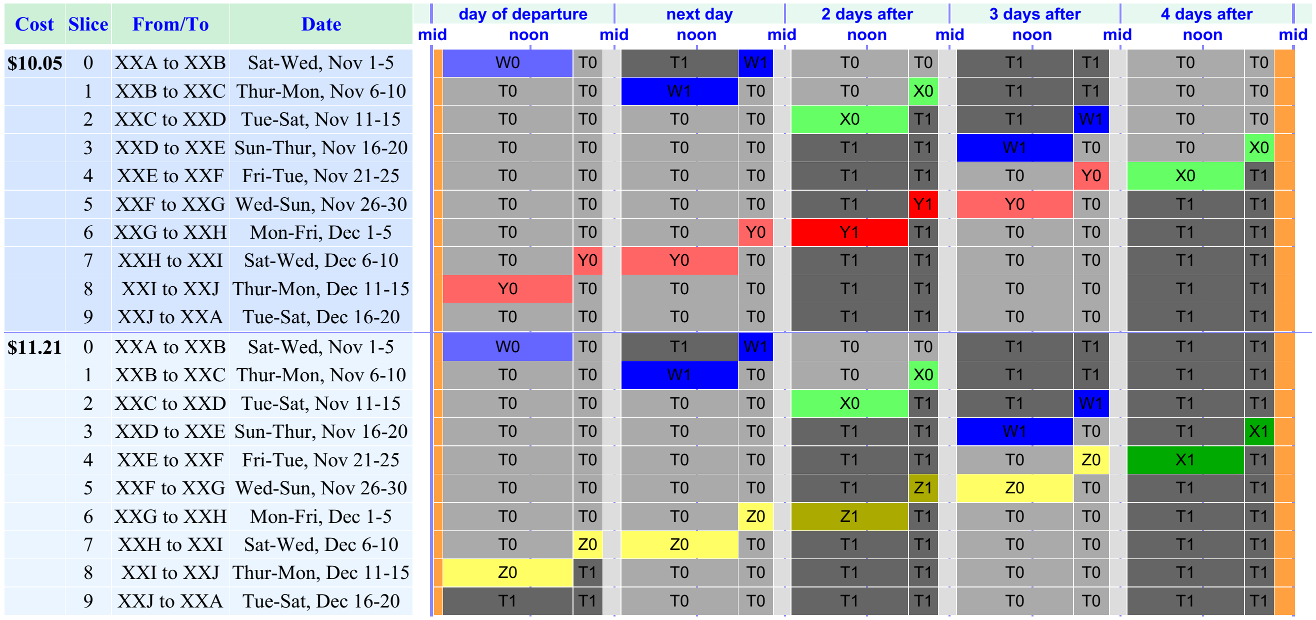
Cost	Slice	From/To	Date	day of dep.		next day		+ 2 days		+ 3 days		+ 4 days		+ 5 days	
				mid noon	mid	noon	mid	noon	mid	noon	mid	noon	mid	noon	mid
$55.56	0	XXA to XXB	Sat-Thur, Nov 1-6	N1	T1	T1	N1	T0	T0	T1	T1	T1	T1	T1	T1
	1	XXB to XXC	Fri-Wed, Nov 7-12	T1	T1	N1	T1	T0	N0	T1	T1	T1	T1	T1	T1
	2	XXC to XXD	Thur-Tue, Nov 13-18	T1	T1	T1	T1	N0	T1	T1	C1	T1	T1	T1	T1
	3	XXD to XXE	Wed-Mon, Nov 19-24	T1	T1	T1	T1	T1	T1	C1	T0	T1	C1	T1	T1
	4	XXE to XXF	Tue-Sun, Nov 25-30	T1	T1	T1	T1	T1	T1	T0	T0	C1	T0	T1	C1
	5	XXF to XXG	Mon-Sat, Dec 1-6	T1	T1	T1	T1	T1	T1	T0	T0	T0	T0	C1	T0
	6	XXG to XXA	Sun-Fri, Dec 7-12	T1	T1	T1	T1	T1	T1	T0	T0	T0	T0	T0	T0
$56.57	0	XXA to XXB	Sat-Thur, Nov 1-6	N1	T1	T1	N1	T1	T1	T0	T0	T0	T0	T0	T0
	1	XXB to XXC	Fri-Wed, Nov 7-12	T1	T1	N1	T1	T1	N1	T0	T0	T0	T0	T0	T0
	2	XXC to XXD	Thur-Tue, Nov 13-18	T1	T1	T1	T1	N1	T1	T0	N0	T0	T0	T0	T0
	3	XXD to XXE	Wed-Mon, Nov 19-24	T1	T1	T1	T1	T1	T1	N0	T0	T0	N0	T0	T0
	4	XXE to XXF	Tue-Sun, Nov 25-30	T1	T1	T1	T1	T1	T1	T0	T0	N0	T0	T0	N0
	5	XXF to XXG	Mon-Sat, Dec 1-6	T1	T1	T1	T1	T1	T1	T0	T0	T0	T0	N0	T1
	6	XXG to XXA	Sun-Fri, Dec 7-12	T1	T1	T1	T1	T1	T1	T0	T0	T0	T0	T1	T1
$57.58	0	XXA to XXB	Sat-Thur, Nov 1-6	N1	T1	T1	N1	T1	T1	T0	T0	T0	T0	T1	T1
	1	XXB to XXC	Fri-Wed, Nov 7-12	T1	T1	N1	T1	T1	N1	T0	T0	T0	T0	T1	T1
	2	XXC to XXD	Thur-Tue, Nov 13-18	T1	T1	T1	T1	N1	T1	T0	N0	T0	T0	T1	T1
	3	XXD to XXE	Wed-Mon, Nov 19-24	T1	T1	T1	T1	T1	T1	N0	T0	T0	N0	T1	T1
	4	XXE to XXF	Tue-Sun, Nov 25-30	T1	T1	T1	T1	T1	T1	T0	T0	N0	T1	T1	C1
	5	XXF to XXG	Mon-Sat, Dec 1-6	T1	T1	T1	T1	T1	T1	T0	T0	T1	T1	C1	T0
	6	XXG to XXA	Sun-Fri, Dec 7-12	T1	T1	T1	T1	T1	T1	T0	T0	T1	T1	T0	T0

This slide graphically depicts three of the solutions the search engine found running on a 6-cell tape. Dark gray flights are 1s, light gray are 0s. The labels in cells are the airline (T0 Airlines represents a cell with a 0 in it, T1 Airlines represents a cell with a 1 in it, N0 indicates a cell with a zero that the head is over, in state N, et cetera). Colored cells indicate the tape head, the green state for “carry” and the blue for “no carry”. Orange flights at start and end delimit the tape. Long morning flights encode the current tape, short evening flights the tape at the next time step. Thus, short flights are identically colored to the long flight diagonally below and to the left – 6 days later. When the machine enters a halt state the state disappears from the tape; hence the last row of each solution has no colored state cells. Notice the non-determinism of the machine: in some cases when in the blue no-carry state over a 0 cell, it increments the cell to 1, in some cases it does not: it guesses the next carry (whether the remaining cells to the right are all ones). No solutions result from wrong guesses, because every guess gets verified as the machine scans further right.

