Building Software Agents for Planning, Monitoring, and Optimizing Travel

Craig A. Knoblock

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292 USA
knoblock@isi.edu

Abstract

Planning and executing a trip requires assembling a wide variety of interacting information from a large number of sources, including information on flight schedules and prices, hotel locations and reviews, ground transportation options, weather conditions, airport delays, flight cancellations, etc. Much of this information is now available on the Internet and it can be used to enable travelers to better plan and execute their trips. This paper describes the use of software agents for extracting, integrating and mining online data sources to improve the ability to plan, monitor, and optimize travel. These agents can dynamically extract data from online travel sources, integrate this data to support interactive travel planning, continuously monitor all aspects of a trip to ensure a trip goes smoothly, and exploit data mining to make predictions that can either save a traveler money or improve the likelihood of a successful trip.

Keywords: software agents, wrappers, interactive planning, data mining, travel, and tourism.

1 Introduction

The standard approach to planning business trips is to select the flights, reserve a hotel, and possibly reserve a car at the destination. The choices of which airports to fly into and out of, whether to park at the airport or take a taxi, and whether to rent a car at the destination are often made in an ad hoc way based on past experience. These choices are frequently suboptimal, but the time and effort required to make more informed choices usually outweighs the cost. Similarly, once a trip has been planned it is usually ignored until a few hours before the first flight. A traveler might check on the status of the flights or use one of the services that automatically notify a traveler of flight status information, but otherwise a traveler just copes with problems that arise, as they arise. Beyond flight delays and cancellations there are a variety of possible events that occur in the real world that one would ideally like to anticipate, but again the cost and effort required to monitor for these events is not usually deemed to be worth the trouble. Schedules can change, prices may go down after purchasing a ticket, flight delays can result in missed connections, and hotel rooms and rental cars are given away because travelers arrive late.

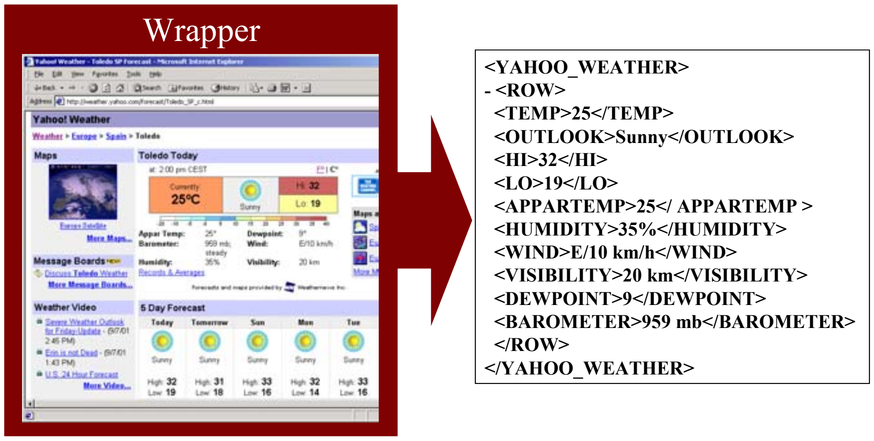
There is now a wealth of information available on the Web that can be exploited for planning, monitoring, and optimizing travel. The data is now readily available to make more informed choices that consider all aspects of a trip, including the cost of ground transportation, availability of restaurants, parking rates, etc. The online information can also be used to monitor all aspects of a trip to ensure it goes smoothly. Finally, this information can also be mined to optimize travel decisions by making predictions about flight prices, hotel prices, or even the likelihood of a delay at a particular airport.

This paper presents a set of software agents (Knoblock 2003) that exploit the availability of online information for travel planning. First, the paper describes how software agents are built to turn online HTML web data into information that can be used by other agents (Section 2). Next, the paper presents an interactive planning system, called Heracles, where all of the information required to make informed choices is available to a user (Section 3). For example, when deciding whether to park at the airport or take a taxi, the system compares the cost of parking and the cost of a taxi given other selections, such as the airport, the specific parking lot, and the starting location of the traveler. Then, the paper describes the construction and execution of software agents that are capable of monitoring all aspects of a trip (Section 4). For example, beyond simply notifying a traveler of flight delays, an agent will also send a fax to a hotel to notify them of the delay and ensure that a room will be available. Next, the paper describes work on mining the information available online to make predictions that aid the traveler in their decision making (Section 5). For example, the paper describes a system called Hamlet that learns a model of flight prices in order to help the user decide whether they should buy a ticket right away or wait to buy a ticket. Finally, the paper reviews related work (Section 6) and concludes with a discussion of the impact of online sources on the travel industry.

2 Agent Access to Online Sources

A key capability for information agents is the ability to reliably access information. As the Web moves towards XML and Web Services, accessing data could become greatly simplified. However, movement in this direction has been quite slow and for various reasons many sources will remain available only in HTML, so there is still a critical need to turn HTML sources into agent-enabled sources.

