Combining CBR and CSP: A case study on holiday scheduling

Beatriz López[1]
Universitat de Girona
Campus Montilivi, Edifici P IV
17071 Girona, Spain
blopez@eia.udg.es

Abstract: Constraint Satisfaction Problems (CSP) are hard problems that require a lot of effort on both the development and the execution time. To improve CSP solving efficiency, some systems have started to incorporate Machine Learning techniques. Among them, Case-Based Reasoning (CBR) is a sub-field of Machine Learning, in which problems are solved based on past experiences. One of the weakest point of CBR methodology is the adaptation process in which past solutions are used to solve new problems. In this paper we introduce an approach to Case-based Reasoning (CBR) and Constraint Satisfaction (CSP) techniques integration to improve both, CBR adaptation trough CSP and CSP efficiency trough the use of past experiences.

1. Introduction

Constraint Satisfaction Problems (CSP) are hard problems that are being studying in Artificial Intelligence from several years. The problem of assigning a set of values to a set of variables in such a way that no constraints are violated, has been tackled from different approaches, from general solutions to domain specific ones [1]. Although there are several successful results on CSP, the effort required to solve a CSP problem is usually huge in both the implementation and the execution time. To improve CSP solving efficiency, some systems have started to incorporate Machine Learning techniques [2,3,4].

Case-based Reasoning is a particular case of Machine Learning techniques in which past experiences (problem solutions) are used to solve new problems [5]. CBR follows a four steps process:

- Retrieval of past experiences similar to the new situation
- Reuse of past solutions to elaborate a solution for the current problem
- Revision of the current solution, and
- Retain the new experience (problem-solution) in memory.

Maybe, the weakest point of this methodology is the reuse phase, also known as adaptation. The goal of this phase is to adapt previous experiences to new ones, in order to solve new problems. Current approaches follow a quite simplistic or null technique, transferring the solution as it is in the past experience.

The integration of CBR and CSP can provide then, moreover than an improvement of CSP problem solving, an improvement on the adaptation phase of CBR. That is, on one hand, CBR can contribute to make CSP solving efficient by providing solutions at one shot based on past experiences. And on the other hand, CSP can be used in the adaptation phase of CBR, providing a solid methodology for case reuse. Particularly, several authors have argued about the synergy of combining Case-based Reasoning (CBR) and CSP techniques in order to improve both [6].

Several developments on CBR and CS integration have been performed in the middle's of 90 [7,8,9,10,11,12,13,14,15,16]. However, few researchers continue this integration work. We think that the main reason is due to the fact that CSP research concentrate on theoretical proves of CSP models, meanwhile the use of CBR in solving CSP problems give practical results hardly to prove by mathematical methods [9].

In this paper we recover the idea of combining CBR and CSP because we thing that benefits of integration are particularly useful in scheduling, real life problems. We provide a model for integration in a particular scenario: holiday scheduling. Our aim is to develop a citizen service available 24 hours through Internet (information desks at airports, train stations, etc.) in such a way that a tourist arriving to a city at anytime can schedule his/her stay. The user selects from a list the set of activities he/she wants to enjoy. Then, our model provides the user with a scheduling of the activities based on the experiences of past users. In the scheduling, activities regarding temporal events in the city as well as other ones no selected by the user are proposed in order to fill time gaps.

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

The paper is organised as follows: In section 2 we provide the formulation of the holiday scheduling problem in terms of CSP. Then, we continue in section 3 providing the CBR methodology to solve problems, and in section 4 the particular CSP techniques used in the adaptation process. We give some examples in section 5, and we end by describing some related work on section 6 and some conclusions in section 7.

2. Problem formulation

Holiday scheduling consists on organising in a period of time a set of activities a tourist wishes to enjoy in a given destination. Holiday scheduling can be seen at different levels of detail. There are some works that concentrate on the elaboration of packages in which the main issue is to allocate different tourist attractions and spots in a set of days [17,18,10]. Other researchers have concentrate on the selection of a given hotel for a destination (AICBR). And some others on particular tourist services in a given region [19]. Our work is in line with the later approach.