Note the complexity of these trips: 98 flights, 98 fares, 36 round-trip PUs, 26 one-way PUs!

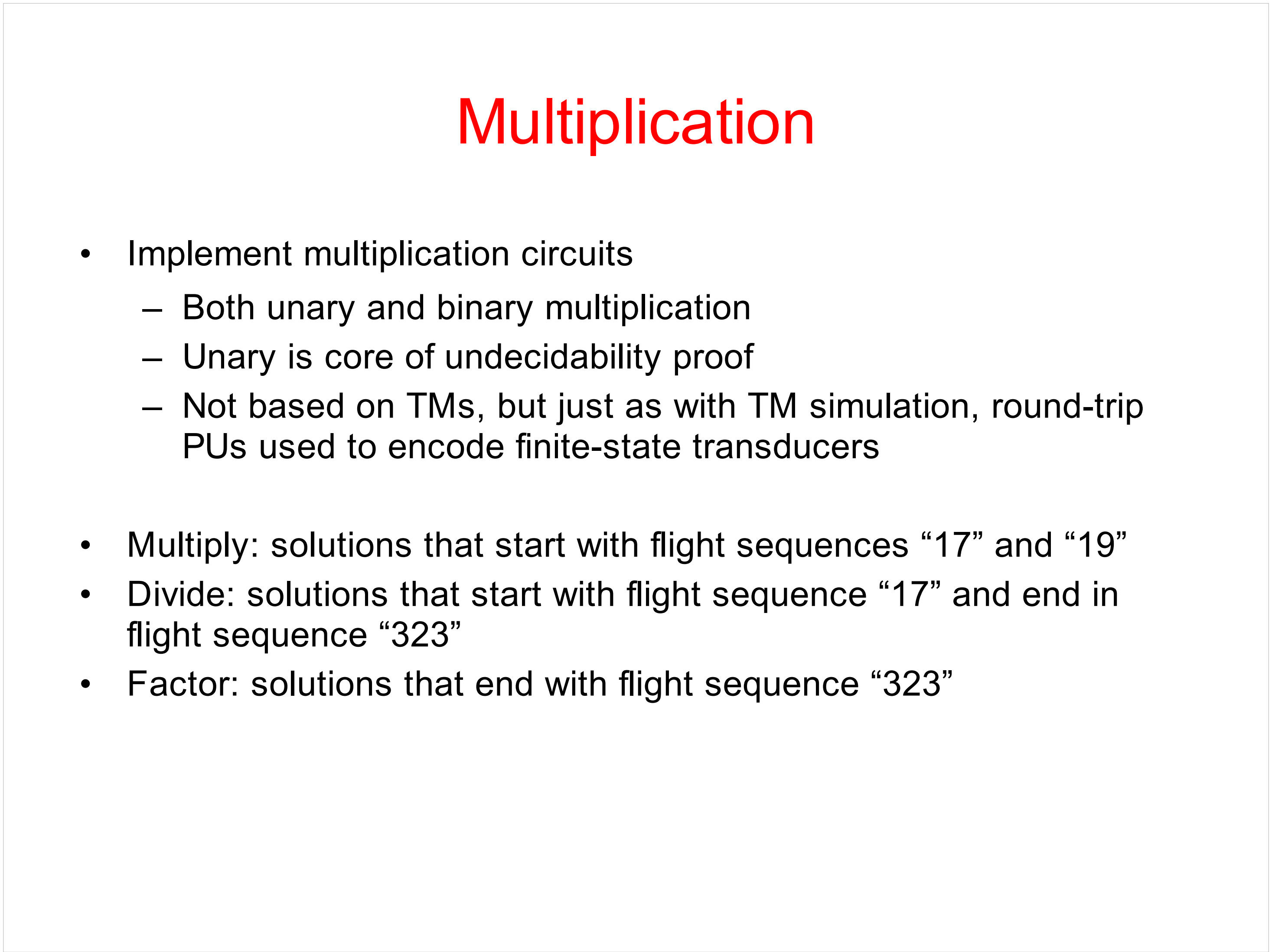
Copyright 2003 ITA Software

Bit rotatation



Cost	Slice	From/To	Date	day of departure		next day		2 days after		3 days after		4 days after	
				mid noon	mid	noon	mid	noon	mid	noon	mid	noon	mid
$10.05	0	XXA to XXB	Sat-Wed, Nov 1-5	W0	T0	T1	W1	T0	T0	T1	T1	T0	T0
	1	XXB to XXC	Thur-Mon, Nov 6-10	T0	T0	W1	T0	T0	X0	T1	T1	T0	T0
	2	XXC to XXD	Tue-Sat, Nov 11-15	T0	T0	T0	T0	X0	T1	T1	W1	T0	T0
	3	XXD to XXE	Sun-Thur, Nov 16-20	T0	T0	T0	T0	T1	T1	W1	T0	T0	X0
	4	XXE to XXF	Fri-Tue, Nov 21-25	T0	T0	T0	T0	T1	T1	T0	Y0	X0	T1
	5	XXF to XXG	Wed-Sun, Nov 26-30	T0	T0	T0	T0	T1	Y1	Y0	T0	T1	T1
	6	XXG to XXH	Mon-Fri, Dec 1-5	T0	T0	T0	Y0	Y1	T1	T0	T0	T1	T1
	7	XXH to XXI	Sat-Wed, Dec 6-10	T0	Y0	Y0	T0	T1	T1	T0	T0	T1	T1
	8	XXI to XXJ	Thur-Mon, Dec 11-15	Y0	T0	T0	T0	T1	T1	T0	T0	T1	T1
	9	XXJ to XXA	Tue-Sat, Dec 16-20	T0	T0	T0	T0	T1	T1	T0	T0	T1	T1
$11.21	0	XXA to XXB	Sat-Wed, Nov 1-5	W0	T0	T1	W1	T0	T0	T1	T1	T1	T1
	1	XXB to XXC	Thur-Mon, Nov 6-10	T0	T0	W1	T0	T0	X0	T1	T1	T1	T1
	2	XXC to XXD	Tue-Sat, Nov 11-15	T0	T0	T0	T0	X0	T1	T1	W1	T1	T1
	3	XXD to XXE	Sun-Thur, Nov 16-20	T0	T0	T0	T0	T1	T1	W1	T0	T1	X1
	4	XXE to XXF	Fri-Tue, Nov 21-25	T0	T0	T0	T0	T1	T1	T0	Z0	X1	T1
	5	XXF to XXG	Wed-Sun, Nov 26-30	T0	T0	T0	T0	T1	Z1	Z0	T0	T1	T1
	6	XXG to XXH	Mon-Fri, Dec 1-5	T0	T0	T0	Z0	Z1	T1	T0	T0	T1	T1
	7	XXH to XXI	Sat-Wed, Dec 6-10	T0	Z0	Z0	T0	T1	T1	T0	T0	T1	T1
	8	XXI to XXJ	Thur-Mon, Dec 11-15	Z0	T1	T0	T0	T1	T1	T0	T0	T1	T1
	9	XXJ to XXA	Tue-Sat, Dec 16-20	T1	T1	T0	T0	T1	T1	T0	T0	T1	T1

Here are two solutions from a more complicated four state deterministic TM that moves both right and left over a 5-cell tape. This machine rotates the input bits one to the right: the blue (W) and green (X) states remember the previous bit when marching to the right, and the red (Y) and yellow (Z) states remember the rightmost bit while marching to the left to deposit it in the first tape position. An alternative implementation would have been a non-deterministic machine that guessed what bit to write in the first cell, remembered its guess, and then validated it against the last cell of the tape.