In order to provide access to the data in an existing HTML source, we construct what is called a *wrapper*. A wrapper is simply a program that understands the structure of a specific web site and uses that knowledge to accept queries to that site and produce answers to those queries in a structured format, such as XML (see Fig. 1). The challenge in building wrappers for online sources is how to achieve broad coverage and high accuracy with minimal user effort. The two general approaches to this problem are supervised machine learning techniques (Kushmerick 1997; Hsu et al. 1998; Muslea et al. 2001) and unsupervised grammar induction techniques (Crescenzi et al. 2001; Lerman et al. 2001). Unsupervised grammar induction has the advantage



```
<YAHOO_WEATHER>
- <ROW>
  <TEMP>25</TEMP>
  <OUTLOOK>Sunny</OUTLOOK>
  <HI>32</HI>
  <LO>19</LO>
  <APPARTEMP>25</ APPARTEMP >
  <HUMIDITY>35%</HUMIDITY>
  <WIND>E/10 km/h</WIND>
  <VISIBILITY>20 km</VISIBILITY>
  <DEWPOINT>9</DEWPOINT>
  <BAROMETER>959 mb</BAROMETER>
  </ROW>
</YAHOO_WEATHER>
```

Fig. 1: Wrapper converts Yahoo Weather into XML data, which can be interpreted by a software agent

of no required user input, but it is not able to handle the full range of online sources. In contrast, the supervised learning techniques apply to a wider set of sites, but can require a significant amount of labeled data to achieve high accuracy.

We developed a machine learning algorithm called Stalker (Muslea 2002) that requires labeled data, but attempts to minimize the amount of information that must be provided by a user. Given labeled data, the system employs an inductive learning algorithm to generate extraction rules that define a wrapper for a source. We minimize the amount of labeled data required by decomposing the learning problem into a number of simpler subproblems, which require fewer examples to learn. The decomposition is based on the hierarchical structure of the information in a web source. This approach allows Stalker to learn how to extract data from complicated sites that involve lists of information and even arbitrary nesting of embedded lists.

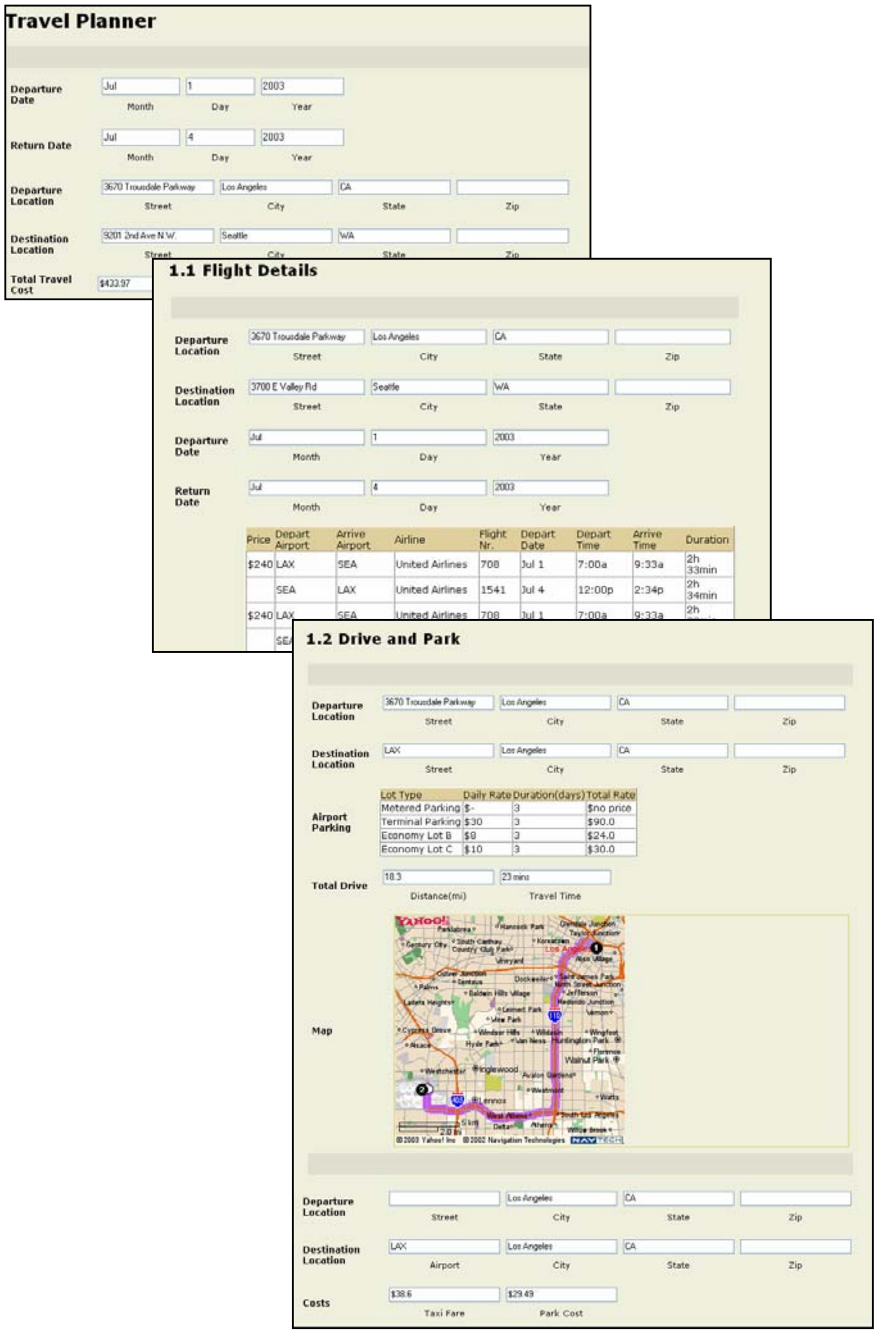
An issue for any learning system, even Stalker, is that to achieve high accuracy the system must see the right set of examples. Since the expectation in a wrapper is to extract the data with 100% accuracy, finding a representative set of examples is a critical part of the problem. Rather than relying on the user to identify these examples, we developed an active learning technique called Co-Testing (Muslea et al. 2000) that selects the most information examples to label. Co-Testing works by learning multiple classifiers using different views of the same problem. In the case of wrapper learning, the system exploits the fact that it can learn equally well a classifier by finding landmarks from the beginning of the page or by finding landmarks from the end of the page. The system can then exploit the fact that both classifiers should agree if they have learned the same concept and any disagreement provides a source of training examples. Both classifiers are applied to the unlabeled examples and the user is asked to label the examples when there is disagreement.