Our aim is to provide to the user with a tool to schedule the different activities he can enjoy at the city of Girona. Activities can be of several kinds: visiting museums, monuments, art exhibitions, temporal cultural events (as festivals, markets, performances), sightseeing, parks, and sportive installations among others. Some of the activities are constrained to timetables, as museums do. Moreover, some ones also require entrance, so the user should have some budged for them. Every visiting spot requires a minimal period of stay estimated for the different holders of the activities. For example, the visit to the City Museum is estimated on 2 hours.

Then, we clearly identify some activities (what the user wants to do), duration of the activities, constraints (timetables), and resources (budged), in such a way that we can formulate a single stay to the city of Girona as constraint satisfaction problem.

Activities

Every activity has a duration d_i that express the minimal staying period. For the sake of simplicity transport time required to go to the activity place is included in the minimal staying period.

Every activity has an associated cost c_i in Euros (resource). Cost could be 0 for free entrance activities (for example in sightseeing activity).

No precedence relations between activities exist. However, some of the activities are only allowed in a given set of times (timetables) that depend on the day of the week and on the day of the year, and also on the season. For example, regarding the day of the weak, the Cathedral Museum is closed on Monday, and is open from Tuesday to Saturday from 10:00 to 14:00, ad from 16:00 to 19:00; and on Sunday: 10:00-14:00. But on Monday June 24th, 2002, being holiday, the Cathedral Museum opens from 10:00 to 14:00. Inversely, on Tuesday January 1st, being holiday, the Cathedral Museum does not open in the afternoon (from 10:00 to 14:00). In addition, the Museum has three different timetables: one from October to February, second from March to June, and third from July to September.

Each activity follows different weak and season parameters. Then, to handle all these temporal variables, we associated to each activity with a timetable matrix, T_i, of month x day dimension:

$$T_i = \begin{bmatrix} L_{Jan,1} & L_{Jan,2} & \cdots & L_{Jan,31} \\ L_{Feb,1} & L_{Feb,2} & \cdots & L_{Feb,31} \\ \cdots & & & \\ L_{Dec,1} & L_{Dec,2} & \cdots & L_{Dec,31} \end{bmatrix}$$

Each $L_{j,i}$ 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 $L_{j,i}$ can be empty, as for example the ones corresponding to closing days as wells as non existing days (i.e. February 30th). Then, for the Cathedral Museum, we have $L_{Jan,1} = ((10,14))$ and $L_{Apr,9} = ((10,14),(16,19))$, being both, January 1st and April 9th, Tuesday.

Constraints

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

- The sum of the minimal staying period of all activities must not exceed the holiday period of the user

$$\sum_i d_i \leq duration \tag{C1}$$

- The sum of the cost of all activities must not exceed the user's budged.

$$\sum_i c_i \leq budged \tag{C2}$$

Depending on duration, restaurant and hotel activities are automatically added. If the duration exceeds half day, then a lunch activity is allocated. It is not mandatory to have budged for lunch, although it is to have some period to have lunch. In this sense the lunch price is a soft constraint: it can be removed in order to find a solution for the scheduling problem. If the duration exceeds one day, a dinner and a hotel activity are allocated. The automatic selection of these activities has effects on the budged constraint; then, at the beginning of the problem formulation restaurants (lunch and dinner) and hotel activities are added to the problem and then treated as any other activity. Cost of the activities is initialised according to user preferences expressed in the user profile or asked explicitly to the user, and taking into account user budged. The selection is done with the help of recommender systems as [20,21,22].

Lunch and dinner have some time constraints. For the sake of simplicity, we assume that the tourist is aware of the country habitudes, and then, in the case of Girona:

- Lunch is scheduled at 13h, with a minimal staying period of 1h
- Dinner is scheduled at 20h, with a minimal staying period of 20h.

However, it is well known that some European tourists, particularly German, enforce their timetable to tourist places. So, the time constraint related to lunch and dinner should be made more flexible, adapted to user profiles. The relaxation of that constraint will be tackled in future work.

Lunch, dinner and hotel are defined as minimal set of components following [17]. Every city has several "mandatory" spots that also belong to that set: places that every tourist should visit. However, it could be the case that the user comes to the city as a second, third, or tenth time. So in our approach no other minimal set of spots are required.

