Information Technology & Tourism, Vol. 6 pp. 195–210
Printed in the USA.

1098-3058/04 $20.00 + .00

www.cognizantcommunication.com

AN AUTOMATED ITINERARY PLANNING SYSTEM FOR HOLIDAY TRAVEL

SIMON DUNSTALL, MARK E. T. HORN, PHILIP KILBY, MOHAN KRISHNAMOORTHY, BOWIE OWENS, DAVID SIER, and SYLVIE THIEBAUX

CSIRO Mathematical and Information Sciences

This article describes a prototype travel recommender system called the Electronic Travel Planner (ETP), which prepares travel itineraries for tourists. The system is driven by models of a traveler's preferences and requirements, and makes reference to databases containing information pertaining to tourism and travel products. Its main tasks are to select destinations for the traveler to visit, to decide which tours or attractions are to be taken, and to compose a detailed itinerary linking up the chosen components. These tasks entail difficult optimization problems, which the prototype addresses by means of an heuristic problem-solving framework. Computational tests confirm the effectiveness of the methods used, and suggest that an automated approach will be feasible in full-scale travel planning applications.

Key words: Traveler recommender systems; Journey planning; Routing; Scheduling; Optimization

Introduction

The Electronic Travel Planner (ETP) is a prototype Internet-based system designed to plan travel itineraries for tourists. The research leading to development of the system was carried out in 1999–2001 by the authors and their colleagues in collaboration with Viator Inc., a company that provides travel information, technology and distribution services to the travel industry (Viator: http://www.viator.com/). The project was supported by a substantial grant under the Australian government's START R&D scheme. The prototype ETP system comprises kernel code for an Internet server: although attention has been given to implications with respect to the user interface and the Internet connection, these elements have not yet been implemented. Negotiations aimed at a commercial implementation of the system are currently under way.

A central aim of the ETP project was to develop technology to support new ways of planning a holiday, by capturing critical aspects of a traveler's needs and desires and exploring the implied possibilities better than could be achieved by existing means. A travel planning system should have value as a tool in the hands of travel agents, and potentially beyond that, of individual travelers. The benefits for travelers and for the tourism industry would include greater satisfaction with the travel planning process, and with the travel experience itself. These aims are clearly

Address correspondence to Mark E. T. Horn, GPO Box 664, Canberra ACT 2601, Australia. Tel: +61 2 6216 7054; Fax: +61 2 6216 7111; E-mail: mark.horn@csiro.au

consonant with the concept of a travel recommender system. Ricci (2002) reviews recent developments in this field, covering several types of choice models and interactive means of determining user preferences. Fesenmaier et al. (2002) discuss research in travel destination choice models, information search and processing, and human–computer interaction. They focus on the styles of travel and decision making preferred by individual travelers, and they propound an "adaptive" style of recommender system, based on dialogue with the system user.

By contrast with the approaches outlined above, the authors have regarded the components of a travel plan—including the choice of destinations and travel products, and the timing of travel—as elements of a mathematical problem, which is to be solved without human intervention. In particular, the aim of ETP as a research activity has been to show that "fully automated" itinerary-planning systems can generate "sensible" itineraries that are well aligned with the expressed travel needs and desires of an abstracted user. This *itinerary planning* approach is distinct from approaches that choose just some of the elements of a holiday, such as one or more travel products, or one or more places to visit (i.e., "destinations"); instead, the entire holiday is planned in space and time, between a specified starting point and end point (usually the same place, e.g., the traveler's home). Such an approach apparently has not been attempted before, due to the complexity of the task and doubts concerning scalability to handle large travel databases (Ricci, 2002). It is emphasized that an itinerary planning system need not be regarded as a deterministic "black box" from whose decisions there is no appeal; on the contrary, such a system is conceived here as a tool to aid users in exploring travel possibilities (system usage is discussed further in the conclusion to this article).

The construction of an itinerary planning system poses technical challenges of several kinds. In the first place, such a system requires a coherent and comprehensive scheme for modeling the requirements, preferences, and interests of a traveler. Secondly, it entails optimization problems of very considerable technical difficulty. Thirdly, the overall optimization task is not limited to a single unambiguous objective, such as minimizing costs, but instead seeks a satisfactory or "balanced" outcome with respect to the various, and often heterogeneous, criteria specified by the user.

The ETP project commenced with a market survey that included discussions with Australian tourism authorities at national and state levels, major travel organizations and travel providers, and travel agents. The survey identified a number of Internet-based tools designed to capture information about a user's preferences and return information about travel products to match those preferences. These "travel-matching" systems were focused generally on "atomic" choices within narrowly defined product domains, such as hotel accommodation in a given place (e.g., British Hotels Reservation Centre [BHRC]: http://www.bhrc.co.uk/), or ski resorts in Europe (Ski-Europe: http://www.ski-europe.com/; TripleHop: http://www.triplehop.com/ie/product_demos/tripmatcher.html). A few systems did address compound travel planning tasks, notably with respect to the construction of multiple-leg itineraries connecting a pair of points by road (e.g., Travelmate: http://www.travelmate.com.au/MapMaker/Mapmaker.asp), or by air or rail (e.g., Internet Travel Network: http://www.itn.com/; TheTrainLine: http://www.thetrainline.com/; see also Lee, 2001). Even so, the latter were limited by a focus on a single transport mode, and by a "least-cost, point-to-point" paradigm more suited to business travel than to tourism.

Despite this lack of commercial precedent, several aspects of the ETP project have been foreshadowed in previous research, notably in the literature on the Traveling Salesman Problem (TSP). In its elementary form the TSP poses the task of finding a shortest Hamiltonian cycle of a network (i.e., the shortest closed path traversing all vertices of the network; see Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1985). In extensions such as the Orienteering Problem, a cycle is to be made of only a subset of the vertices, which are to be selected on the basis of their scores with respect to a measure of attractiveness to maximize the total score; in addition, time windows may be imposed on individual visits and on the tour as a whole (Kantor & Rosenwein, 1992).

Another task arising in an itinerary planning context is to find the shortest path between a given pair of vertices of a network. This has been the subject of an extensive literature, commencing with the work

of Dijkstra (1959), and extended subsequently to the planning of multiple-leg journeys in passenger transport networks with timetabled services (e.g., Dial, 1967; Tong & Richardson, 1989). This shortest-path research was intended primarily for use in a simulation and modeling context, but it also has direct application to real-time journey planning and booking services, such as those arising in the context of intercity travel. For intraurban travel, journey planning is seen as a means of facilitating the usage of urban passenger transport services (see also Infopolis: http://www.ul.ie/~infopolis), and is represented by several systems (MVA: http://www.mvagroup.com/brochures/mjp.pdf; Opcom: http://www.opcom.com/au/iptis/), installed in several cities (e.g., TransPerth: http://www.transperth.wa.gov.au/; Go-Ventura: http://www.goventura.org/transit.htm/). The objective in such systems is generally to minimize a compound measure of generalized cost or a surrogate such as elapsed time, total fares, or the number of interchanges. Further research (Horn, 2003, 2004) has propounded a wider journey planning framework encompassing demand-responsive modes (e.g., taxis) as well as regular timetabled services, different styles of travel (e.g., early departure vs. early arrival, and discretionary activity in mid-journey), and a more nuanced representation of cost (e.g., to represent a traveler's mounting impatience during long intervals of waiting).