Another important challenge in building wrappers is to ensure that they continue to work properly over time. We developed a wrapper maintenance system that can repair wrappers by learning a description of the content extracted by a wrapper (Lerman et al. 2003). This approach learns a pattern by using a hierarchy of pattern types, such as number or capitalized word, and then learning a description of the beginning and ending of the information that is being extracted. The resulting description or learned patterns are then stored and compared to the information being extracted. The patterns are compared statistically to avoid false positives due to examples that have not been seen before. In a large test set, this approach was able to identify 95% of the Web sites that had changed. Once the system has identified a source that has changed, the learned patterns can then be used to automatically relabel the site and run the labeled examples through Stalker to relearn the wrapper.

3 Interactive Planning of a Trip

We developed a general, interactive, constraint-based planner, called Heracles (Knoblock et al. 2001), which we then applied to the problem of planning travel. The resulting system integrates a wide variety of travel related data from web sources to provide the data that travelers need to plan a trip (Ambite et al. 2002). This system uses information agents described in the previous section to provide real-time access to the many online sources related to travel. A traveler enters their origin and destination addresses and the dates of his/her trip and then the travel planner interactively helps the traveler plan the trip. The system provides the choices of flights, hotels, ground transportations, etc. For each decision, the system makes a recommendation that optimizes a user-specified criterion, such as minimizing the overall cost of a trip.

Consider the planning process shown in Figs. 2 and 3. The user first specifies their starting and destination address and dates of a trip. The system then plans the trip, selecting the closest airports and proposing specific flights (1.1), comparing the cost of driving and parking a car at the airport to the cost of taking a taxi (1.2), comparing the cost of renting a car and driving to the destination address versus taking a taxi (1.3), and recommending a hotel near the destination that has the lowest price (2). At each step the user can view all of the choices and override the recommendation made by the system. All of the data needed to make informed choices is not only available to the user, but is also made explicit to the system using constraints that connect the choices. This means that the choices in one part of a plan are immediately reflected in other parts as well. For example, if the traveler decides to depart from Long Beach airport instead of LAX, then the system would immediately recompute the cost of the flight, determine the cheapest form of ground transportation to the airport from the starting address, retrieve new directions to the airport, and update the time that the traveler would need to leave his/her house based on both the distance and updated flight time. The propagation of updates of all of the related information occurs automatically as part of the constraint network.



Travel Planner

Departure Date	Jul (Month)	1 (Day)	2003 (Year)	
Return Date	Jul (Month)	4 (Day)	2003 (Year)	
Departure Location	3670 Trousdale Parkway (Street)	Los Angeles (City)	CA (State)	(Zip)
Destination Location	9201 2nd Ave N.W. (Street)	Seattle (City)	WA (State)	(Zip)
Total Travel Cost	$433.97			

1.1 Flight Details

Departure Location	3670 Trousdale Parkway (Street)	Los Angeles (City)	CA (State)	(Zip)
Destination Location	3700 E Valley Rd (Street)	Seattle (City)	WA (State)	(Zip)
Departure Date	Jul (Month)	1 (Day)	2003 (Year)	
Return Date	Jul (Month)	4 (Day)	2003 (Year)	

Price	Depart Airport	Arrive Airport	Airline	Flight Nr.	Depart Date	Depart Time	Arrive Time	Duration
$240	LAX	SEA	United Airlines	708	Jul 1	7:00a	9:33a	2h 33min
	SEA	LAX	United Airlines	1541	Jul 4	12:00p	2:34p	2h 34min
$240	LAX	SEA	United Airlines	708	Jul 1	7:00a	9:33a	2h
	SEA							

1.2 Drive and Park

Departure Location	3670 Trousdale Parkway (Street)	Los Angeles (City)	CA (State)	(Zip)
Destination Location	LAX (Street)	Los Angeles (City)	CA (State)	(Zip)

Airport Parking

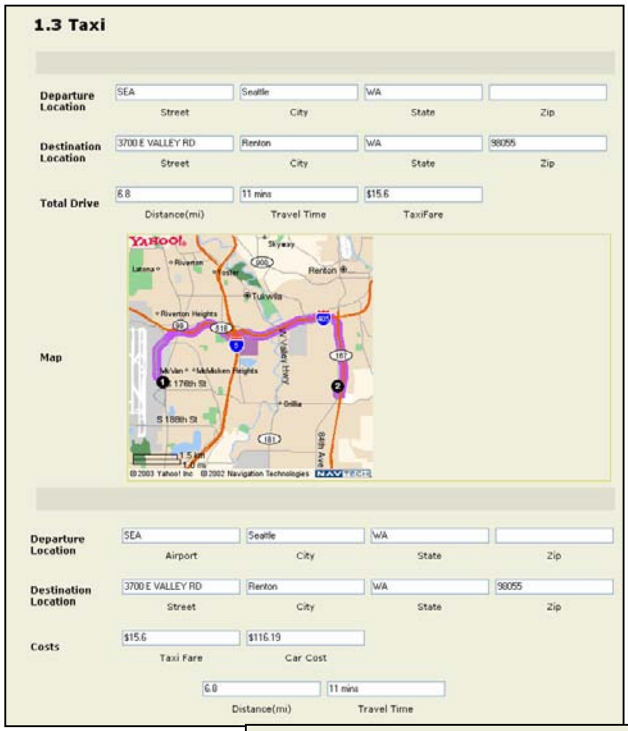
Lot Type	Daily Rate	Duration(days)	Total Rate
Metered Parking	$-	3	$no price
Terminal Parking	$30	3	$90.0
Economy Lot B	$8	3	$24.0
Economy Lot C	$10	3	$30.0

