Holiday Scheduling for City Visitors[1]

Beatriz López

Institut d'Informàtica i Aplicacions
Universitat de Girona, Spain
blopez@eia.udg.es

Abstract

This paper presents an approach to holiday scheduling which schedules the activities a tourist can enjoy in a city through an Internet service available 24 hours a day, 7 days a week (at information desks in airports, train stations, etc.). Holiday scheduling implies a constraint satisfaction problem, with the particularity that the amount of constraints in this problem is not critical, but the number of activities is huge. Hence, the methodology proposed is a combination of case-based reasoning and techniques for solving constraint satisfaction problems. The benefits of this combination are particularly useful in scheduling real life problems, and is suitable in terms of the particularities of the domain. A case study at the city of Girona is provided.

Keywords: City tourist, museums, case-based reasoning, constraint satisfaction, scheduling.

1 Introduction

Holiday scheduling consists of organizing, for a given period of time, a set of activities that a tourist wants to do at a given destination. Holiday scheduling can be looked at with different degrees of detail. Some researchers concentrate on the elaboration of packages in which the main idea is to allocate different tourist spots and attractions to a set of days (Soo and Liang, 2001; Ricci and Werthner, 2001; Huang and Miles, 1995). Others concentrate on the selection of a given hotel for a given destination (http://www.cbr-web.org/ [July, 2002]). While others look at particular tourist services in a given region (Blanzieri and Ebranati, 2000). This work is in line with this latter approach.

The aim of this project is to develop a 24-hour Internet service (at information desks in airports, train stations, etc.) that will enable a tourist arriving at anytime in a city to schedule his/her stay. The user selects, from a list, the set of activities he/she wants to

[1] The research was supported by the Spanish CICYT project TAP1999-1086-C03-02.

enjoy. There are several kinds of activities: visiting museums, monuments, art exhibitions, temporary cultural events (such as festivals, markets and performances), sightseeing, parks and sports facilities among others). Some of the activities are constrained by timetables such as museums. Furthemore, some also require an entrance fee, so the user should have a budget for them. Each place to visit requires a minimal period of stay, estimated by those responsible for the activities. For example, a visit to the City Museum is estimated as lasting 2 hours. Once it is checked that the user has enough time and a big enough budget to do the selected activities, the system schedules them, based on the experiences of past users using a combination of Case-Based Reasoning and Constraint Satisfaction Problem Solving techniques. In addition, temporary events in the city, as well as other activities not selected by the user, are proposed to the user in order to fill in time gaps.

This paper is organized as follows: Section 2 deals with the formulation of the holiday scheduling problem in terms of constraint satisfaction. Section 3 provides the methodology. Since this paper is exploratory, the results given in section 4 are case studies regarding the scheduling of activities for a visitor to the city of Girona (Spain). Finally, section 5 provides some conclusions and the details of related work.

2 Problem formulation

Holiday scheduling consists of organizing, for a given period of time, a set of activities that a tourist wants to do at a given destination. Every activity has a duration d_i which expresses the minimal staying period. For the sake of simplicity, the time required to get to the location of the activity is included in the minimal staying period.

Every activity has an associated cost c_i in Euros (resource). The cost can be 0 € for free entrance activities (for example, a sightseeing activity). No precedence relations between activities exist. However, some of the activities are only allowed at a given set of times (timetables) that may depend on the day of the week or a particular date, as well as on the season. For example, the Cathedral Museum is closed on Monday, and is open from Tuesday to Saturday from 10:00 to 14:00, and from 16:00 to 19:00. On Sundays and holidays it opens in the morning: 10:00-14:00. In addition, the Museum has three different timetables: one from October to February, another from March to June, and yet another from July to September. Hence, to handle all these temporal variables, we associated each activity with a timetable array, T_i, of month and day dimension. Each cell of the array, $L_{j,i}$, is a list of available timetables for the corresponding month (j) and day (i) of the activity, $L_{j,i} = ((init_1, end_1),...(init_n, end_n))$. Some $L_{j,i}$ values may be empty such as, for example, the ones corresponding to closing days as wells as non-existing days (i.e.

February 30th). Thus, for the Cathedral Museum, we have $L_{Jan,1} = ((10,14))$ and $L_{Apr,9} = ((10,14),(16,19))$, both dates, January 1st and April 9th, being a Tuesday.