There is also a rich literature on the handling of user preferences (for an extensive review, see Fesenmaier et al., 2002). In informal practice these are usually expressed in categorical terms (e.g., "I want that," "I must have that"), but for mathematical purposes they must be translated from the categorical to numerical domains, where the mathematical problem is to make decisions involving numerous and perhaps conflicting preferences. Problems of this kind have been studied extensively in the literature on decision theory and multicriteria optimization, and Loban (1997) describes the application of a preference-weighting scheme to the selection of a single package tour. Also of interest is the travel planning system developed by Low, Cheng, Wong, and Motwani (1996), which selects and connects daily itineraries within a given city on the basis of user preferences.

The ETP system has been informed by the various approaches outlined above, but represents a substantial step beyond any previous system that the authors are aware of. It uses heuristic techniques to obtain good results in "reasonable" time; that is, not necessarily with near-real-time responsiveness, but with some confidence that the execution times for instances of problems will be no more than a few minutes. The present account is concerned primarily with the preference modeling and travel planning aspects of the system. However, the importance of an effective user interface is acknowledged, and this aspect of system design has been a recurring concern throughout the ETP project, especially with reference to the formulation of the preference modeling scheme.

The remainder of this article is organized as follows. The next two sections outline the main informational elements of ETP, and the scheme by which preferences with respect to travel components are modeled in the system's inputs. The following section gives an outline of the system architecture, and is followed by descriptions of the main travel planning modules. Some example itineraries produced by the system are then discussed, along with experience with respect to computational performance. The article concludes with a discussion of avenues for further research.

Itinerary Elements and Their Attributes

A travel itinerary comprises a sequence of locationally referenced travel products of three main kinds, namely *tours*, *lodgings*, and *transport*. A tour is typically a packaged activity that may extend over a few hours or several days, finishing possibly at a different place from where it started, and with generic attributes describing its content (e.g., "historic buildings" or "skindiving"). Similarly, a lodging (e.g., a hotel) has attributes that prospective travelers may or may not find attractive, such as "swimming pool" or "vegetarian food." A *tour* or *transport* product may convey travelers from one place to another, and may have attributes of interest such as "mode *m*" (e.g., air or rail), "frequent flier scheme *f*," or "airline *a*."

Thus, a list of possible attributes is predefined for each type of travel product. The definition of attributes in this respect is loose, in that some attributes are mutually exclusive (e.g., "one star," "two star," etc., in the case of lodgings) and some are not. Simi-

larly, an attribute may refer to an intrinsic characteristic of a travel product (e.g., a travel mode), or merely to the presence of a particular facility (e.g., skindiving).

Information about travel products and locations is stored in a relational database. Each location is either a particular place or a set of places (e.g., a "region" containing other locations in the usual spatial sense). Each travel product has a unique database identifier, together with information about its start location, duration, cost, and availability (dates and times); a product (e.g., a certain class of room at a certain hotel) can thus be read from the database as a set of instances, one for each time period when the product is available. A set of attributes is recorded for each product: for example, a given lodging may have attributes such as "three star, Golden Chain, swimming pool, wheelchair access, in-house restaurant, vegetarian food." The database also includes specialized information pertinent to the different types of travel product, such as the terminal points of each transport route and tour, the set of locations served by each such point (e.g., for the case of Dallas–Fort Worth airport), and various lower bounds with respect to cost and time. The latter is exemplified by the least cost of lodgings at a given location, and is kept up to date by means of maintenance software that is run whenever the database is updated.

Specification of Requirements and Preferences

The requirements stipulated by an ETP user include locations where the itinerary is to start and end (possibly the same place), a time window, upper and lower budget limits, and the number of adults and children in the party (for costing purposes). An example is given below.

> Request for: **3** travel plans, for **2** adults and **3** children.
> Travel from **Paris** on **1 September 2003** to **Marseilles** on **22 September 2003**.
> Budget range: **$US6000** to **$8000**

In addition, the user may have preferences with respect to the various types of travel product and the locations to be visited. As indicated earlier, the ETP database includes a predefined set of attributes and a predefined set of locations; the user's preferences are expressed as a set of descriptors, which may be applied to all or some of these attributes and locations. The preference descriptors are Mandatory, AtLeastOnce, Desired, Undesired, Permitted, and Forbidden; they are mutually exclusive, in that the user can specify no more than one descriptor for any given attribute or location.

Suppose that one of the preference descriptors is applied to an attribute *f* (e.g., skindiving) of products of type *t* (e.g., tours). Then the possible values for the descriptor are defined by reference to a potential itinerary *I*, as follows.

- Mandatory: *f* must be present in all products of type *t* in *I*. For example, if an attribute "Skindiving" is specified as Mandatory for tours, every tour (if any) in *I* must have skindiving. Similarly, if an attribute "Accept pets" is specified as Mandatory for lodgings, every lodging (if any) in *I* must accept pets.
- Forbidden: *f* must be absent from all products of type *t* in *I*.
- Permitted: the presence or absence of *f* in products of type *t* has no bearing on the quality of *I*. This is the default valuation, which applies to all attributes for which no preference is explicitly specified.
- AtLeastOnce: *f* must be present in at least one product of type *t* in *I*.
- Desired: as many as possible products of type *t* with attribute *f* are to be included in *I*. Even so, the absence of *f* will not vitiate *I*. For example, the inclusion of even one product with attribute *f* might be impossible for budgetary reasons.
- UnDesired: as few as possible products of type *t* with attribute *f* are to be included in *I*. Even so, the presence of *f* will not vitiate *I*. For example, it may be necessary to include an UnDesired tour that has attribute *f* because that tour also has another attribute *f′* marked as AtLeastOnce, and no other tours are available to satisfy this preference.

This set of descriptor values has been designed to reflect the orientation and "hardness" of a travelers' desires, and the "quantity" of the objects desired. Although more comprehensive preference protocols

are possible (e.g., with additional descriptors such as DesiredAtLeastOnce or ExactlyOnce, or with continuous-valued variables), the simpler approach described here has the advantage of conceptual intelligibility, notably with respect to user interface design.

Some adjustments to the above scheme are required for a coherent application of the preference descriptors to locations. For example, applying the Mandatory descriptor to a given location would prevent travel anywhere else. Furthermore, the provision for "compound" locations (i.e., "regions," as discussed earlier) requires the enunciation of additional conventions. It seems intuitively reasonable that a preference for a given location L can be satisfied by a visit to one or more of the places in L, but there are potential difficulties with respect to spatial structure (a location may include other locations) and scope (the set of visitable locations is potentially large). To resolve these difficulties, the possible preferences with respect to a location L are interpreted as follows, again with reference to an itinerary I.