Total Drive: 18.3 (Distance(mi)), 23 mins (Travel Time)

Map

Departure Location	(Street)	Los Angeles (City)	CA (State)	(Zip)
Destination Location	LAX (Airport)	Los Angeles (City)	CA (State)	(Zip)
Costs	$38.6 (Taxi Fare)	$29.49 (Park Cost)		

Fig. 2: Screen shots showing planning of flights and ground transportation

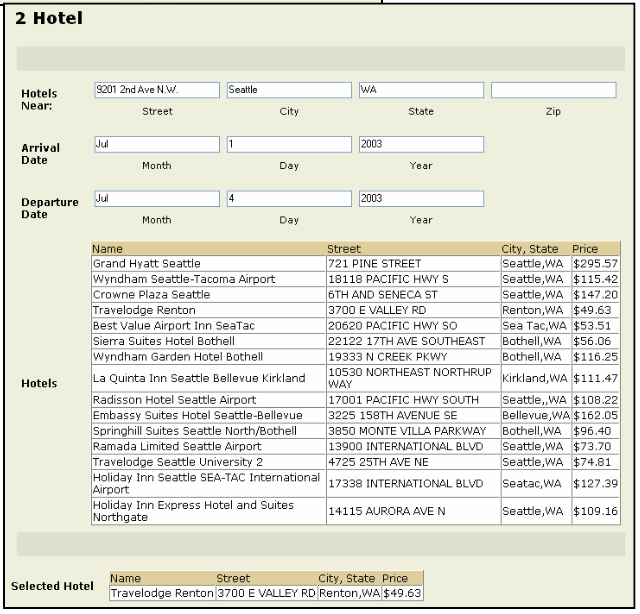


1.3 Taxi

Departure Location	SEA	Seattle	WA	
	Street	City	State	Zip
Destination Location	3700 E VALLEY RD	Renton	WA	98055
	Street	City	State	Zip
Total Drive	6.8	11 mins	$15.6	
	Distance(mi)	Travel Time	TaxiFare	

Map

Departure Location	SEA	Seattle	WA	
	Airport	City	State	Zip
Destination Location	3700 E VALLEY RD	Renton	WA	98055
	Street	City	State	Zip
Costs	$15.6	$116.19		
	Taxi Fare	Car Cost		
	6.8	11 mins		
	Distance(mi)	Travel Time		



2 Hotel

Hotels Near:	9201 2nd Ave N.W.	Seattle	WA	
	Street	City	State	Zip
Arrival Date	Jul	1	2003	
	Month	Day	Year	
Departure Date	Jul	4	2003	
	Month	Day	Year	

Hotels

Name	Street	City, State	Price
Grand Hyatt Seattle	721 PINE STREET	Seattle,WA	$295.57
Wyndham Seattle-Tacoma Airport	18118 PACIFIC HWY S	Seattle,WA	$115.42
Crowne Plaza Seattle	6TH AND SENECA ST	Seattle,WA	$147.20
Travelodge Renton	3700 E VALLEY RD	Renton,WA	$49.63
Best Value Airport Inn SeaTac	20620 PACIFIC HWY SO	Sea Tac,WA	$53.51
Sierra Suites Hotel Bothell	22122 17TH AVE SOUTHEAST	Bothell,WA	$56.06
Wyndham Garden Hotel Bothell	19333 N CREEK PKWY	Bothell,WA	$116.25
La Quinta Inn Seattle Bellevue Kirkland	10530 NORTHEAST NORTHRUP WAY	Kirkland,WA	$111.47
Radisson Hotel Seattle Airport	17001 PACIFIC HWY SOUTH	Seattle,,WA	$108.22
Embassy Suites Hotel Seattle-Bellevue	3225 158TH AVENUE SE	Bellevue,WA	$162.05
Springhill Suites Seattle North/Bothell	3850 MONTE VILLA PARKWAY	Bothell,WA	$96.40
Ramada Limited Seattle Airport	13900 INTERNATIONAL BLVD	Seattle,WA	$73.70
Travelodge Seattle University 2	4725 25TH AVE NE	Seattle,WA	$74.81
Holiday Inn Seattle SEA-TAC International Airport	17338 INTERNATIONAL BLVD	Seatac,WA	$127.39
Holiday Inn Express Hotel and Suites Northgate	14115 AURORA AVE N	Seattle,WA	$109.16

Selected Hotel

Name	Street	City, State	Price
Travelodge Renton	3700 E VALLEY RD	Renton,WA	$49.63