Multiplication

- Implement multiplication circuits
  - Both unary and binary multiplication
  - Unary is core of undecidability proof
  - Not based on TMs, but just as with TM simulation, round-trip PUs used to encode finite-state transducers
- Multiply: solutions that start with flight sequences “17” and “19”
- Divide: solutions that start with flight sequence “17” and end in flight sequence “323”
- Factor: solutions that end with flight sequence “323”

Here two multiplication circuits are implemented, one unary and one binary. However the unary multiplication circuit does not include the complications necessary for unbounded tape length. These are custom circuits not based on Turing Machines, but that use the same mechanisms to copy information forward through the steps of a computation. The search engine searches over all possible inputs, so its output is a “times table”. To aid interpretation, values have been assigned to fares such that the dollar amount is equal to the product of the 10-cent and 1-cent position (i.e., $21.73 indicates that 21 is 7 times 3). Multiplication, division and factoring are the imposition of constraints on different parts of the trip.

Unary Multiplication

mid mid mid mid mid mid mid mid mid mid mid

Cost	Slice	From/To	Date	day of dep.	next day	+ 2 days	+ 3 days	+ 4 days	+ 5 days	+ 6 days	+ 7 days	+ 8 days	+ 9 days
$16.28	0	XXA to XXB	Sat-Mon, Nov 1-10	AA	AA	BB	BB	BB	BB	BB	BB	BB	BB
	1	XXB to XXC	Tue-Thur, Nov 11-20	SS	SS	II	II	II	II	II	II	II	II
	2	XXC to XXD	Fri-Sun, Nov 21-30	SS	SS	II	II	II	II	II	II	II	EE
	3	XXD to XXE	Mon-Wed, Dec 1-10	SS	SS	II	II	II	II	II	II	EE	EE
	4	XXE to XXF	Thur-Sat, Dec 11-20	SS	SS	II	II	II	II	II	EE	EE	EE
	5	XXF to XXG	Sun-Tue, Dec 21-30	SS	SS	II	II	II	II	EE	EE	EE	EE
	6	XXG to XXH	Wed-Fri, Dec 31-Jan 9	SS	SS	II	II	II	EE	EE	EE	EE	EE
	7	XXH to XXI	Sat-Mon, Jan 10-19	SS	SS	II	II	EE	EE	EE	EE	EE	EE
	8	XXI to XXJ	Tue-Thur, Jan 20-29	SS	SS	II	EE	EE	EE	EE	EE	EE	EE
	9	XXJ to XXA	Fri-Sun, Jan 30-Feb 8	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
$16.44	0	XXA to XXB	Sat-Mon, Nov 1-10	AA	AA	AA	AA	BB	BB	BB	BB	EE	EE
	1	XXB to XXC	Tue-Thur, Nov 11-20	SS	SS	SS	SS	II	II	II	II	EE	EE
	2	XXC to XXD	Fri-Sun, Nov 21-30	SS	SS	SS	SS	II	II	II	EE	EE	EE
	3	XXD to XXE	Mon-Wed, Dec 1-10	SS	SS	SS	SS	II	II	EE	EE	EE	EE
	4	XXE to XXF	Thur-Sat, Dec 11-20	SS	SS	SS	SS	II	EE	EE	EE	EE	EE
	5	XXF to XXG	Sun-Tue, Dec 21-30	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	6	XXG to XXH	Wed-Fri, Dec 31-Jan 9	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	7	XXH to XXI	Sat-Mon, Jan 10-19	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	8	XXI to XXJ	Tue-Thur, Jan 20-29	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	9	XXJ to XXA	Fri-Sun, Jan 30-Feb 8	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
$16.82	0	XXA to XXB	Sat-Mon, Nov 1-10	AA	AA	AA	AA	AA	AA	AA	AA	BB	BB
	1	XXB to XXC	Tue-Thur, Nov 11-20	SS	SS	SS	SS	SS	SS	SS	SS	II	II
	2	XXC to XXD	Fri-Sun, Nov 21-30	SS	SS	SS	SS	SS	SS	SS	SS	II	EE
	3	XXD to XXE	Mon-Wed, Dec 1-10	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	4	XXE to XXF	Thur-Sat, Dec 11-20	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	5	XXF to XXG	Sun-Tue, Dec 21-30	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	6	XXG to XXH	Wed-Fri, Dec 31-Jan 9	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	7	XXH to XXI	Sat-Mon, Jan 10-19	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	8	XXI to XXJ	Tue-Thur, Jan 20-29	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE
	9	XXJ to XXA	Fri-Sun, Jan 30-Feb 8	EE	EE	EE	EE	EE	EE	EE	EE	EE	EE

Unary multiplication is fairly easy (at least for the case of a bounded tape). The red and green cells are the input, and the number of pink cells is the output. Blue cells implement a decrementing counter, and gray cells are "empty".

Copyright 2003 ITA Software

Binary Multiplication

Cost	Slice	From/To	Date	day of dep.	next day	+ 2 days	+ 3 days	+ 4 days	+ 5 days	+ 6 days	+ 7 days	+ 8 days
$18.36	0	XXA to XXB	Sat-Sun, Nov 1-9	A0	A1	A1	B1	S0	B1	S0	B0	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A0	A1	B1	S0	B1	S1	B0	S1	S0
	2	XXC to XXD	Wed-Thur, Nov 19-27	A0	B1	S1	B1	S0	B0	S0	S1	S0
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B1	S0	B1	S1	B0	S0	S0	S1	S0
$18.63	0	XXA to XXB	Sat-Sun, Nov 1-9	A1	A1	A0	B0	S0	B1	S0	B1	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A1	A1	B0	S0	B1	S0	B1	S0	S0
	2	XXC to XXD	Wed-Thur, Nov 19-27	A1	B0	S0	B1	S0	B1	S1	S1	S0
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B0	S0	B1	S1	B1	S0	S0	S1	S0
$20.45	0	XXA to XXB	Sat-Sun, Nov 1-9	A1	A0	A0	B1	S0	B0	S0	B1	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A1	A0	B1	S0	B0	S0	B1	S0	S0
	2	XXC to XXD	Wed-Thur, Nov 19-27	A1	B1	S0	B0	S0	B1	S0	S0	S0
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B1	S0	B0	S1	B1	S0	S1	S0	S0
$20.54	0	XXA to XXB	Sat-Sun, Nov 1-9	A1	A0	A1	B1	S0	B0	S0	B0	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A1	A0	B1	S0	B0	S1	B0	S0	S0
	2	XXC to XXD	Wed-Thur, Nov 19-27	A1	B1	S0	B0	S0	B0	S1	S0	S0
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B1	S0	B0	S1	B0	S0	S1	S0	S0
$21.37	0	XXA to XXB	Sat-Sun, Nov 1-9	A0	A1	A1	B1	S0	B1	S0	B1	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A0	A1	B1	S0	B1	S1	B1	S1	S1
	2	XXC to XXD	Wed-Thur, Nov 19-27	A0	B1	S1	B1	S0	B1	S1	S0	S1
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B1	S0	B1	S1	B1	S0	S1	S0	S1
$21.73	0	XXA to XXB	Sat-Sun, Nov 1-9	A1	A1	A1	B0	S0	B1	S0	B1	S0
	1	XXB to XXC	Mon-Tue, Nov 10-18	A1	A1	B0	S0	B1	S0	B1	S1	S1
	2	XXC to XXD	Wed-Thur, Nov 19-27	A1	B0	S0	B1	S1	B1	S0	S0	S1
	3	XXD to XXA	Fri-Sat, Nov 28-Dec 6	B0	S0	B1	S1	B1	S0	S1	S0	S1