- Mandatory: not used.
- AtLeastOnce: I must include a visit to [at least one location in] L.
- Desired: I should include a visit to [at least one location in] L.
- UnDesired: I should not include a visit to [any locations in] L.
- Forbidden: I must not include any visit to [any locations in] L.
- Permitted: I may include visits to [any location in] L.

This locational preference scheme is augmented by the specification of a single Boolean parameter, PermittedDefault, to indicate whether or not visits to locations for which no locational preferences are specified are to be treated as Permitted or Forbidden. For example, a default of Forbidden would be appropriate for a traveler who wishes to restrict search to locations for which explicit preferences are recorded.

A set of locational preferences has the effect of defining positive or negative preferences (AtLeastOnce, Desired, or UnDesired) for some locations, while labeling some other locations as Permitted. Such a set is given below, for illustrative purposes.

PermittedDefault = false
Provence: Permitted
Nice: Forbidden
Avignon: Desired
Aix-en-Provence: UnDesired
Arles: AtLeastOnce
Languedoc: Desired

Here Provence and Languedoc are adjacent regions and the other locations are Provençal towns. Intuitively, it seems that the user wishes to visit the Provence–Languedoc area with the exception of Nice, so that the visitable region is {Provence ∪ Languedoc – Nice}; Languedoc is generally preferred over Provence, but a visit to Arles is definitely required, and other (soft) preferences are expressed with respect to Avignon and Aix. An interpretation of this kind depends on rules for intersecting cases; for example, to specify a preference for Provence and another preference for a location Nice lying within Provence is to make Nice an exception with respect to the general rule stated for Provence. ETP embodies comprehensive rules (encompassing *identity*, *inclusion*, and *overlap*) to handle such cases.

System Architecture

ETP is designed as a server application, and makes use of a relational database. Access to the server is by way of an XML document containing requirements and preferences specified by a user, and XML is also used for returning results to the user (a client-side graphical user interface has not yet been implemented, as indicated in the Introduction to this article). The main functional components of the system are shown in Figure 1. Each incoming user request is processed through the following stages.

1. The user request is decoded in the *DataPrepare* module, and relevant data are extracted from the database. The outcome is a subset of parts of the database relevant to the request, for reference by other modules. *DataPrepare* checks the request for prima facie feasibility, and fa-

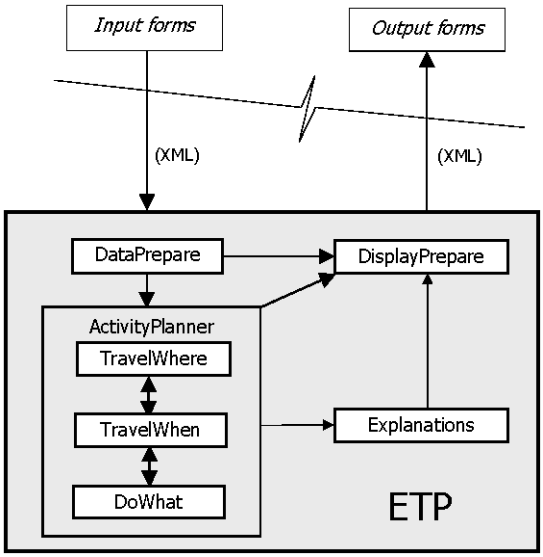

Figure 1. Architecture of ETP.

cilitates efficient system operation by caching data items in memory, thus avoiding excessive overheads with respect to database access.

2. If a prima facie inconsistency or error is discovered in Stage 1, a message is returned immediately to the user.
3. The user request is passed to the *ActivityPlanner* module, which has several components. *TravelWhere* generates a sequence of locations to be visited; *TravelWhere* passes this locational plan to *TravelWhen*, which makes a timetable for travel, and then calls the *DoWhat* component to generate a set of tours to be taken. Taken together, the locational plan, the timetable, and the tours constitute a fully fledged itinerary.
4. Step 3 is repeated until (if possible) *n* itineraries have been generated, where *n* is the number of itineraries specified in the user request. Upon completion, the itineraries are annotated by the *Explanations* module and returned to the user via *DisplayPrepare*. Itineraries, explanatory messages, warnings, and other output texts are accumulated in internal data structures at all stages of the travel planning process, and the function of *DisplayPrepare* is mainly to compose these texts as an XML document for return to the user.

The *DataPrepare* Module

In the *DataPrepare* module a user request is checked, relevant data are obtained from the database, and initial filtering is carried out with respect to user requirements. The main stages here are as follows.

1. The user request is checked for completeness and internal inconsistency, and basic data structures are set up for attribute and location preferences. This involves initial queries on the database to obtain data dictionary entries for the attributes and locations explicitly specified by the user. Each location name specified by the user is resolved as one or more "atomic" locations, and intersecting regions are handled by application of the interpretive rules mentioned in the preceding discussion of locational preferences.
2. An object called a *DataCache* is constructed by reference to the user request. This object is an in-memory repository containing sections of the database that are relevant to the current user request. The cached data objects are of three main kinds.
   - *Locations.* The locations of interest comprise the start and end points specified in the user request, together with the other locations specified explicitly as Desired or AtLeastOnce. If the PermittedDefault parameter for locations is true, the list of locations may be augmented in the course of queries for tours (see below), such that every location at the start or end of a tour is represented in the cached set of locations.
   - *Tours.* If the user has specified any Mandatory tour attributes, a database query is made for tours with these attributes; otherwise, queries are made for tours with attributes specified as AtLeastOnce and Desired. The selection in each case excludes tours that have Forbidden attributes or that are unavailable during the period specified in the user request; also excluded are tours that commence at, end at, or make visits to Forbidden locations. In a final check, if the cached set of tours does not include at least one tour to satisfy each tour

attribute specified as AtLeastOnce by the user, the request is evidently infeasible and search is terminated.

- *Lower bounds.* A matrix of lower bounds is compiled for travel between each pair of locations in the location cache. The bounds for a given origin–destination pair refer to the shortest travel time and least cost of travel between the pair. Similarly, the details of costs obtained from each location include lower bounds on lodging costs.

3. In addition to the information discussed above, the *DataCache* serves as a repository for several kinds of data that are obtained from the database on demand. This "lazy-accumulation" technique avoids multiple querying with respect to any given item, and eliminates query overhead altogether for items that are never required by the other parts of the system. The main additional data types are described below.
   - *Travel routes.* The *DataCache* can be interrogated to obtain details of all transport routes (i.e., scheduled services) available between a given pair of locations, subject to the time window and Mandatory and Forbidden travel attributes specified in the user request.
   - *Lodgings.* Another *DataCache* query yields details of all lodgings at a given location, subject to the time window and Mandatory and Forbidden lodging attributes specified in the user request.