Fig. 3: Planning of where to stay and how to get there

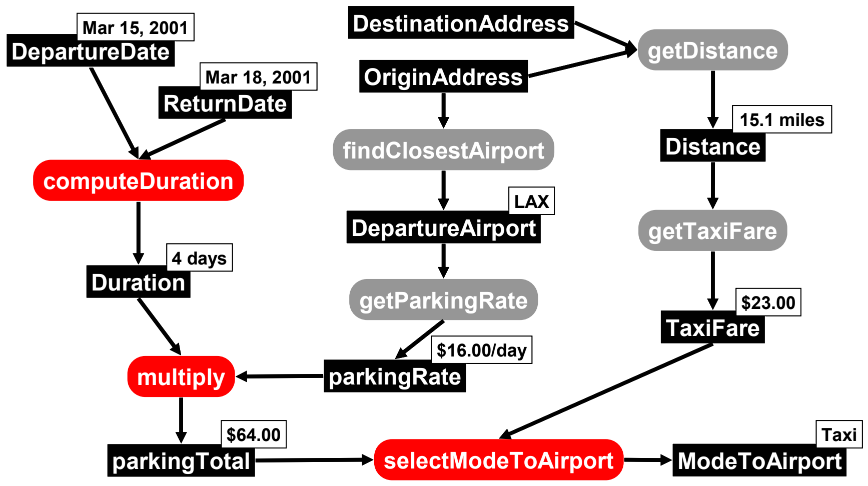


Fig. 4: Constraint network for deciding whether to drive or take a taxi

Consider the choice of driving and parking a car at the airport or taking a taxi to the airport. Which choice is more appropriate depends on a variety of factors and most people simply make these kinds of choices based on some simple heuristics, but their choices are often suboptimal because it is difficult to assemble and integrate all of the data needed to make more informed choices. Fig. 4 shows an example constraint network for making these types of decisions, where the total cost of parking a car, which is calculated based on the airport, the selected parking lot, and the number of days of travel, is compared to the cost of taking a taxi, which depends on the distance and taxi rate. Instead of simply guessing about these types of choices, the system can carefully evaluate the different choices and then make a recommendation, which can still be overriden by the user.

Besides integrating the data to help users make choices about tradeoffs, the system also makes much more information available directly to the user to help the user make better decisions. For example, instead of simply choosing arbitrarily between two flights, the system can tell the user the types of the planes, how the seats are configured, and what the on-time performance is of each of the flights. All of this information is organized into the constraint network to provide the user with the information they need to make informed decisions to plan their trip.

4 Building Agents for Monitoring Travel

As part of the Electric Elves project (Chalupsky et al. 2001; Ambite et al. 2002) we have applied our agent technologies to build a set of agents for various monitoring tasks. In the case of monitoring travel plans, this task is particularly well-suited for applying agent technology for several reasons: a) this is a fairly complicated task with many possible forms of failure ranging from flight cancellations and schedule changes to hotel rooms being given away when a traveler arrives late at night, b) there are a large number of online resources that can be exploited to anticipate problems and keep a traveler informed, and c) these tasks would be tedious and impractical for a human to perform with the same level of attention that could be provided by a set of software agents.

To deploy a set of agents for monitoring a planned trip, the user first enters the travel itinerary as described in the previous section and then specifies which aspects of the trip he/she would like to have the agents monitor. A set of information agents are then spawned to perform the requested monitoring activities. For the travel planning application, we developed the following set of agents to monitor a trip:

- An airfare-monitoring agent that tracks the current price of a flight itinerary.
- A schedule-change agent that keeps track of the published schedule for a given flight itinerary and notifies a traveler of there is any change to this itinerary.
- A flight-status agent that continually monitors the status of a flight. This agent also sends a fax to the hotel if the flight arrival is delayed past 5pm in order to ensure that the hotel room is held for the traveler.
- An earlier-flight agent that checks for flights that will depart before the scheduled flight.
- A flight-connection agent that monitors a traveler's connecting flights, checks the status and gate information of the connecting flights, and checks for earlier flights to the same destination.
- A restaurant-finding agent that locates the closest restaurants based on the traveler's GPS location.

Fig. 5 shows the messages that the various agents generated during actual use.

Ideally, a user of the system could define his/her own monitoring agents. To support this capability we have developed a system that we call the AgentWizard, which allows a user to define agents by answering a series of questions. The user can build agents that extract data, combine data from different sources, monitoring the sources for various types of changes over time, and notify the user by email, fax, or phone. This system allows users to construct a monitoring agent that is tailored to their own specific needs without requiring any programming skills. For example, someone might want an agent that monitors airfares and notifies that person the moment he/she can buy a ticket to Hawaii for less then $300. Someone else might want an agent to monitor for travel delays in their connecting airport and notify them when the average delay exceeds 30 minutes. The possibilities are endless.