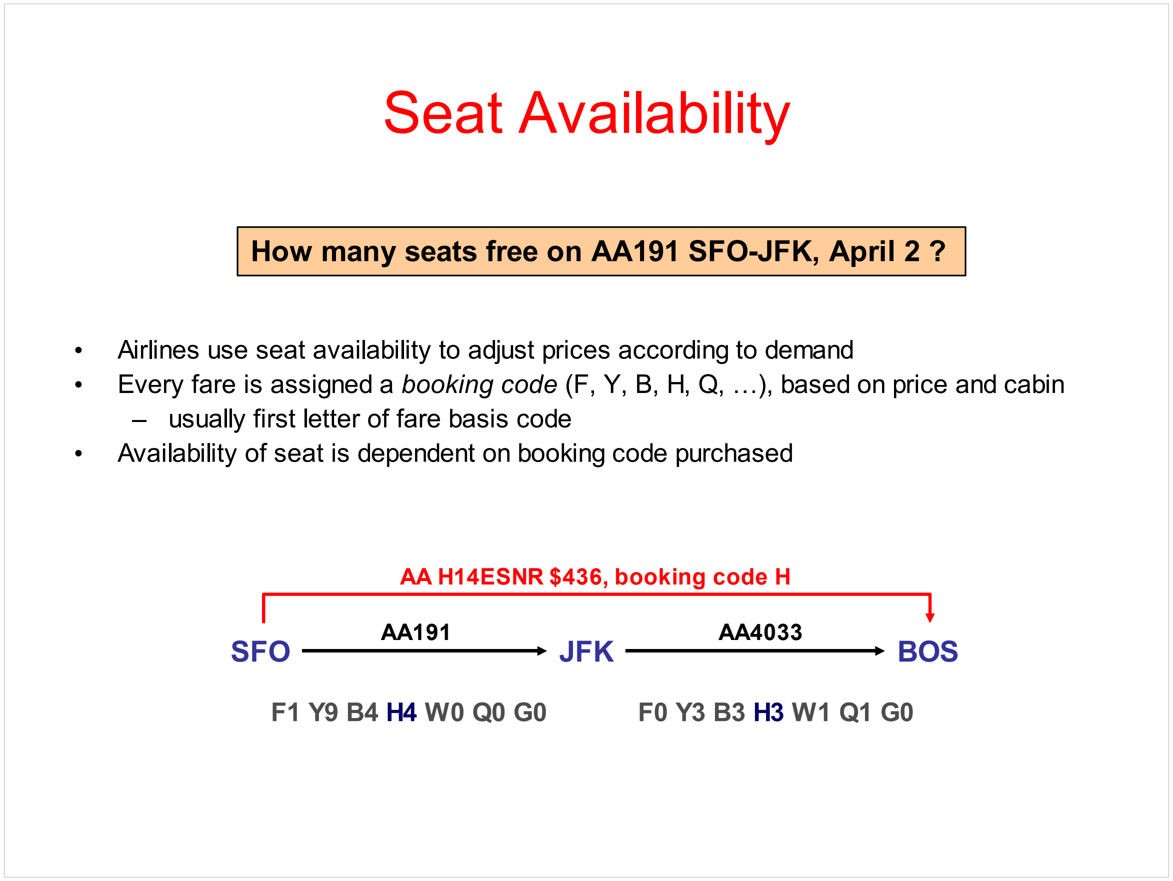
Binary multiplication is a more complicated circuit, but can multiply bigger numbers with less tape and fewer time steps. Six solutions are shown, from a 3-bit x 3-bit multiplier. Can you figure out the logic? It's standard grade-school multiplication. Green and blue cells represent the input numbers, red the accumulated result. (Exercise: write the one-step FST for this circuit.)

The binary circuit's time and space advantages don't come for free. In the unary circuit the number of possible flight sequences for each line was quadratic in the length of the tape: there was one transition from red to blue, and one from blue to gray. But the binary multiplier has choices of 0 or 1 for each cell, exponential in the length of the tape. Thus there are exponentially greater many flight sequences to search over. As of September, 2003 the ITA Software search engine maxes out its search capabilities at 4-bit multiplication – the times table up to 15 times 15, with 130 flights and fares per solution.

Outline

- Introduction
- Flights
- How airline prices work
- Complexity of travel planning
- Demos
- **Seat Availability**
- Further Reading

Although not such a fundamental issue in the theoretical computational complexity of travel planning, airline seat availability processing is a key component of the air travel pricing framework and has a huge impact on the prices passengers see, as well as the practical difficulty of finding the lowest price. Another reason to understand seat availability is that it is one of the areas of the airline industry most studied in academia, including at MIT.



Seat Availability

How many seats free on AA191 SFO-JFK, April 2 ?

- Airlines use seat availability to adjust prices according to demand
- Every fare is assigned a *booking code* (F, Y, B, H, Q, …), based on price and cabin
  - usually first letter of fare basis code
- Availability of seat is dependent on booking code purchased