The *ActivityPlanner* and *Explanations* Modules

When the *ActivityPlanner* is invoked, *DataPrepare* module has identified the travel products that are available at the time of the trip and that should be considered for an itinerary. The role of the *ActivityPlanner* is to match the user request with a subset of available travel products, and to construct a time schedule specifying the times when these products are used, with additional periods to allow for travelers' rest and preparation for travel.

The *ActivityPlanner* generates answers to the fundamental travel-related questions of "where, what, and when." It selects the locations visited by the traveler (where), decides which tours are to be taken by the traveler (what), and determines a detailed timetable for the chosen transport and tourism activities (when). In broad terms, the *ActivityPlanner* addresses these questions in that order, and progression through the itinerary-generation process is associated with increasing itinerary detail, less emphasis on selecting travel products, and more emphasis on scheduling discrete activities. The *ActivityPlanner* treats itinerary generation as a constrained optimization problem. This means that the problem of creating an itinerary is expressed mathematically in terms of constraints and an objective function. The best itinerary is then that which satisfies all of the constraints and which maximizes the value of the objective function. To this end, the *ActivityPlanner* transforms relevant information from the user request into a set of constraints and objective–function components, which are then addressed by a variety of procedures. Although some of the procedures are "exact," the overall framework is approximate, in that it offers no guarantee of finding the best of all possible itineraries.

The constraints can be classified as being either attribute based or logistically based. As an example of the latter, a traveler must have a lodging every night, must spend time resting before and after each interval of travel, and may make use of only one travel product at a time. The attribute-based constraints refer to the preference descriptors. As indicated in the preceding section, the *DataPrepare* module has applied Mandatory and Forbidden to include some travel products in the problem addressed by the *ActivityPlanner*, while excluding others. The AtLeastOnce descriptor implies a "hard" constraint, in that a feasible itinerary must have at least one product with the associated attribute. The other descriptors, namely Desired, Undesired, and Permitted, are treated as soft constraints contributing to the objective function, rather than imposing strict limitations on the contents of a travel plan.

To quantify the impacts of the preference descriptors, each attribute is associated with an attribute response function representing the penalty or benefit obtained from varying quantities of the attribute (i.e., from varying numbers of products possessing the attribute). The response functions are conceptually similar to utility functions, but ignore cost (see below). Some simple examples of attribute response functions are illustrated in Figure 2, in which Q_k denotes the quantity of some attribute k, and $V_k(Q_k)$

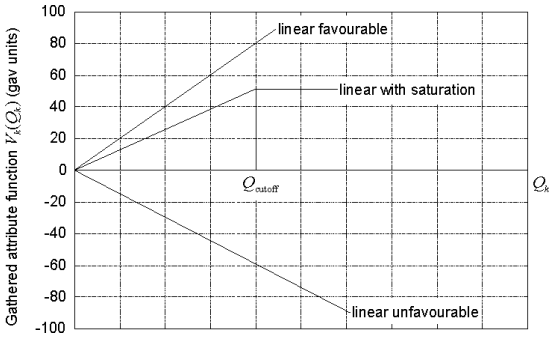

Figure 2. Simple attribute response functions.

signifies the corresponding value of the attribute response function. The attribute response functions are attribute specific in functional form and defining parameters, in that different sets of functions are predefined for "soft preferences" expressed with regard to travel, tours, and locations.

It is instructive at this point to observe that in our mathematical model we: (a) assume that all attribute quantities are nonnegative integers; (b) assume that the value of an attribute response function is dependent only on the quantity of a single attribute; and (c) further assume that each attribute response function is monotonic (i.e., it is either nonincreasing or nondecreasing with attribute quantity). The assumption of monotonic attribute response functions is not onerous and this degree of generality is attributable to the local-search-inspired approach that characterizes the system's itinerary-generating heuristics. At various points during itinerary generation the quantity of each attribute in a complete or partial solution is determined, and the solution is assessed through an explicit calculation of the objective function. The results of this assessment are used to guide subsequent decisions in a search for improved solutions. We assume that attribute response functions are monotonic so that we can tailor our postassessment decision making to exploit this property.

It is notable in this respect that although the *ActivityPlanner* can support the full generality of the mathematical model, that generality is not available in the current ETP implementation, due to the assignment of attributes to products in a Boolean rather than continuous fashion. In particular, a travel product is either associated with a given attribute, or it is not, yielding an attribute quantity of 1 or 0, respectively. This restriction is present in the markup of products with attributes in the database, and reflects the practical difficulty of obtaining the relevant information with respect to numerous attributes and travel products. The result is that it is not possible at present to apply a fine-grained emphasis to particular attributes within or between the various types of travel product.

To apply the attribute response functions pertinent to a current problem, their values are summed by the *ActivityPlanner* to obtain a quantity called the gathered attribute value (GAV). The GAV reflects the total "enjoyment value" of the itinerary. As a benchmark for comparison with the GAV, the *ActivityPlanner* obtains at the outset a utility function based on the budget and preference information in the user request. The utility function expresses the GAV value that the traveler might expect for a total itinerary cost, and is defined as the sum of a linear function and an exponential "barrier function" that becomes increasingly significant as the itinerary cost approaches the traveler's upper budget limit (three utility functions with different sets of parameters are shown in Fig. 3). For a given GAV and total cost, the overall objective function value is the difference between the GAV and the utility function value for that cost, and the task is to maximize this difference. It should be noted that the objective function value is a notional value only; for example, a negative difference between the GAV and the utility function value does not indicate a substandard itinerary.

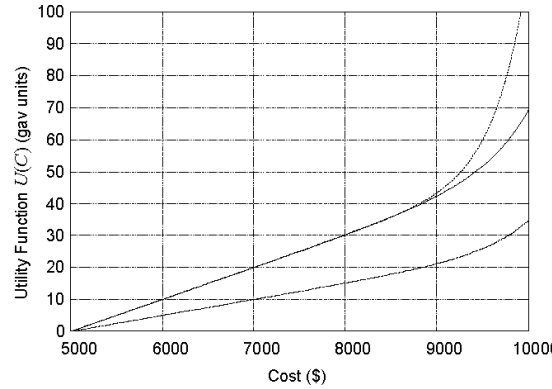

Figure 3. Three examples of utility functions.

An itinerary is composed in three scheduling layers. The *locational* layer determines the location occupied by the traveler, or travel on a transport service, at each instant during the period of the itinerary. The *lodgings* layer selects a lodging for each night that is not spent on a transport service. The *activities* layer schedules day tours and travel between interlocation travel intervals.