In addition to the activities, the user also specifies constraints: the duration of his/her stay and his/her budget. Duration and budget establish the following constraints:

- The sum of the minimal staying period of all activities must not exceed the holiday period of the user. That is: $duration \geq \sum_i d_i$.
- The sum of the cost of all activities must not exceed the user's budget. That is: $budget \geq \sum_i c_i$.

If the above constraints are not satisfied, there is no schedule possible. So the user is forced either to remove activities or to change the constraints before starting the problem solving.

3 Methodology

To solve the holiday scheduling problem the methodology proposed is based on the combination of Constraint Satisfaction Problem solving techniques and Case-Based Reasoning (CBR). On the one hand, holiday scheduling can be formulated as a Constraint Satisfaction Problem (CSP), in which we need to assign time slots to a set of activities without violating any tourist constraints (budget, trip duration). Constraint Satisfaction Problems are hard to solve, however. On the other hand, Case-Based Reasoning is a particular example of a Machine Learning Technique in which past experiences (problem solutions) can be used to solve new problems, making CSP solving efficient by providing solutions at one shot.

With regard to the problem at hand, i.e., holiday scheduling, the advantage of following this integrated approach is twofold. First, CBR provides complete past schedules in which other activities, in addition to the ones selected by the user, can be suggested. The number of activities in a city is huge, and some of them are temporary events, or depend on the season, etc. Thus, a similar past case can provide alternatives to the user who may not be aware of them. And second, CSP is not started from scratch, but from a past solution. Since the domain is not critical (not too constrained), the effort required for adapting a past solution to the current problem following a CSP is feasible.

3.1 The CBR component

According to the standard methodology, any CBR has four main steps: retrieval, reuse, revise and retain. In the retrieval phase, old solutions from the case base are recovered in order to solve a new problem. In the reuse phase, the solution of a past

problem is used to solve a new problem. It is precisely during this phase when CSP techniques are used in order to adapt the old solution to the new problem. Then in the revise phase, the solution for the case is checked, and finally, it is learned in the retain phase.

Case Base. The case base is a collection of solved CSPs. A case is a CSP in which the following parts are distinguished (see Figure 1 for an example):

- Identification: a numerical value
- Constraints to be provided by the user: duration of the stay, budget, and temporal constraints (season, starting date, starting week day).
- Selected activities: activities the user wants to do.
- User profile: age, sex, marital status, profession and hobbies.
- Solution of the case: the final price, the starting time, the final time, and the scheduling. The latter consists of an ordered list of tuples representing the sequence of activities. Each tuple (A, S, D, C) contains the information of the activity name (A), the starting time (S), the duration (D), and the cost (C).

Case Retrieval. Case retrieval is based mainly on a three-step process: activity matching, constraint matching and selection. First of all, past cases containing the same activities are recovered, $\{r_i\}$. Then, a constraint matching process is applied in which a similarity degree for each case r_i is computed, based on the constraints. No case is rejected. The constraints must be satisfied in the final scheduling, but previous approximations to these constraints may be useful when producing the new one. Finally, the one that best matches the probe (highest similarity degree) is selected for adaptation in the reuse phase.

Activity matching is based on the number of activities of the probe P present in the past experience C_i:

$$Sim_a(P, C_i) = \frac{|Act(P) \cap Act(Ci)|}{|Act(P)|} \quad (1)$$

where Act (X) are the activities of a case. Constraint matching is based on a similarity function that follows an Ordered Weighted Average (OWA) (Yager, 1988):

$$Sim_C(P, C_i) = \sum_{j=1}^{n} w_j * sim_{\sigma(j)}(cons_j(P), cons_j(C_i)) \quad (2)$$

where: n is the number of constraints; $cons_j(X)$ is the j-constraint of case X; $sim_{\sigma(j)}(x_j, y_j)$ is the similarity between the two j-constraints (see (López, 2002)); $\sigma(j)$ is a permutation of the values 1…n so that $sim_{\sigma(j-1)}(x_{j-1}, y_{j-1}) \geq sim_{\sigma(j)}(x_j, y_j)$ $\forall i = 2,\ldots,n$; and w_i is the relevance of each constraint in the similarity. The OWA is suitable for computing similarity, because we can order the different constraint

similarities based on the degree of their similarity. In our model we have n=5, and we set w_1=0.35, w_2=0.25, w_3=0.17, w_4=0.13 and w_5=0.1.