AA H14ESNR $436, booking code H

SFO → AA191 → JFK → AA4033 → BOS

F1 Y9 B4 **H4** W0 Q0 G0

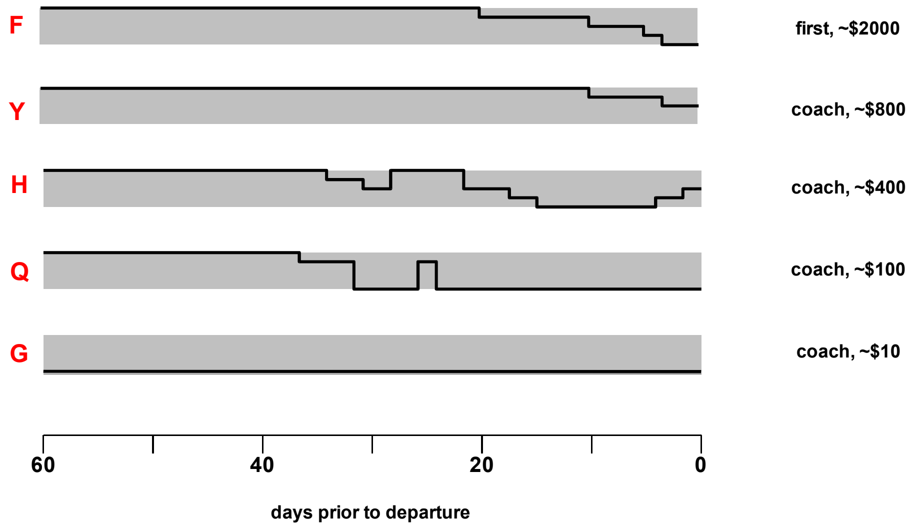
F0 Y3 B3 **H3** W1 Q1 G0

Airline seat availability is much more complicated than just the question of whether the number of reserved seats equals the capacity of an aircraft. Airlines use seat availability as a way to dynamically adjust prices according to demand. Simplifying somewhat, each published fare is assigned a letter of the alphabet called a **booking code**, typically also the first letter of the fare's basis code. The airline chooses the booking code for a fare based primarily on the fare's cabin (coach, business or first) and the fare's price.

Asked whether there are any seats available on a plane, the response an airline gives is not "yes" or "no" but rather a per-booking-code vector of seat counts. For example, in the figure the first flight has 1 F booking code and 4 H booking codes available; the second has no F's and 3 H's. To fly on these two flights using the H14ESNR fare (with booking code H), H seats must be available for both flights. They are, and up to 3 people could buy H fares, but a (cheaper) fare with booking code Q could not be used because no Q seats are available for the first flight.

Airlines do not usually publish seat counts higher than 9, so even when a plane is empty it is common to see F9 Y9 B9...

Availability Dynamics

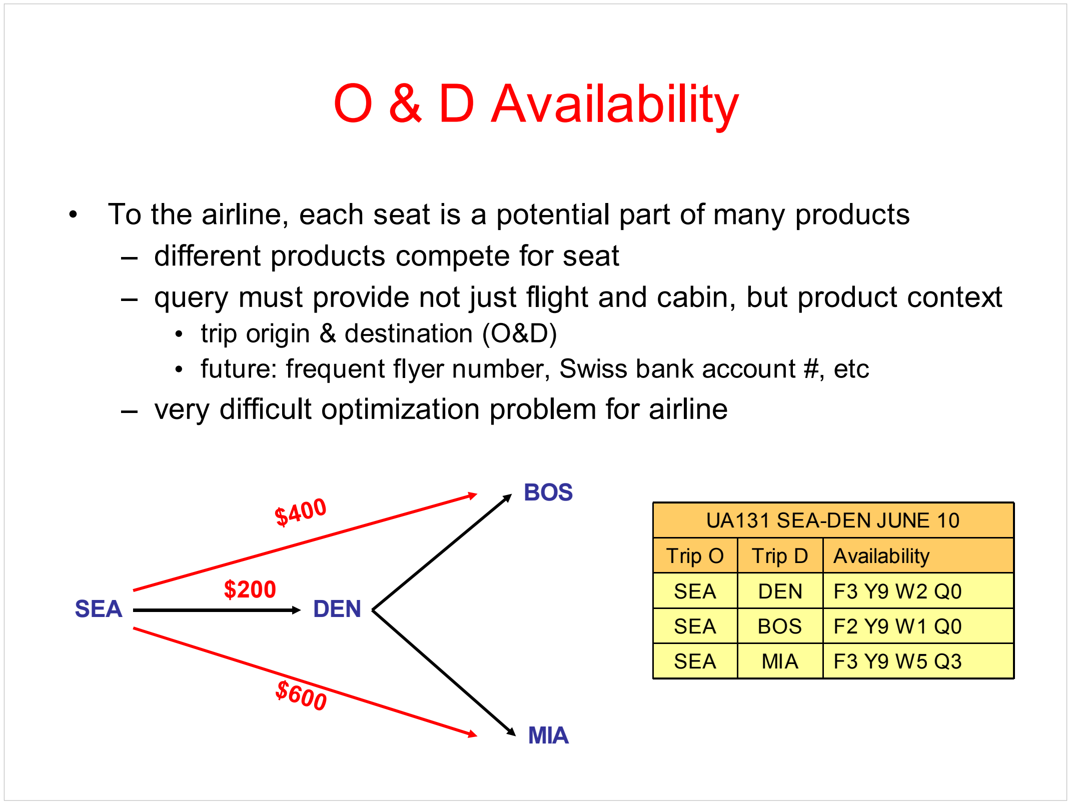


Airlines dynamically adjust their responses to seat availability queries as they estimate demand for flights. The simplest case is for the most expensive fares. For example, months before a flight leaves all (first class) F seats will be available but as seats are reserved the counts slowly drops until the plane is full; similarly for the most expensive coach class booking code, Y.

But cheaper coach class booking codes like H have response profiles that reflect demand as well as capacity. Suppose the airline sees very high demand for this flight relative to similar flights in the past. They may decide to stop selling cheaper seats so as to force passengers to pay more, or viewed another way, so as to save seats for those who would pay more. Some cheap booking codes might not normally be available at all, and might only be enabled in very low demand situations. Importantly, the information the airline uses to estimate demand changes constantly, so seat availability responses may fluctuate up and down even in absence of any reservations.

One of the biggest problems for the airline is predicting demand. They devote huge efforts to “cleaning” historical data for use in training statistical demand models. Imagine trying to accurately predict demand immediately after a strike, or a plane crash, or for flights to the city hosting the Olympics.

Copyright 2003 ITA Software



O & D Availability

- To the airline, each seat is a potential part of many products
  - different products compete for seat
  - query must provide not just flight and cabin, but product context
    - trip origin & destination (O&D)
    - future: frequent flyer number, Swiss bank account #, etc
  - very difficult optimization problem for airline