The "where" question is addressed by the *TravelWhere* component. *TravelWhere* uses an algorithm that combines TSP tour scheduling (using simulated annealing) with decision tree searching. The root-node sequence consists of the user-specified first and last locations. Constraint programming is used to weight and rank candidate locations, and the locations are inserted one or two at a time into the sequence of locations. The TSP algorithm minimizes expected travel cost (Carpaneto, Dell'Amico, & Toth, 1995). *TravelWhere* defines the locations that are to be visited and the order in which visits are to occur. This amounts to what may be called a sketch itinerary. The *TravelWhen* and *DoWhat* modules attempt to refine each "promising" sketch itinerary into a complete itinerary, by selecting transport services between locations and determining the time of travel (the "when" question), and scheduling tours and selecting lodgings (the "what" question). As an example of the trade-offs made in these respects, an itinerary that specifies a substantial amount of time to be spent in a location with inexpensive lodgings and tours will save cost, yet allocating more time to an interesting location may result in the gathering of greater attribute value over the same time interval.

A sketch itinerary is refined by a nested series of algorithms that operate on specific parts of the problem. The outermost algorithm selects transport services and tours from the *DataCache*. A priority list of these services and tours is constructed initially on a heuristic basis. Items are then selected from the priority list and introduced into the itinerary. The introduction of an item spawns a process termed *location scheduling*, where the travel products assigned to each location are scheduled according to their availability timetables. The methods used during location scheduling include bisection search, auction algorithms, and critical path algorithms. Lodgings are selected as part of the location scheduling process, and "free time" is allocated to locations to maximize the objective function value in the neighborhood of the current solution.

The outermost algorithm executes iteratively, adding one or more products at each iteration. The priorities of items in the priority list are dynamically updated using measures of their objective–function influence and their temporal impact on the itinerary. Priorities are also strongly influenced by a factor that is computed using an *ActivityPlanner* component that implements constraint handling logic. This component of the system models a selection of the problem constraints, and analyzes their state periodically. This state determines the criticality of the inclusion or exclusion of a particular product, and computes a corresponding weighting factor. The constraint handling modules (there are in fact two of them in the system) are also used to influence decisions that have the potential to affect the feasibility or desirability of an itinerary.

A full specification of the mathematical model and the algorithms used in the ETP system is beyond the scope of this article. This is due to their complexity. There are, for example, no less than 14 distinct submodules in the system, each of which deals with a subproblem resulting from a decomposition of the master problem. Each submodule implements a specialized algorithm or method, and higher-level submodules will decompose their subproblem further and iteratively call lower-level submodules. The following discussion focuses on one particular lower-level subproblem.

At a certain stage of the *TravelWhen–DoWhat* solution process we identify a "trial" set of travel products excluding lodgings. Each of these travel products is associated with a starting location and possibly different finishing location, and *TravelWhere* has sequenced all of these locations based on a cost-of-travel measure. We have also identified by this stage those travel products that will be used to move the traveler from one location to the next in the sequence. We refer to such travel products as *displacing activities*. Each travel product is viewed as a set of *timetabled services* (i.e., pairs of available starting and finishing dates and times). The timetabled service information is obtained from the *DataCache* (see preceding section).

The immediate goal at this point in the solution process is to either select a service for each of the travel products in the "trial" set to create a fully fea-

sible itinerary, or to determine that a feasible itinerary using this "trial" set does not exist. A secondary goal is to select services for displacing activities such that the value of the GAV function is maximized: in effect, this equates to distributing "free" time in the itinerary amongst the locations. For the latter purpose (on which we will not dwell on here) we seek to create *forward-loaded* and *backward-loaded* itineraries. A forward-loaded (backward-loaded) itinerary schedules interlocational travel at as early (late) as possible, and thus spends the maximum possible time at the last (first) location. The formation of a forward-loaded itinerary proceeds as follows.

Let there be m locations in the sequence. The first location L_1 in the sequence is taken and the set of products P at this location is identified. We choose the earliest possible service of the displacing activity arriving at this location, and the latest possible service of the displacing activity departing from this location. From these services we determine an interval of time $[T_{lo},T_{hi})$ over which activities can be undertaken at the location, given suitable periods of postarrival recuperation and predeparture preparation. We then solve what we term the DoWhat Decision Problem (DWDP): "is there a selection of services, one from each product in P, such that all services are completed within the interval $[T_{lo},T_{hi})$?"

Our chosen method of solving the DWDP is based on partitioning $[T_{lo},T_{hi})$ into time slots and mapping the service periods of each activity to these time slots. In general, a service will require one or more adjacent time slots. Let the set of time slots be T, and let a Boolean variable x_{ij} equal 1 if and only if service j of product $i \in P$ is utilized. An integer programming formulation of the constraints of the DWDP is as follows:

$$\sum_j x_{ij} = 1 \quad \forall i \in P$$

$$\sum_{i,j} x_{ij} \le 1 \quad \forall (i,j) \in \theta_t, \forall t \in T$$

$$x_{ij} \in \{0,1\}$$

where $(i,j) \in \theta_t$ if and only if service j of activity i overlaps time slot $t \in T$.

In the prototype ETP system we solve the DWDP via an auction algorithm (e.g., see Bertsekas, Castanon, & Tsaknakis, 1993). By reference to "conventional" auction algorithm terminology, time slots equate to "objects" and activity instances (or parts thereof) relate to "people." We wish to assign one or more "adjacent" objects to each person (i.e., the assignment is asymmetric). The objects are adjacent if one time slot follows the other temporally. A service of an activity is scheduled feasibly if and only if it is assigned to all of the time slots it overlaps with in time. We will denote the required set of time slots for service (i,j) by τ_{ij}. During the auction each time slot has a current price, p_t, and this price starts at zero.

We begin a bidding phase in the auction algorithm by determining the subset of activities P' that do not have a feasibly scheduled service. For each activity i in P' we find the service (i,j) for which the value of $|\tau_{ij}|.\max_{t \in \tau_{ij}} \{p_t\}$ is a minimum, and place a bid of $\max_{t \in \tau_{ij}} \{p_t\} + \varepsilon$ on each of the time slots in τ_{ij}, where ε is a suitably defined value. In the subsequent assignment phase we award time slots one-by-one to the highest-bidding activity: this may mean that some but not all of the required time slots for a service are awarded to an activity.

After the assignment phase the solution is evaluated to determine if a feasible schedule has been produced, and if it has, the auction algorithm terminates and flags a "success." Otherwise, further iterations of bidding and assignment are attempted until one of two termination criteria are reached:

- A time slot receives a bid that is in excess of a given bid limit. We have set this bid limit to be equal to half the total number of services in the instance, unless this value lies outside a certain range, in which case the nearest of the bounds of this range is chosen. In the prototype system we have used the range [25,100].
- The algorithm does not find a feasible solution after a certain number of iterations. The maximum number of iterations of the algorithm is set to equal the total number of timetabled services, as long as this value lies between defined lower and upper limits. In the prototype system we have used limits of 50 and 200 iterations, respectively.

The given instance of the DWDP is presumed to be infeasible if either of these termination criteria is

met within the auction algorithm. Clearly this is a heuristic approach to solving the DWDP. We have not as yet been able to determine whether the auction algorithm is guaranteed to find a feasible solution, if one exists, in polynomial time. In our computational testing we have observed that the auction algorithm performs adequately. In a future research paper we will report in more detail on experiments with this algorithm and study alternative methods (including exact methods) that might perform better, at least for certain special cases of the problem.