Case 1
Duration: 3
Budget: 200
Temporal: (Season June), (Date 22-June-2002), (Week day Saturday)
Activities: Jewish Museum, Cathedral Museum, City Museum
Profile: 23, Female, Single, Computer Science, (Traveling, Sports, Photography)
Solution: (Price 190.85•),(Starts at10h),(Ends at 20h)
((Jewish Museum, 10h, 3h, 1.8•),(Lunch, 13h, 1h, 9•),(Beach, 14h, 6h, 0•),
(Dinner, 20h, 2h, 15•),(Pub, 22h, 2h, 20•),(Disco, 24h, 3h, 20•),
(Hotel, 3h, 7h, 45•),(Cathedral Museum, 10h, 2h, 4.25•),
(Sightseeing, 12h, 1h, 0•),(Lunch, 13h, 1h, 7•),(Beach, 14h, 6h, 0•),
(Dinner, 20h, 2h, 15•),(Pub, 22h, 2h, 20•),
(Disco, 24h, 3h, 20•),(Hotel, 03h,7h, 45•),(City Museum, 10h, 2h, 1.80•),
(Sightseeing, 12h, 1h, 0•),(Lunch, 13h, 1h, 7•),(Beach, 14h, 6h, 0•)

Case 2
Duration:2
Budget: 100
Temporal: (Season February),(Date 2-Feb-2002),(Week day Saturday)
Activities: Art Museum, Cathedral Museum, Cinema Museum, City Museum
Profile: 23, Female, Single, Computer Science, (Travel, Sports)
Solution: (Price 90.05•),(Starts at 10h),(Ends at 19h),((Art Museum, 10h, 3h, 3•),
(Lunch, 13h, 1h, 7•),(Sightseeing, 14h, 2h, 0•),
(Cathedral Museum, 16h, 2h, 4.25•), (Devessa, 18h, 2h, 0•),
(Dinner, 20h, 2h, 12•),(Pub, 22h, 2h, 8•),(Hotel, 24h, 10h, 45•),
(Cinema Museum, 10h, 3h, 2•),(Lunch, 13h, 1h, 7•),
(Sightseeing, 14h, 2h, 0•), (City Museum, 16h, 2h, 1.80•)

Fig.1. Case base used in the example.

Case Reuse. In the reuse phase, we adapt the past case solution to the new problem. On a few occasions we will have an identical case, so the solutions are carefully elaborated to guarantee that constraints are satisfied. The procedure for adapting the past solution is based on CSP techniques and is explained in the next section.

Case Revise. Once the solution has been produced, it is given to the user. All solutions are valid ones from the point of view of a CSP, that is, all constraints are satisfied. However, the solution might not be the best one. Moreover, the user may not like the activities suggested for filling in time, or the solution in general. Thus, he/she can accept it or modify it. That is to say, it is the user who evaluates the outcomes.

Case Retain. The new solution is not always retained in memory. The storage of new solutions depends on the adaptation effort. If the solution comes from an adaptation process in which more than one access to memory has been performed (i.e. more than one case has been used to solve the case), then the new case is stored. It will probably be useful in other situations, and avoid wasting time in accessing the case base. Otherwise, the case will not be stored, since the adaptation process is simple and not time consuming. Avoiding the storage of all cases, we keep the case base to a reasonable size.

3. 2 The CSP component

The procedure for adapting the retrieved solution to the current problem is as follows:

1. Copy the past solution to the new case
2. Remove activities not selected by the user and those that do not take place in the corresponding season
3. Check for the following inconsistencies:
   3.a. Duration exceeded
   3.b. Budget exceeded
   3.c. Timetable violation according to calendar holidays and day of the week
   3.d. Overlapping activities
4. Solve one inconsistency and go to 3 until no more inconsistencies hold
5. Complete the set of selected activities
6. Add activities for filling up time according to user profile

Solving duration inconsistencies. This situation may occur when, for example, the user has specified a 1-day duration and the retrieved case is a 2-day duration. In such a situation, only the scheduling of the activities requested by the user are transferred, together with the components that make up the minimal set (lunch, dinner, hotel). Remember that at the beginning of the problem solving, it is guaranteed that the user will have enough time to perform all the activities he/she has selected.

Solving budget inconsistencies. Budget violation may occur as a consequence of the constraint matching process, since recovering cases with similar budgets is allowed. However, it does not mean that the case has no solution. Added activities, such as lunch and dinner, are revised in order to suit the budget.

Solving timetable inconsistencies. Timetable violation may occur when the scheduling retrieved does not exactly match either the date or the week day. Hence, certain activities planned for one day cannot be performed. To solve this inconsistency, a new case that provides an alternative scheduling for the inconsistent activities is recovered.