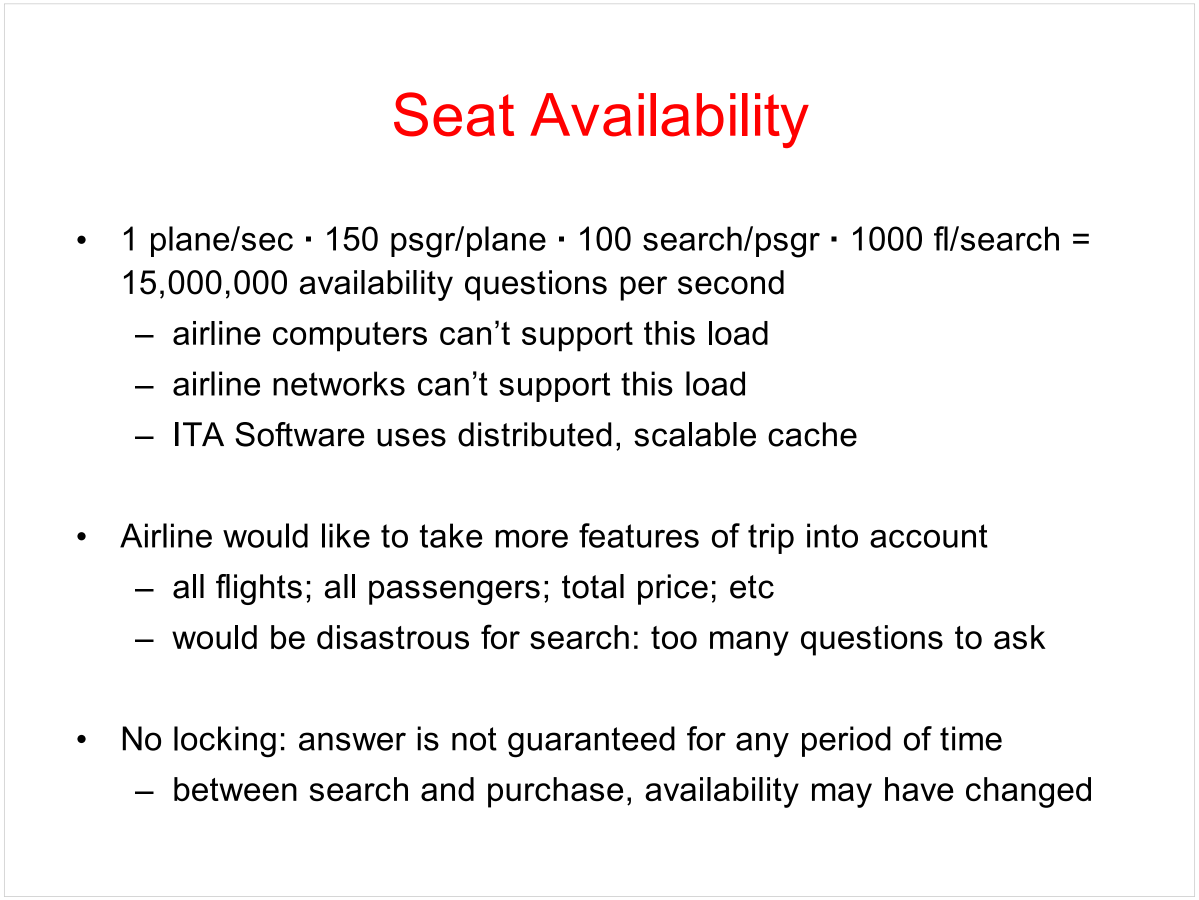
BOS
$400
$200
SEA
DEN
$600
MIA

UA131 SEA-DEN JUNE 10		
Trip O	Trip D	Availability
SEA	DEN	F3 Y9 W2 Q0
SEA	BOS	F2 Y9 W1 Q0
SEA	MIA	F3 Y9 W5 Q3

A further complication to seat availability, especially to the airline, is that each seat is a potential part of many “products”, and all these products compete for that seat. The same seat on a SEA to DEN flight could be used for a Y fare for a passenger traveling from SEA to DEN, or could be part of a Q fare from SEA to BOS with a stop in DEN. Selling a seat on one flight might fill that plane, making it impossible for someone else to use that plane as part of a bigger trip that might bring more money to the airline. So many routes and many fares are all competing for seats. This makes it much more challenging for an airline to decide whether or not to offer a given booking code on a given flight. A consequence is that many airlines demand information about the entire route of a trip before providing availability for any section: this is often called O&D (origin and destination) availability. The table shows how the seat availability for a single flight can vary depending on what trip the flight is part of. When the availability of two flights depends on the passenger taking both, the flights are said to be **married**.

Research in **network revenue management** is a very popular topic. MIT's operations research (OR) community participates heavily. Most work is based on linear programming models but simulation techniques are gaining in popularity. One fundamental result is that complicated revenue management techniques only significantly increase profits when planes are flying near capacity.

Copyright 2003 ITA Software

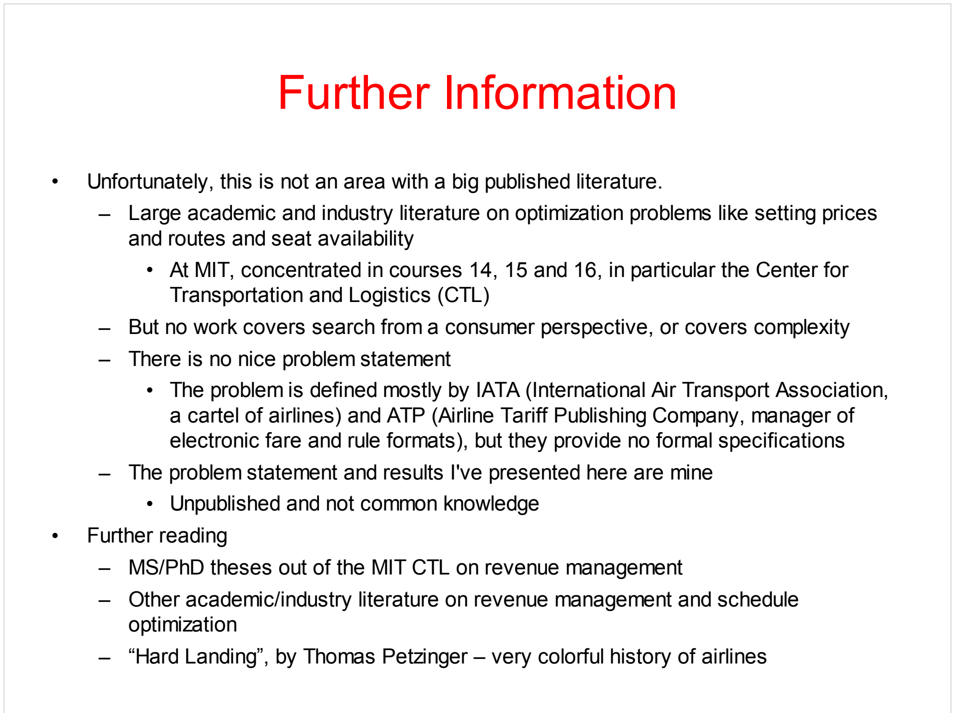


Seat Availability

- 1 plane/sec · 150 psgr/plane · 100 search/psgr · 1000 fl/search = 15,000,000 availability questions per second
  - airline computers can't support this load
  - airline networks can't support this load
  - ITA Software uses distributed, scalable cache
- Airline would like to take more features of trip into account
  - all flights; all passengers; total price; etc
  - would be disastrous for search: too many questions to ask
- No locking: answer is not guaranteed for any period of time
  - between search and purchase, availability may have changed

Another interesting aspect of seat availability is the number of questions that the airlines must answer. If every passenger poses 100 searches before buying a ticket (a number in line with actual behavior) and each search looks at 1,000 flights, then the airlines would need to answer 15,000,000 questions a second. Neither their networks nor their computers can handle this, a situation that forced ITA Software to develop a sophisticated seat availability caching system. The problem is aggravated by O&D availability: if the entire trip must be included in each query, something many airlines desire, then search becomes impossible because every potential solution must be independently validated with the airline.