The auction algorithm that heuristically solves the DWDP is nested within a bisection search. This bisection search varies the value of T_{hi} through the selection of alternative services for the displacing activity that is used to depart the location in question. This bisection search allows us to determine the minimum amount of time required at the location given a starting time of T_{lo}. After carrying out the DWDP bisection method for location L_1, we then repeat the process for each location L_i ($i \leq m$) in turn. This establishes the forward-loaded itinerary. The construction of the backward-loaded itinerary adheres to a similar process.

We now turn our attention briefly to the constraint handling modules. There are two constraint handling modules (CHMs) in the prototype system, one associated with *TravelWhere* and the other with *TravelWhen* and *DoWhat*. Both are bespoke implementations that have a functionality that is very similar to a typical Constraint Logic Programming over Finite Domains (CLP-FD) solver, except that we implement only domain restriction and propagation (i.e., we do not encode or execute a labeling strategy). A CHM maintains an internal stack to facilitate branching and backtracking. The optimization heuristics that utilize a CHM directly control the branching and backtracking carried out by this CHM.

Within the itinerary generation process a CHM's role is to attach weights to travel products, where these weights represent a measure of "importance" of the travel product. This "importance" is computed relative to both "hard" and "soft" constraints (e.g., AtLeastOnce and Undesired preference levels, respectively). Travel products are assigned a higher importance if they are promising candidates for inclusion into a partial itinerary that is currently under construction. A CLP-FD solver implements one or more variable and value ordering heuristics (such as first-fail or regret) that choose the next variable and value to be assigned. A CHM adopts the spirit of these heuristics, but rather than implementing the assignments itself, it provides the weights to the optimization heuristics. An optimization heuristic combines CHM weights with its own state knowledge to make a more informed assignment. The optimization heuristic updates the CHM state as required, based on assignments, backtracks, and so on.

The *Explanations* module generates textual information to supplement each itinerary produced by the ETP. This information is presented in the form of natural-language sentences, which are composed on an ad hoc basis by components of the *DataPrepare* and *ActivityPlanner* modules, and then are merged together by the *Explanations* module. Explanations are concerned with the reasoning underlying the itinerary planning process (see the examples in the following section); in particular, they provide reasons why certain elements are present in an itinerary returned to the user, and why certain (requested) elements are absent.

For example, if an itinerary includes a visit to a location to which the Desired descriptor was not applied, the *Explanations* module will explain that such a visit is necessary to satisfy certain (nominated) Mandatory or AtLeastOnce tour attributes. Conversely, if a required location is not visited (so making the itinerary infeasible), a reason (e.g., insufficient time or insufficient budget) will be given as to why that location could not be included. This sort of information is particularly important when restrictions in a user request force the inclusion of unattractive (e.g., Undesired) features in an itinerary; sometimes the restrictions may even preclude the construction of a feasible itinerary, and the *ActivityPlanner* can produce only sketch itineraries. In such cases explanations may provide insight into how the restrictions can be relaxed, thus helping the user to respecify a request to obtain a better result.

Tests of ETP

Tests of ETP are reported below, based on data provided by Viator (see the Introduction). The focus is on the ways in which the system can be used to explore available travel options, highlighting the

role of the explanation module discussed above, and then summarizing some tests of computational performance.

Illustrative Scenario

The scenario discussed below concerns a fictitious person called Martha Green. We suppose that she will be visiting relatives in Dublin in November and wishes to take the opportunity to see other parts of Europe. She has 2 weeks to spare and expects to spend between US$2000 and $5000 ($5000 is the very upper limit). When on holiday, she enjoys seeing and learning new things, as well as sightseeing from trains and ferries. However, she has no taste for adventure tours and despises golf. She likes to stay in comfortable accommodation, the more plush and pampering the better, and after previous trips knows that she should request a room with a bar fridge.

The scenario is designed to show how the ETP can be used in practice. It comprises a series of user requests supposedly made by Martha, and makes use of a database relating to Western Europe. Although the scenario is fictitious, the results are real. They have been obtained by applying ETP to the specified scenarios, and are reported below (in smaller type), literally as output from the system.

Initial Preferences

With the help of her travel agent, Martha formulates her travel requirements and preferences and enters them in a "front end" to the ETP. The contents of Martha's request are shown in Table 1. The request is sent to the ETP and three itineraries are returned.

Table 1
Martha's Initial Travel Requirements, as Seen by ETP

Field	Entry
CustomerID	524XEL3
CustomerName	Martha Green
NumItineraries	3
BudgetLow	2000.00
BudgetHigh	5000.00
TripStart	Dublin
TripFinish	Dublin
DateFrom	1 November 2002
DateTo	14 November 2002
NumAdults	1
NumChildren	0
PlanObjective	MinCost
LocationPreferences	
PermittedLocations	Algarve & S.Portugal
DesiredLocation	Tuscany & Florence
DesiredLocation	London
TourPreferences	
AtLeastOnceTour	Day Cruise
DesiredTour	Rail Day Tour
DesiredTour	Learning and How To
DesiredTour	Sightseeing Tour
UndesiredTour	Adventure
ForbiddenTour	Golf Package
LodgingPreferences	
DesiredHotelStars	5
DesiredHotelStars	4
UndesiredHotelStars	2
ForbiddenHotelStars	1
DesiredLodgingType	Heritage Manors
DesiredLodgingType	Guest House
UndesiredLodgingType	Self Catering Cottages
DesiredLodging	Refrigerator

Initial Preferences: Itinerary 1

The explanatory report and the main travel details are shown below for the first (best) itinerary generated by ETP.

Explanation, Initial Preferences: Itinerary 1
This is one of the best itinerary we found. It complies with all your basic requirements, but satisfies your additional preferences only partially. The itinerary includes Algarve and Lisbon in order to go on Day Cruise tours. We cannot explain why Tuscany and Florence has not been included in this itinerary. It may appear in further itineraries. The itinerary does not include locations at which Guesthouse lodging or Heritage Manors lodging is available.
Cost Summary: Total Cost $2453, Tour Cost $659, Travel Cost $1589, Lodging Cost $205
All Costs $US
Travel Ref Eurail13b from Dublin 1/11/2002 9:30 to Algarve 3/11/2002 3:24, Cost $793
Accommodation Hotel Alisios, Algarve, from 3/11/2002 to 4/11/2002, 4 Star, Refrigerator, Cost $25.
Day Cruise Top Cat Cruise, from Algarve 3/11/2002 9:30 to Algarve 3/11/2002 13:29, Cost $30
Accommodation Hotel Alisios, Algarve, from 4/11/2002 to 5/11/2002, 4 Star, Refrigerator, Cost $25
Day Cruise Grottoes Cruise, from Algarve 4/11/2002 9:30 to Algarve 4/11/2002 17:29, Cost $37
Accommodation Hotel Alisios, Algarve, from 5/11/2002 to 6/11/2002, 4 Star, Refrigerator, Cost $25
Day Cruise Top Cat Cruise and lunch, from Algarve 5/11/2002 9:30 to Algarve 5/11/2002 17:29, Cost $44