Solving overlapping activities. To solve this inconsistency the following procedure is applied (following the Dynamic Minimum Conflicts Algorithm developed by Lisa Purvis (Purvis, 1997)):

1. Choose, from among the conflicting activities, the one that is least constrained.
2. Search the case-base for alternative schedulings for that activity.
3. If one is found, return the new scheduling.
4. Otherwise, search for available time gaps in the current schedule compatible with the activity. If one is found, then re-schedule it.
5. Otherwise, leave the conflicting activity chosen in step 1 unchanged and proceed by selecting the next least conflicting activity for re-scheduling.
6. Continue step 2 until all overlapping activities are solved.

Complete the set of selected activities. If an activity selected by the user was missing (it was not among past cases), it is added to the scheduling either by exchanging it with activities to fill up time or by placing it in empty slots.

Adding activities to fill up time. The addition of activities to fill up time is performed according to the user profile, following recommender system approaches, such as the ones in (González et al. 2002).

4 Case study

In this section, the methodology presented in this paper is illustrated by means of an example in the city of Girona. For this purpose, consider the cases that make up the case base, shown in Figure 1. Assume that a visitor to the city selects the following activities (new problem P_1): Art Museum, Cinema Museum and City Museum. At the first step of the retrieval phase, the activity matching formula is applied for every case in memory (equation 1) and gives the following results: $Sim_a(P_1, C_1) = 1/3$, $Sim_a(P_1, C_2) = 1$. Case C_2 is therefore retrieved as the most similar case. No further similarity computations are performed. The next step is the adaptation process.

The past solution is transferred to the new case. Since Season is the same in both cases, all activities scheduled in C_2 are still valid for P_1. So, no activities are removed.

Next, the inconsistencies are checked: the current case is for a duration of 1 day whereas the retrieved case is for 2 days, so, the activities selected by the user are only transferred as a provisional solution:

(Art Museum, 10h, 3h, 3€), (Lunch, 13h, 1h, 7€),
(Cinema Museum, 10h, 3h, 2€), (City Museum, 16h, 2h, 1.80€)

The budget is OK. According to the calendar, the starting day for C_2 is Saturday, while for P_1 it is Sunday. Such a difference is ignored since the museums have the same timetable on Saturday and Sunday.

At this point, the following activities are found to overlap: (Art Museum, 10h, 3h, 3€) and (Cinema Museum, 10h, 3h, 2€). According to the procedure for adaptation (see section 3.1), the Art Museum is more constrained than the Cinema Museum (because of a more restrictive timetable). Hence, the Cinema Museum activity is selected for re-scheduling. First, an alternative past case is looked for, where the Cinema Museum was planned for a different time slot. But no case is found in the current case base. Second, the available empty slots in the current scheduling are 14-16h and 18-24h. The minimal staying period for the Cinema Museum is 3h, so it is not possible to place it between 14 and 16h. The museum closes at 18h, so it cannot be placed after 18h, either. So no solution is found for changing the Cinema Museum. As a consequence, the scheduling for the Cinema Museum is left unchanged and the Art Museum is re-scheduled. This time, there is a previous case, C_1, where the Art Museum was scheduled at 14h. Minimal staying period for the Art Museum is 3h, so the end of the activity is 17h. This new scheduling for the Art Museum does not violate any constraint. At this point the solution is: (Cinema Museum, 10h, 3h, 2€), (Lunch, 13h, 1h, 7€), (Art Museum, 14h, 3h, 3€), (City Museum, 16h, 2h, 1.80€).

In this solution, a new inconsistency arises between (Art Museum, 14h, 3h, 3€) and (City Museum, 16h, 2h, 1.80€). The visit to the Art Museum finishes at 17h, while the scheduled time for the City Museum is 16h. Since no other alternative has been found for the Art Museum, the next step is to re-schedule the City Museum.

There is no previous case in the case base regarding an alternative schedule for the City Museum. The available empty slots in the current schedule are from 17-24h. Hence, the City Museum can be scheduled from 17h to 19h, two hours duration, leaving at closing time. The final schedule is as follows: (Cinema Museum, 10h, 3h, 2€), (Lunch, 13h, 1h, 7€), (Art Museum, 14h, 3h, 3€), (City Museum, 17h, 2h, 1.80€).

5 Conclusions

In this paper, the problem of holiday scheduling for city visitors has been formulated as a CSP and then a methodology to solve it with the integration of Case-Based

Reasoning and Constraint Satisfaction techniques has been provided. The approach needs to be tested with experimental work. It would be especially interesting to evaluate the gain in critical situations, that is, to measure the trade-off between an exhaustive exploration of the CBR and the use of a CSP mechanism, in a quite unconstrained domain such as the one studied in this paper.