Finally, the airlines' seat availability infrastructure does not include any locking mechanism, so even if an airline responds that a booking code is available, there is no guarantee that by the time a traveler says "yes, I'd like to buy it" it still will be.



Further Information

- Unfortunately, this is not an area with a big published literature.
  - Large academic and industry literature on optimization problems like setting prices and routes and seat availability
    - At MIT, concentrated in courses 14, 15 and 16, in particular the Center for Transportation and Logistics (CTL)
  - But no work covers search from a consumer perspective, or covers complexity
  - There is no nice problem statement
    - The problem is defined mostly by IATA (International Air Transport Association, a cartel of airlines) and ATP (Airline Tariff Publishing Company, manager of electronic fare and rule formats), but they provide no formal specifications
  - The problem statement and results I've presented here are mine
    - Unpublished and not common knowledge
- Further reading
  - MS/PhD theses out of the MIT CTL on revenue management
  - Other academic/industry literature on revenue management and schedule optimization
  - “Hard Landing”, by Thomas Petzinger – very colorful history of airlines

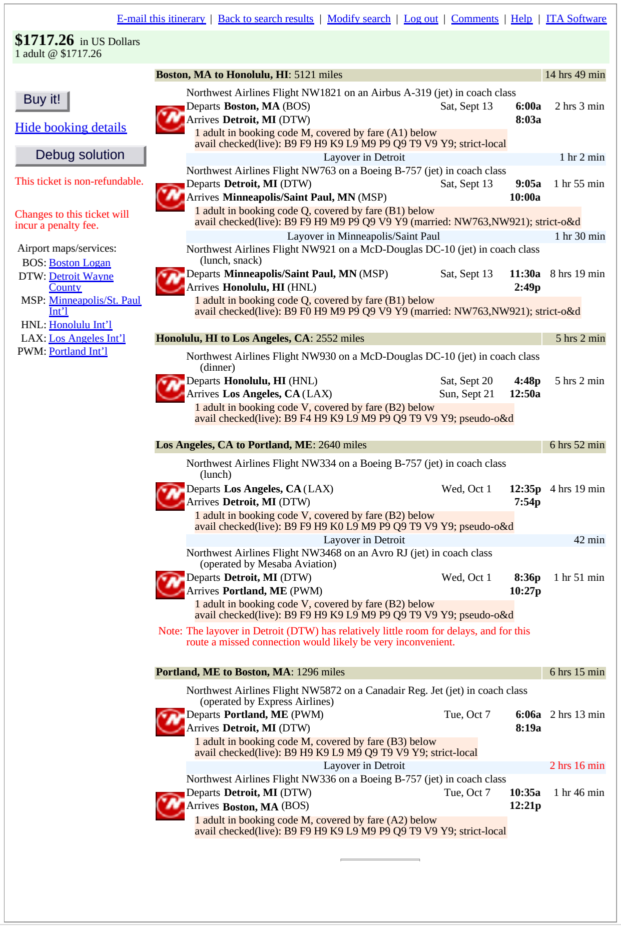
A variety of MS and PhD theses from the MIT Center for Transportation and Logistics, in particular from the students of Prof. Peter Belobaba, have readable reviews of the setting of airline prices and availability. Read, for example, Belobaba's 1987 and Williamson's 1992 MIT PhD theses to get a (now dated) understanding for some basic problems in airline revenue management. However this work concentrates on seat availability and fails to detail most of the complexities of the industry's pricing logic, and does not address search. Revenue management has spread to many other industries, and is used heavily in telecommunication and energy pricing.

Standard introductions to complexity theory include Hopcroft and Ullman (Introduction to Automata Theory, Languages and Computation), Sipser (Introduction to Theory of Computation) and Garey and Johnson (Computers and Intractability). Also highly recommended as background is Aho and Ullman (The Theory of Parsing, Translation and Compiling, volume 1) for a broad introduction to formal languages. For more information on unsolvability read the collection of papers The Undecidable (Davis, editor) and for a superb review of the unsolvability of the Diophantine decision problem read Davis's Computers and Unsolvability. This last is fascinating for any computer scientist with a mathematical inclination, as it presents a complete proof of one of the most important mathematical results of the 20th century in a form accessible to a dedicated CS undergraduate.

Hard Landing by Thomas Petzinger is a light and very enjoyable history of the airlines that includes a lot of the history behind their complicated pricing schemes.

Exercise

- This is a solution as displayed on the ITA Software web site
  - How much of this output can you understand now?
  - Draw the trip with fares, priceable units and booking codes



E-mail this itinerary | Back to search results | Modify search | Log out | Comments | Help | ITA Software

$1717.26 in US Dollars
1 adult @ $1717.26

This ticket is non-refundable.

Changes to this ticket will incur a penalty fee.