Accommodation Hotel Alisios, Algarve, from 6/11/2002 to 7/11/2002, 4 Star, Refrigerator, Cost $25
Day Cruise Blue Cruise, from Algarve 6/11/2002 9:30 to Algarve 6/11/2002 17:29, Cost $35
Accommodation Hotel Alisios, Algarve, from 7/11/2002 to 8/11/2002, 4 Star, Refrigerator, Cost $25
Day Cruise Fun Cruise Guadiana River. Algarve 7/11/2002 9:30 to Algarve 7/11/2002 17:29, Cost $25
Travel Ref Eurail09a from Algarve 8/11/2002 9:30 to London 9/11/2002 20:09 Cost $572
Accommodation Forum Hotel, London, from 10/11/2002 to 10/11/2002, 4 Star, Refrigerator, Cost $16
Sightseeing Tour British Airways London Eye, from London 10/11/2002 9:30 to London 10/11/2002 9:54, Cost $45.
Accommodation Forum Hotel, London, from 10/11/2002 to 11/11/2002, 4 Star, Refrigerator, Cost $16
Learning and How To Tour Classic Coach Tour of London by "Vintage" Coach, from London 11/11/2002 9:30 to London 11/11/2002 12:29, Cost $18.
Accommodation Forum Hotel, London, from 11/11/2002 to 12/11/2002, 4 Star, Refrigerator, Cost $16
Learning and How To Tour Legoland from London 12/11/2002 9:30 to London 12/11/2002 17:29 Cost $29
Accommodation Forum Hotel, London, from 12/11/2002 to 13/11/2002, 4 Star, Refrigerator, Cost $16
Rail/Day Tour, Day Trip to Paris, from London 13/11/2002 9:30 to London 13/11/2002 17:29, Cost $396
Accommodation Forum Hotel, London, from 13/11/2002 to 14/11/2002, 4 Star, Refrigerator, Cost $16
Travel Ref Eurail30a from London 14/11/2002 17:30 to Dublin 14/11/2002 18:19, Cost $US 224

Initial Preferences: Itinerary 2

The second itinerary generated in response to Martha's request has a total cost of $2271, compared with $2453 for Itinerary 1. It is similar to the Itinerary 1 in its early stages, but subsequently includes a trip from Algarve to Florence with two nights' stay at the hotel Adriatico from 11/11/2002 to 13/11/2002, and then a flight from Florence to Dublin on 13/11/2002. ETP provides the following explanation in this case.

> This is one of the best itinerary we found. It complies with all your basic requirements, but satisfies your additional preferences only partially. The itinerary includes Algarve in order to go on Day Cruise tours. The itinerary does not include locations at which Guesthouse lodging or Heritage Manors lodging is available.

Initial Preferences: Itinerary 3

The third itinerary has a total cost of $2159. It involves an initial stay of 3 days in London, followed by a flight to Lisbon and 1 day there, the remainder of the allotted time being spent in Algarve. The explanation in this case is as follows.

> This is one of the best itinerary we found. It complies with all your basic requirements, but satisfies your additional preferences only partially. The itinerary includes Algarve and Lisbon in order to go on Day Cruise tours. We cannot explain why Tuscany and Florence has not been included in this itinerary. It may appear in further itineraries. The itinerary does not include locations at which Guesthouse lodging or Heritage Manors lodging is available.

Adjusting Preferences

Martha decides that she is not especially interested in Algarve and Southern Portugal. She changes her preferences so that "Tuscany and Florence" and "London" remain Desired locations, but "Algarve and Southern Portugal" is now Undesired. All the other requests and preferences are kept the same.

The ETP is invoked to generate three new itineraries. The first has a total cost of $2637 and involves flying from Dublin to London, spending 5 days in London followed by 2 days in Lisbon and then 3 days in Florence, and finally flying back to Dublin from Florence.

The second itinerary has a total cost of $2674 and involves flying from Dublin to Florence, spending 2 days in Florence followed by 2 days in Lisbon, 2 days in Algarve, 4 days in London, and finally flying back to Dublin from London. The explanation includes the following statement.

> [This itinerary] complies with all your basic requirements, but satisfies your additional preferences only partially. The itinerary includes Algarve and Lisbon in order to go on Day Cruise tours.

Adjusting Preferences Again

Martha decides now that she doesn't want to go to Portugal at all, even if there are plenty of cruises available there, so she changes her preferences so that "Algarve and Southern Portugal" and "Lisbon" are ruled out (Forbidden locations), "Tuscany and

Florence" *must* be visited (AtLeastOnce), and "London" and "Edinburgh" are Desired.

Again the ETP is used to obtain three itineraries. The first has a total cost of $2716 and involves flying from Dublin to Florence, spending 4 days in Florence, 4 days in London, and then 4 days in Edinburgh, and finally flying from Edinburgh to Dublin. The accommodation is in four-star hotels in Florence, and in five-star hotels in London and Edinburgh. A number of day trips are scheduled but no day cruises, despite the AtLeastOnce request made in this respect (see Table 1). The explanation accompanying the itinerary runs as follows.

> This is one of the best itinerary we found. It complies with all your basic requirements, but satisfies your additional preferences only partially. We could not find Day Cruise tours anywhere which are available within your time range and matching your requirements. The itinerary does not include locations at which Guesthouse lodging is available.

This itinerary through Florence, London and Edinburgh appeals to Martha, particularly given the accommodation selected in London and Edinburgh.

Computational Behavior

Running time can be an issue when using the current version of the ETP. An extended program of computational testing has been undertaken on a Sparc 10 workstation, with a substantial degree of realism obtained through the use of a database constructed from over 250KB of textual travel data. Solutions for many of the test problems could be obtained in a matter of seconds, but some requests required several minutes to process. For the example discussed above, the observed computation times ranged from 15 to 45 seconds depending on the user request and natural sampling variation (due in part to the use of randomized local search methods). These results indicate that the approach taken is feasible from a computational point of view, but suggest that further development will be required to achieve performance suitable for an online, Web-based application of the technology.

Conclusion

The research reported in this article has shown how a wide range of user preferences with respect to holiday travel can be expressed in mathematical form, and has established a procedural framework capable of handling those conditions. The computational tests carried out with the ETP prototype confirm that an automated approach to itinerary planning is achievable in real-world applications. The main research aim of the ETP project is thus fulfilled.