The monitoring agents are defined as plans in the Theseus Agent Execution language (Barish et al. 2002). In the Web environment, sources can be quite slow and the latencies of the sources are also unpredictable since they can be caused by heavy loads on both servers and networks. Since the primary bottleneck of most agent plans on the web is retrieving data from online sources, the information requests need to be executed as early as possible. To address these issues, we have developed a streaming dataflow execution system, which is optimized for the Web environment in the

Airfare Monitoring Agent:

Airfare dropped message:
```
The airfare for your American Airlines itinerary (IAD - LAX) dropped
  to $281.
```

Schedule-Change Agent:

Schedule change message:
```
The schedule of your United Airlines flight 1287 has changed from
  7:00 PM to 7:31 PM.
```

Flight-Status Agent:

Flight delayed message:
```
Your United Airlines flight 190 has been delayed. It was  originally
  scheduled to depart at 11:45 AM and is now scheduled to depart at
  12:30 PM. The new arrival time is 7:59 PM.
```
Flight cancelled message:
```
Your Delta Air Lines flight 200 has been cancelled.
```
Fax to hotel message:
```
Attention: Registration Desk
I am sending this message on behalf of David Pynadath, who has a
  reservation at your hotel. David Pynadath is on United Airlines
  190, which is now scheduled to arrive at IAD at 7:59 PM. Since the
  flight will be arriving late, I would like to request that you
  indicate this in the reservation so that the room is not given
  away.
```

Earlier-Flight Agent:

Earlier flights message:
```
The status of your currently scheduled flight is:
# 190 LAX (11:45 AM) - IAD (7:29 PM) 45 minutes Late
If you would like to return earlier, the following  United Airlines
 flights will arrive earlier than your scheduled flights:
# 946 LAX (8:31 AM) - IAD (3:35 PM) 11 minutes Late
```

Flight-Connection Agent:

Connecting flights message:
```
Your connecting United Airlines flight 925 will depart at 9:45 PM (25
  minutes late) at gate C6.
```

Restaurant-Finding Agent:

Closest restaurants message:
```
These are the five closest restaurants from your location.
 Wingmaster's on I St, American, 1825 I St NW, 202-429-0058, $5-10,
 Lat: 38.90111, Lon: -77.04158, 0.23 miles
 …
```

Fig. 5: Actual messages sent by travel monitoring agents

following three ways. First, since the executor is based on a dataflow paradigm, actions are executed as soon as the data becomes available. Second, Theseus performs the actions in a plan in separate threads, so they can be run asynchronously and in parallel. Third, the system streams the output from one action to the next so that sequential operations can be executed in parallel.

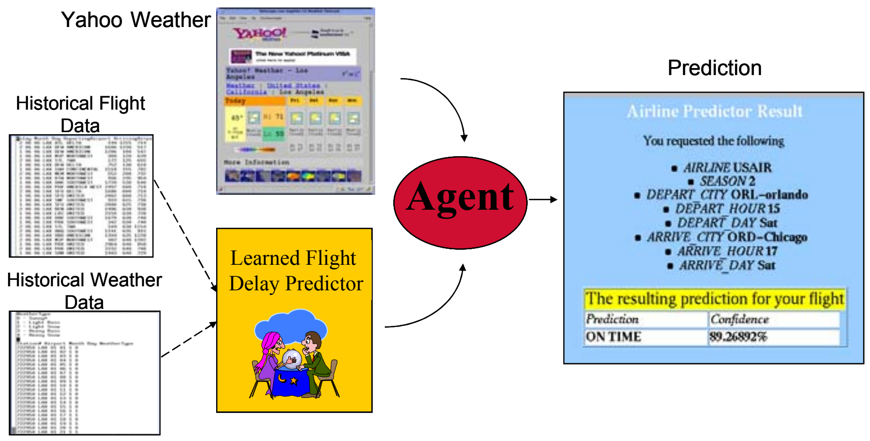
Recently we developed an approach to increase the potential parallelism in a streaming dataflow execution system. This optimization technique, called speculative execution (Barish et al. 2002; Barish et al. 2003), attempts to predict the results of an operation based on data and patterns that it has seen in the past. The predicted results can then be used to speculate about the operations that will need to be performed later in the plan. The system decides where to speculate by analyzing a plan and determining the critical paths. On these paths it then inserts a “speculate” operation, which uses input to earlier operations to predict the input to later operations. The system also inserts a “confirm” operation, which ensures that the final result is correct regardless of whether the prediction is correct. This approach to optimizing streaming dataflow plans can achieve arbitrary speedups by speculating on the speculations. If the system is able to make accurate predictions, the executor could speculate on all of the input, execute the entire plan in parallel, and then confirm all of the results.

5 Mining Online Sources to Optimize Travel Decisions

Beyond using current travel data for both planning and monitoring travel, software agents can make predictions about the world by mining online data sources. For example, an agent can make predictions about prices by mining historical price data to determine how prices change over time. Or an agent can learn which airports are most likely to experience delays or which freeways are most likely to be congested at a particular time of day. This section describes two different data mining applications that exploit the availability of online information sources.

A common problem for airline travelers is flight delays. There are a variety of causes of flight delays, but one of the most common causes is weather conditions. We developed a data mining system that predicts whether an airline flight will be delayed during the upcoming week. As shown in Fig. 6, the prediction is made by combining three sources of online information: (1) historical information about flight delays due to weather conditions (available from the Office of Airline Information, which collects information about every flight in the US), (2) historical information about the actual weather conditions at the airports where the delays occurred (available from the National Weather Service, which records weather conditions at every airport in the US at fifteen minute intervals), and current weather forecasts (available from Yahoo Weather, which provides forecasts of weather conditions at airports for up to five days in advance).

We developed a predictor that correlates the historical flight delay data with the historical weather information. The predictor was built by learning a Naïve-Bayes classifier from the historical data sources. The prediction is based on the day of the week, the time of day, the airline, the source and destination airports, and the weather forecast. The system takes as input the particular flight and date and it then checks the weather forecast for that date at the source and destination airports. By combining the forecast with the predictor, the system makes statements, such as, “Your flight is expected to be more than 15 minutes late, with a 75% confidence.”