There are previous works in which the integration of CBR and CSP has been tackled. Purvis (Purvis,1997) has developed COMPOSER, a system that combines both techniques in order to solve configuration problems. Her practical experiments showed that the approach is efficient. Sqalli and Freuder also combine CBR and CSP in the problem of testing protocol interoperability (Squalli and Freuder, 1998). In their system, CBR is used when CSP fails. Scott and Simpson propose using CBR to solve a nurse-rostering problem (Scott and Simpson,1994). Cases are then used when hard constraints are satisfied. The work presented here is in line with the work of Scot and Simpson since the amount of constraints considered is not critical. Avesani et al. propose integrating CBR and CSP to build plans for fire fighting (Avesani et al. 1993). Bilgic and Fox use a case library to store design problems indexed by constraints (Bilgic and Fox, 1996). However, no adaptation process is performed on the solutions.

With regard to holiday planning, Huang and Miles also propose the use of CBR and CSP techniques, to configure packages for users according to a pre-established catalogue (Huang and Miles, 1995). His main goal is to help users making reservations. The approach presented in this paper focuses on scheduling at a different level. The two approaches may complement one another. Other approaches to holiday planning use CBR alone, such as (Blanzieri and Ebranati, 2000), or (Haig et al. 1997). The latter uses several cases to build a whole route, and then an A* algorithm to combine the results. Soo and Liang (1997) and Ricci and Werthner (2000) also work at the level of travel packages configured from catalogues. The innovation of the work presented in this paper, is that it is designed to provide a service at the basic level of holiday planning, that is, when visiting a city.

In future work, capacity constraints on resources should be considered. Particular museums accept a maximum number of visitors per day, theatres have a limited forum, hotels may be full in festival seasons, etc. A good scheduling system should be able to cope with these problems.

References

Avesani, P., Perini, A. & Ricci, F. (1993). Combining CBR and Constraint Reasoning in Planning Forest Fire Fighting. *Proc. EWCBR, Kaiserslautern,* 235-239.

Bilgic, T. & Fox, M.S. (1996). Constraint-Based Retrieval of Engineering Design Cases: Context as constraints. J. Gero and F. Sudweeks (eds.), *Artificial Intelligence in Design*, Kluwer Academic Publishers, 269-288.

Blanzieri, E. & Ebranati, A. (2000). COOL-TOUR: A Case Based Reasoning System for Tourism Culture Support. *Technical Report #0002-05*, Instituto per la Ricerca Scientifica e Tecnologica, Italy.

González, G., López, B. & de la Rosa, J.L. (2002). The Emotional Factor. An Innovative Approach to User Modelling for Recommender Systems. *RPeC02*.

Haigh, K.Z., Shewchuk, J.R., & Veloso, M.M. (1997). Exploiting Domain Geometry in Analogical Route Planning. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:509-541.

Huang, Y. & Miles, R. (1995). A Case Based Method for Solving Relatively Stable Dynamic Constraing Satisfaction Problems. *ICCBR-95.* LNAI, 1010, Springer.

López, B. (2002). Combining CBR and CSP: A case study on holiday scheduling. Technical Report, University of Girona, Spain.

Purvis, L. (1997). Dynamic Constraing Satisfaction using Case-Based Reasoning Techniques. *Constraint Programming'97, Workshop on Dynamic Constraint Satisfaction* .

Ricci, F. & Werthner, H. (2001). Case-based destination recommendations over and XML data repository. *Proc. Enter 2001.*

Scott, S. & Simpson, S. (1998). Case-Bases Incorporating Scheduling Constraint Dimensions. Experiences in Nurse Rostering. *EWCBR'98*, 392-401

Soo, V.W. & Liang, S.H. (2001). Recommending a trip plan by negotiation with a software travel agent. *Proc. Cooperative Information Agents.*

Sqalli, M.H. & Freuder, E.C. (1998). Diagnosing InterOperability Problems by Enhancing Constraint Satisfaction with Case-Based Reasoning. *Ninth International Workshop on Principles of Diagnosis,* Sea Creast Resort, Massachussetts, 266-273.

Yager, R.R. (1988). On Ordered Weighted Averaging Aggregation Operators in Multi-Creiteria Decision Making. *IEEE Transactions on SMC*, vol. 18, 183-190.