In derivatives of the ETP system intended solely for commercial applications it may be desirable to provide opportunities for a user to participate more directly in the planning process; for example, by specifying sections of an itinerary in advance and leaving the ETP travel planning modules to fill in the gaps, and conversely, by modifying an itinerary constructed by the travel planning modules. This will pose challenges for user interface design, and will require further elaboration of the problem-solving scheme described in this article. More broadly, we may envisage a range of travel planning systems extending from manual to "advisory," semiautomated, and fully automated: given the availability of systems at several points along this range, a comparative investigation of usability would be of considerable interest.

Further development of travel planning systems such as ETP will be conditioned generally by the close relationship between technical "form" and substantive "content." A concern in this respect is that the database should be sufficiently rich in information to allow credible itineraries to be constructed. This requires the devotion of substantial effort to the tagging of database elements with respect to the presence of the various attributes in which users may be interested. In addition, the contents of the database should not be unduly restricted by commercial considerations; in particular, the travel planner should be seen as a service in its own right, not merely a device for adding value to a database.

The presence of a commercial bias or emphasis in database content has a bearing even on the "kernel" itinerary planning system described in this article. In particular, the ETP system currently makes no provision for the inclusion in a travel itinerary of noncommercial or "discretionary" activities: a morning at a beach, a visit to an art museum, a shopping expedition, an afternoon's rest, a call upon a friend. Given the prevalence of such activities in practice,

it is desirable from the point of view of system users that they should be accommodated in future travel planning systems. This could be done by defining discretionary activities as variants of tours, with looser time specifications.

The ETP system might also be extended to handle self-drive travel modes such as cars or campervans. A distinguishing feature of these modes that will need to be recognized in any extension is their more or less continuous availability, by contrast with the discrete instantiation of timetabled services. Self-drive modes also raise logistical issues, such as the participation of vehicle rental outlets in an itinerary, and the need to park and return to one's vehicle when self-drive and timetabled modes are combined in a journey.

Acknowledgments

The support and encouragement of Rod Cuthbert and Jordan Digby of Viator, Inc., is gratefully acknowledged. The primary instigator of the ETP project was Graham Mills of CSIRO. CSIRO participants in the project included John Colton, Simon Dunstall, Mark Horn, Michael Kearney, Phil Kilby, Mohan Krishnamoorthy, Ian Mathieson, Bowie Owens, Sylvie Thiebaux, and Paul Thomas. The authors also wish to thank two anonymous reviewers for their helpful and incisive comments, which have considerably improved the content and readability of this article.

Biographical Notes

Simon Dunstall joined CSIRO Mathematical and Information Sciences in 2000, subsequent to completing a Ph.D. in Engineering Science at the University of Melbourne. His research interests include algorithm and system development for decision making in scheduling, timetabling, and supply-chain management domains. At CSIRO he has been involved in several projects that have developed and applied such research to the solution of commercial and industrial problems.

Mark E. T. Horn is a research scientist with CSIRO Mathematical and Information Sciences. A background in architecture (B.Arch-Hons, UNSW 1973) and city planning (MCP, U of PA 1979) led to his involvement in software development for land planning and in the mining and petroleum industries. Since joining CSIRO in 1987, he has engaged in research focused on algorithms and decision support systems for spatial planning, transport, and scheduling applications.

Philip Kilby is a research scientist with CSIRO Mathematical and Information Sciences. His research interests are principally in combinatorial optimization, and he has worked on such problems in the transport industry, nurse rostering, geographical information systems, travel planning, vehicle simulation, and high-speed transit vehicle optimization. He enjoys the challenge of studying a process and ensuring the best possible use is made of all resources.

Mohan Krishnamoorthy is a Science and Industry Manager at CSIRO. His responsibilities include research strategy, setting research and business directions and goals as well as engaging with industry. He has a background in Operations Research (decision support provision through optimization and simulation techniques). Mohan is currently managing a portfolio of research disciplines including optimization, image analysis, data mining, and bioinformatics.

Bowie Owens completed a Bachelor of Computing (Computer Science) at Monash University before joining CSIRO's OR Group. He has been involved in a number of research projects including rostering, travel planning, and languages for optimization. A recurring theme of many of these projects is that of constrained scheduling and constraint representation.

David Sier is a senior scientist at CSIRO with 10 years of experience in applying Operations Research techniques to the development of large-scale optimization models for practical applications. One of his major interests is the transfer of management science methods to organizations with easy-to-use, robust computer-based algorithms for handling complex decision-making processes.

Sylvie Thiebaux is a researcher in the Research School of Information Sciences & Engineering at the Australian National University. Prior to that, she was a scientist with CSIRO, where she took part in the ETP project. Her research interests are in artificial intelligence, and include planning, diagnosis, search, and intelligent transport systems applications.

References

Bertsekas, D. P., Castanon, D. A.; & Tsaknakis, H. (1993). Reverse auction and the solution of inequality constrained assignment problems. *SIAM Journal of Optimization, 3*(2), 268–299.

Carpaneto, G., Dell'Amico, M., & Toth, P. (1995). Exact

solution of large-scale, asymmetric traveling salesman problems. *ACM Transactions on Mathematical Software, 21*(4), 394–409. Available at: http://portal.acm.org/citation.cfm?doid=212066.212081

Dial, R. B. (1967). Transit pathfinder algorithm. *Highway Research Record* 205, 67–85. National Research Council (USA), Highway Research Board.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik, 1*, 269–271.

Fesenmaier, D., Pan, B., Gretzel, U., Hwang, Y., Grabler, K., Zins, A., & Mazanec, J. (2002). *Tourist decision model report.* Project IST-2000- 29474. http://dietorecs.itc.it/PubDeliverables/D2.2-V1.0.pdf

Horn, M. E. T. (2003). An extended model and procedural framework for planning multi-modal passenger journeys. *Transportation Research B, 37*(7), 641–660.

Horn, M. E. T. (2004). Procedures for planning multi-leg journeys with fixed-route and demand-responsive passenger transport services. *Transportation Research C* (forthcoming).

Kantor, M. G., & Rosenwein, M. B. (1992). The orienteering problem with time windows. *Journal of the Operational Research Society, 43*(6), 629–635.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D.B. (Eds.). (1985). *The travelling salesman problem.* New York: John Wiley & Sons.

Lee, F. (2001, May). Twenty-first century journey-planning. *Operational Research Society Newsletter, 365*. Also at: http://www.uk.cgey.com/services/or/docs/21_century_journ_plan.pdf

Loban, S. R. (1997). A framework for computer-assisted travel counselling. *Annals of Tourism Research, 24*(4), 813–834.

Low, B. T., Cheng, C. H., Wong, K. F., & Motwani, J. (1996). An expert advisory system for the tourism industry. *Expert Systems With Applications, 11*(1), 65–77.

Ricci, F. (2002). Travel recommender systems. *IEEE Intelligent Systems, 17*(6), 55–57.

Tong, C. O., & Richardson, A. J. (1984). A computer model for finding the time-dependent minimum path in a transit system with fixed schedules. *Journal of Advanced Transportation, 18*(2), 145–161.