You requested the following

- *AIRLINE* USAIR
- *SEASON* 2
- *DEPART_CITY* ORL-orlando
- *DEPART_HOUR* 15
- *DEPART_DAY* Sat
- *ARRIVE_CITY* ORD-Chicago
- *ARRIVE_HOUR* 17
- *ARRIVE_DAY* Sat

The resulting prediction for your flight

Prediction	*Confidence*
ON TIME	89.26892%

Fig. 6: Predicting flight delays from historical flight delays, historical weather data, and current weather predictions

We also developed a system called Hamlet that makes predictions about airline ticket prices and advises a traveler whether he/she should buy or wait to buy a ticket on a particular flight (Etzioni et al. 2003). We collected airfare data from online sources every three hours over a period of several months. In the study, we considered only non-stop flights between LAX and BOS and between SEA and IAD. In the collected data, we found that there was an average of more than five price changes per flight on both of the routes. On some flights there were as many as seven price changes per day. Fig. 7 shows the price changes for a ticket on a round-trip from LAX to BOS on American Airlines flights 192 and 223 on January 2 and 9, 2003. As shown in the graph, the price for this particular ticket ranges from roughly $300 to $2300. Thus, there are tremendous opportunities for savings if one can properly time the ticket purchases.

Hamlet makes recommendations about whether to buy or wait to buy at ticket by learning a predictive model of airline ticket pricing. The system uses a technique called stacking (Wolpert 1992) to combine rule learning, reinforcement learning, and time-series analysis. Each of these learning techniques has different strengths and weaknesses and we found that the combination of the three techniques produces the most accurate predictions. In order to test the efficacy of the learning, we compared Hamlet to a clairvoyant algorithm with complete knowledge of future prices. In a simulation on the real-world data, we found that Hamlet saved 341 passengers a total of $198,074 out of a possible savings of $320,572. This was 61.8% of the possible savings and provided an average savings of 23.8% for the 341 passengers for whom savings are possible. This study clearly demonstrates the potential of data mining techniques for large savings in the purchase of airline tickets. We plan to apply Hamlet to hotel room pricing to see if similar savings are possible in related industries.

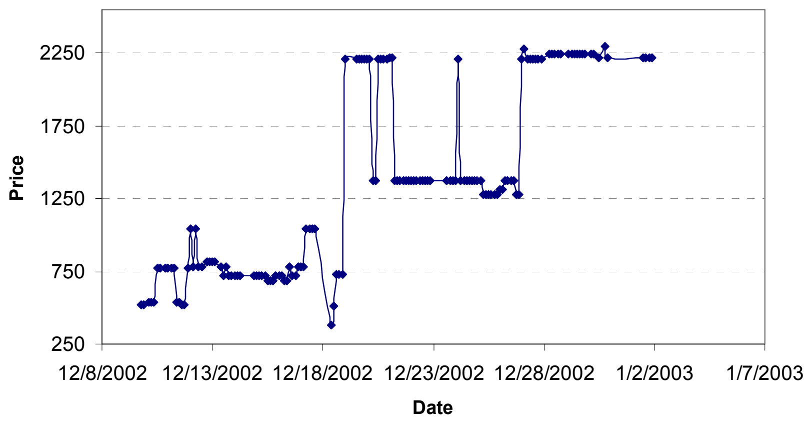


Fig. 7: Graph of prices changes for a ticket on American Airlines flights 192 & 223 for January 2 & 9, 2003

6 Related Work

Most commercial systems for travel planning take the traditional approach of providing tools for selecting flights, hotel, and car rentals in separate steps. There are two integrated approaches to this problem. The first one is a system called MyTrip from XTRA On-line. Based on personal calendar information, the system automatically produces a complete plan that includes the flights, hotel and car rental. Once it has produced a plan, the user can then edit the individual selections made by the system. Unlike the Travel Assistant, the user cannot interactively modify the plan, such as constraining the airlines or departure airport. Also, MyTrip is limited to only the selection of flights, hotels, and car rentals. The second approach being developed commercially by i:FAO Switzerland SA uses constraint satisfaction techniques to find a complete itinerary (Torrens et al. 2002). However, this system assumes that all of the relevant data has already been retrieved prior to the constraint satisfaction process and does not address the problem of how to interleave the information gathering with the constraint satisfaction to handle the enormous amount of potentially relevant information.

For monitoring a trip, there exist a number of commercial systems (such as the one run by United Airlines) that provide basic flight status and notification. However, these systems do not actually track changes in the flight status over time (they merely notify passengers a fixed number of hours before flights) and they do not notify hotels about flight delays or suggest earlier flights or better connections when unexpected events (e.g., bad weather) occur.

The work on efficient agent plan execution is similar to network query engines, such as Telegraph (Hellerstein et al. 2000) or Tukwila (Ives et al. 2002), in that they are also streaming dataflow execution systems. However, the network query engines focus on the efficient execution of of XML queries, while Thesues provides an expressive language for defining information gathering and monitoring plans. The Theseus language supports capabilities that go beyond network query engines in that it supports recursion, notification operations, and writing and reading from databases to support monitoring tasks.

The most closely related work that mines online data are comparison shopping bots that gather price data available on the web for a wide range of products. These are descendants of the Shopbot (Perkowitz et al. 1997), which automatically learned to extract product and price information from online merchants' web sites. None of these services attempts to analyze and predict the behavior of product prices over time. Thus, the data mining techniques we developed complement the body of work on shopbots. See (Etzioni et al. 2003) for a detailed comparison of our data mining techniques with other approaches.