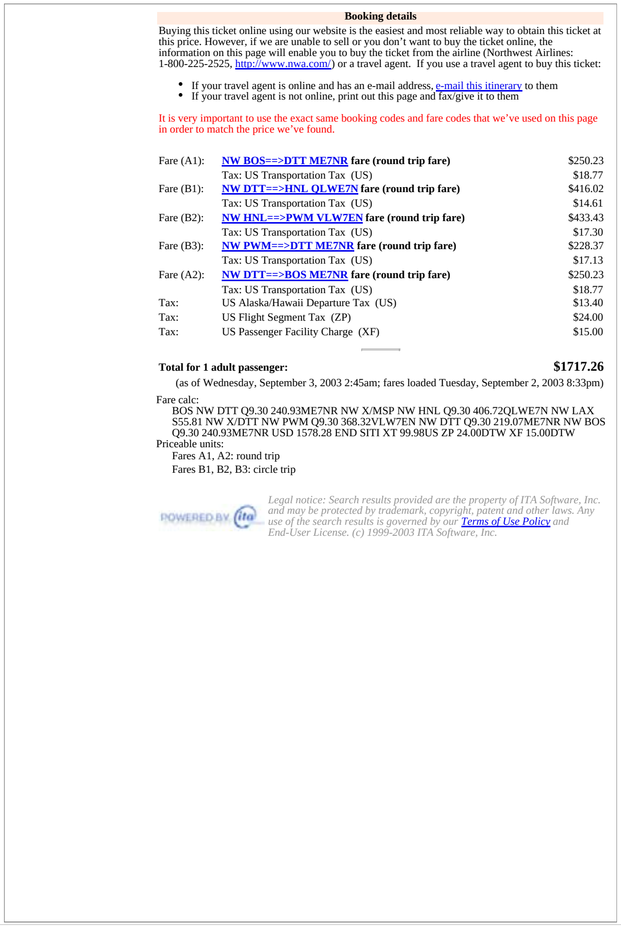
Airport maps/services:
BOS: Boston Logan
DTW: Detroit Wayne County
MSP: Minneapolis/St. Paul Int'l
HNL: Honolulu Int'l
LAX: Los Angeles Int'l
PWM: Portland Int'l

```
Boston, MA to Honolulu, HI: 5121 miles                                14 hrs 49 min
  Northwest Airlines Flight NW1821 on an Airbus A-319 (jet) in coach class
  Departs Boston, MA (BOS)                     Sat, Sept 13   6:00a   2 hrs 3 min
  Arrives Detroit, MI (DTW)                                   8:03a
    1 adult in booking code M, covered by fare (A1) below
    avail checked(live): B9 F9 H9 K9 L9 M9 P9 Q9 T9 V9 Y9; strict-local
                        Layover in Detroit                            1 hr 2 min
  Northwest Airlines Flight NW763 on a Boeing B-757 (jet) in coach class
  Departs Detroit, MI (DTW)                    Sat, Sept 13   9:05a   1 hr 55 min
  Arrives Minneapolis/Saint Paul, MN (MSP)                   10:00a
    1 adult in booking code Q, covered by fare (B1) below
    avail checked(live): B9 F9 H9 M9 P9 Q9 V9 Y9 (married: NW763,NW921); strict-o&d
                  Layover in Minneapolis/Saint Paul                   1 hr 30 min
  Northwest Airlines Flight NW921 on a McD-Douglas DC-10 (jet) in coach class
     (lunch, snack)
  Departs Minneapolis/Saint Paul, MN (MSP)     Sat, Sept 13  11:30a   8 hrs 19 min
  Arrives Honolulu, HI (HNL)                                  2:49p
    1 adult in booking code Q, covered by fare (B1) below
    avail checked(live): B9 F0 H9 M9 P9 Q9 V9 Y9 (married: NW763,NW921); strict-o&d

Honolulu, HI to Los Angeles, CA: 2552 miles                           5 hrs 2 min
  Northwest Airlines Flight NW930 on a McD-Douglas DC-10 (jet) in coach class
     (dinner)
  Departs Honolulu, HI (HNL)                   Sat, Sept 20   4:48p   5 hrs 2 min
  Arrives Los Angeles, CA (LAX)                Sun, Sept 21  12:50a
    1 adult in booking code V, covered by fare (B2) below
    avail checked(live): B9 F4 H9 K9 L9 M9 P9 Q9 T9 V9 Y9; pseudo-o&d

Los Angeles, CA to Portland, ME: 2640 miles                           6 hrs 52 min
  Northwest Airlines Flight NW334 on a Boeing B-757 (jet) in coach class
     (lunch)
  Departs Los Angeles, CA (LAX)                Wed, Oct 1    12:35p   4 hrs 19 min
  Arrives Detroit, MI (DTW)                                   7:54p
    1 adult in booking code V, covered by fare (B2) below
    avail checked(live): B9 F9 H9 K0 L9 M9 P9 Q9 T9 V9 Y9; pseudo-o&d
                        Layover in Detroit                            42 min
  Northwest Airlines Flight NW3468 on an Avro RJ (jet) in coach class
     (operated by Mesaba Aviation)
  Departs Detroit, MI (DTW)                    Wed, Oct 1     8:36p   1 hr 51 min
  Arrives Portland, ME (PWM)                                 10:27p
    1 adult in booking code V, covered by fare (B2) below
    avail checked(live): B9 F9 H9 K9 L9 M9 P9 Q9 T9 V9 Y9; pseudo-o&d
Note: The layover in Detroit (DTW) has relatively little room for delays, and for this
      route a missed connection would likely be very inconvenient.

Portland, ME to Boston, MA: 1296 miles                                6 hrs 15 min
  Northwest Airlines Flight NW5872 on a Canadair Reg. Jet (jet) in coach class
     (operated by Express Airlines)
  Departs Portland, ME (PWM)                   Tue, Oct 7     6:06a   2 hrs 13 min
  Arrives Detroit, MI (DTW)                                   8:19a
    1 adult in booking code M, covered by fare (B3) below
    avail checked(live): B9 H9 K9 L9 M9 Q9 T9 V9 Y9; strict-local
                        Layover in Detroit                            2 hrs 16 min
  Northwest Airlines Flight NW336 on a Boeing B-757 (jet) in coach class
  Departs Detroit, MI (DTW)                    Tue, Oct 7    10:35a   1 hr 46 min
  Arrives Boston, MA (BOS)                                   12:21p
    1 adult in booking code M, covered by fare (A2) below
    avail checked(live): B9 F9 H9 K9 L9 M9 P9 Q9 T9 V9 Y9; strict-local
```



Booking details

Buying this ticket online using our website is the easiest and most reliable way to obtain this ticket at this price. However, if we are unable to sell or you don't want to buy the ticket online, the information on this page will enable you to buy the ticket from the airline (Northwest Airlines: 1-800-225-2525, http://www.nwa.com/) or a travel agent. If you use a travel agent to buy this ticket:

- If your travel agent is online and has an e-mail address, e-mail this itinerary to them
- If your travel agent is not online, print out this page and fax/give it to them

It is very important to use the exact same booking codes and fare codes that we've used on this page in order to match the price we've found.

Fare (A1):	**NW BOS==>DTT ME7NR fare (round trip fare)**	$250.23
	Tax: US Transportation Tax (US)	$18.77
Fare (B1):	**NW DTT==>HNL QLWE7N fare (round trip fare)**	$416.02
	Tax: US Transportation Tax (US)	$14.61
Fare (B2):	**NW HNL==>PWM VLW7EN fare (round trip fare)**	$433.43
	Tax: US Transportation Tax (US)	$17.30
Fare (B3):	**NW PWM==>DTT ME7NR fare (round trip fare)**	$228.37
	Tax: US Transportation Tax (US)	$17.13
Fare (A2):	**NW DTT==>BOS ME7NR fare (round trip fare)**	$250.23
	Tax: US Transportation Tax (US)	$18.77
Tax:	US Alaska/Hawaii Departure Tax (US)	$13.40
Tax:	US Flight Segment Tax (ZP)	$24.00
Tax:	US Passenger Facility Charge (XF)	$15.00

Total for 1 adult passenger: **$1717.26**

(as of Wednesday, September 3, 2003 2:45am; fares loaded Tuesday, September 2, 2003 8:33pm)

Fare calc:

```
    BOS NW DTT Q9.30 240.93ME7NR NW X/MSP NW HNL Q9.30 406.72QLWE7N NW LAX
    S55.81 NW X/DTT NW PWM Q9.30 368.32VLW7EN NW DTT Q9.30 219.07ME7NR NW BOS
    Q9.30 240.93ME7NR USD 1578.28 END SITI XT 99.98US ZP 24.00DTW XF 15.00DTW
```

Priceable units:
  Fares A1, A2: round trip
  Fares B1, B2, B3: circle trip

Legal notice: Search results provided are the property of ITA Software, Inc. and may be protected by trademark, copyright, patent and other laws. Any use of the search results is governed by our Terms of Use Policy and End-User License. (c) 1999-2003 ITA Software, Inc.

This solution was one of 2,197,704,882,975,408 the ITA Software search engine found for a BOS-HNL-LAX-PWM-BOS circle-trip query, with one-day departure windows for each part of the trip. It was neither the cheapest nor the most expensive.