7 Conclusion

The availability of data on the Web was the key to building the various technologies and tools described in this paper. We use the various travel-related data sources to build an integrated tool for planning a trip, to build the agents for monitoring a trip, and to mine the sources to make predictions about what was likely to happen in the future. There are many more opportunities to exploit the data available on the Internet for improving travel.

The widespread availability of travel-related information on the Web will continue to improve the experience for travelers. The tools that allow end-users to plan their own travel will continue to improve, making it possible for traveler's to optimize their trips with respect to their own preferences to a much greater extent. The use of agents for monitoring all aspects of a trip will become more widespread as the tools become easier to use and traveler's can build their own agents. And the ability to mine the data sources to minimize prices paid or better predict travel problems will continue to put pressure on the major carriers to keep their prices low and improve their service.

Acknowledgements

I want to thank my many collaborators for their many contributions to the various projects described in this paper. Ion Muslea, Steve Minton, and Kristina Lerman helped develop the wrapper learning and wrapper maintenance and repair techniques. Jose Luis Ambite, Steve Minton, and Maria Muslea all contributed to the design and implementation of the travel planner. Jose Luis Ambite, Greg Barish, Maria Muslea, Jean Oh, Snehal Thakkar, and Rattapoon Tuchinda all helped build the travel application of the Electric Elves. Rattapoon Tuchinda and Parag Samdadiya helped build the AgentWizard. Greg Barish helped develop the Theseus executor and speculative execution techniques. Steve Minton and Josh Margulis developed the flight delay predictor. And Oren Etzioni, Rattapoon Tuchinda, and Alexander Yates all contributed to the development of the Hamlet price prediction system.

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation.

References

Ambite, J. L., G. Barish, C. A. Knoblock, M. Muslea, J. Oh and S. Minton (2002). Getting from Here to There: Interactive Planning and Agent Execution for Optimizing Travel. *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*. AAAI Press, Menlo Park, CA**:** 862--869.

Barish, G. and C. A. Knoblock (2002). An efficient and expressive language for information gathering on the web. *Proceedings of the AIPS-2002 Workshop on Is there life after operator sequencing? -- Exploring real world planning*. Tolouse, France**:** 5--12.

Barish, G. and C. A. Knoblock (2002). Speculative Execution for Information Gathering Plans. *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*. AAAI Press, Menlo Park, CA**:** 184-193.

Barish, G. and C. A. Knoblock (2003). Learning Value Predictors for the Speculative Execution of Information Gathering Plans. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*. Acapulco, Mexico.

Chalupsky, H., Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ and M. Tambe (2001). Electric Elves: Applying Agent Technology to Support Human Organizations. *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*.

Crescenzi, V., G. Mecca and P. Merialdo (2001). RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *Proceedings of 27th International Conference on Very Large Data Bases***:** 109-118.

Etzioni, O., C. A. Knoblock, R. Tuchinda and A. Yates (2003). To Buy or Not to Buy: Mining Airline Fare Data to Minimize Ticket Purchase Price. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Hellerstein, J. M., M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman and M. A. Shah (2000). "Adaptive query processing: technology in evolution." IEEE *Data Engineering Bulletin* **23**(2): 7--18.

Hsu, C.-N. and M.-T. Dung (1998). "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web." *Information Systems* **23**(8): 521-538.

Ives, Z. G., A. Y. Halevy and D. S. Weld (2002). "An XML Query Engine for Network-Bound Data." *VLDB Journal* **11**(4): 380--402.

Knoblock, C. A. (2003). Deploying Information Agents on the Web. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*. Acapulco, Mexico.

Knoblock, C. A., S. Minton, J. L. Ambite, M. Muslea, J. Oh and M. Frank (2001). Mixed-Initiative, Multi-source Information Assistants. *Proceedings of the World Wide Web Conference*. ACM Press, New York, NY**:** 697-707.

Kushmerick, N. (1997). Wrapper Induction for Information Extraction, Ph.D. Thesis, Department of Computer Science and Engineering, University of Washington.

Lerman, K., C. A. Knoblock and S. Minton (2001). Automatic Data Extraction from Lists and Tables in Web Sources. *Proceedings of the IJCAI 2001 Workshop on Adaptive Text Extraction and Mining*. Seattle, WA.

Lerman, K., S. N. Minton and C. A. Knoblock (2003). "Wrapper Maintenance: A Machine Learning Approach." *Journal of Artificial Intelligence Research* **18**: 149-181.

Muslea, I. (2002). Active Learning with Multiple Views, Ph.D. Thesis, Department of Computer Science, University of Southern California.

Muslea, I., S. Minton and C. A. Knoblock (2000). Selective sampling with redundant views. *Proceedings of the 17th National Conference on Artificial Intelligence*.

Muslea, I., S. Minton and C. A. Knoblock (2001). "Hierarchical Wrapper Induction for Semistructured Information Sources." *Autonomous Agents and Multi-Agent Systems* **4**(1/2).

Perkowitz, M., R. B. Doorenbos, O. Etzioni and D. S. Weld (1997). "Learning to Understand Information on the Internet: An Example-Based Approach." *Journal of Intelligent Information Systems* **8**(2): 133-153.

Torrens, M., B. Faltings and P. Pu (2002). "SmartClients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems." *Constraints (Special Issue on Constraints and Agents)* **7**: 49-69.

Wolpert, D. H. (1992). "Stacked Generalization." *Neural Networks* **5**(2): 241-259.