Finally, the final scheduling is completed with filling activities in order to deal with time gaps. Such activities can also contribute to the budged, but they are no matter of constraint. If no budged is left, no cost extra activities will be planned.

To solve the holiday scheduling problem we propose the use of a combination of CBR techniques and CSP heuristics. We keep in a case base a collection of solved CSP. When a user asks for a new scheduling, previous cases are retrieved upon similarity on activities, constraints, and budgets on past experiences following a CBR methodology. CSP techniques are used to adapt past solutions to new situations.

3. The CBR component

When a new problem is formulated, it is solved by using past experiences stored in a case base following CBR methodology in which it is distinguished four main phases: retrieval, reuse, revise, retain. In the retrieval phase, old solutions from the case base are recovered to solve the new problem. In the reuse phase, the solution of the past problem is used to solve the new problem. It is precisely in this phase when we use CSP techniques in order to adapt the old solution to the new problem. Then in the revise phase, the solution of the case is checked, and then learned in the retain phase.

3.1 Case Base

We keep in a case base a collection of solved CSP. A case is then a CSP in which we distinguish the following parts:

- Identification: a numerical value
- Constraints:
  - Duration of the stay
  - Budged
  - Temporal constraints:
    - Season: seasons mark eventual events.
    - Starting day of the year
    - Starting weekly day
- Selected activities: activities the user wants to do
- User profile: age, sex, civil state, profession, and hobbies.

- Solution of the case: the final price, the starting time, the final time, and the scheduling. The later consists of an ordered list of tuples representing the sequence of activities. Each tuple (A, S, D, C) contains information of the activity name (A), the starting time (S), the duration (D), and the cost (C).

A case example is shown in figure 1.

Identification	1
Duration:	1
Budged:	20
Temporal:	((Season Februrary), (Year_day 3-Feb-2002), (Weekly_day Sunday)
Selected Activities	(Archaeological Museum, Art Museum, Cathedral Museum)
User profile	((Age 23), (Sex Female), (Civil state Single), (Profession Computer Science), (Hobbies Travel, Sports, Photography))
Solution	((Price 18.05), (Initial time 10h), (Final time 19h)
	(Archaeological Museum, 10h, 3h,1.80)
	(Lunch,13h,1h,9.00)
	(Art Museum, 14h, 3h,3)
	(Cathedral Museum, 17h, 2h,4.25))

Fig. 1. Case example

3.2 Case Retrieval

When solving a new case, two kind of information is required from the user:

- Constrains. This information is used by to filter temporal activities and presenting to the user the available activities for the corresponding days.
- Second, the user selects the activities he plans to enjoy: museums, cultural events, etc.

Then, a first consistency checking is performed to guarantee constraints C1 and C2, that is to say, the duration and budged of the trip are enough. Until these constraints are not satisfied, no further steps are performed. The last information gathered constitutes a new case or probe.

Case retrieval is based mainly on a three-step process: Activity matching, constraint matching and selection (see figure 2). 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 constrains. No case is rejected. Constraints must be accomplished in the final scheduling, but previous approximations to these constraints can be useful for 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) [23]:

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

where:

- n is the number of constraints (there are 5 in total)
- $cons_j(X)$ is the j-constraint of case X.
- $sim_{\sigma(j)}(x,y)$ is the similarity between the two j-constraints (see below)
- σ(j) is a permutation of the values 1…n so that $sim_{\sigma(j-1)}(cons_{j-1}(P), cons_{j-1}(C_i) \bullet sim_{\sigma(j)}(cons_j(P), cons_j(C_i)$ $\forall i = 2,\ldots,n$.
- w_i is the relevance of each constraint in the similarity.

The OWA is suitable to compute similarity since we can order the different constraint similarities based on their degree. Thus the key of the similarity function is the weight definition. In our model we use the following function

$$w_i = \begin{cases} \dfrac{1}{2i} & i > 1 \\ 1 - \sum_{i=2}^{n} w_i & i = 1 \end{cases} \quad (3)$$

For the number of constrains we have, n=5, and it does hold that $w_{i-1} > w_i$ and $\sum_{i=1}^{n} w_i = 1$. More precisely, w_1=0.35, w_2=0.25, w_3=0.17, w_4=0.13 and w_5=0.1.

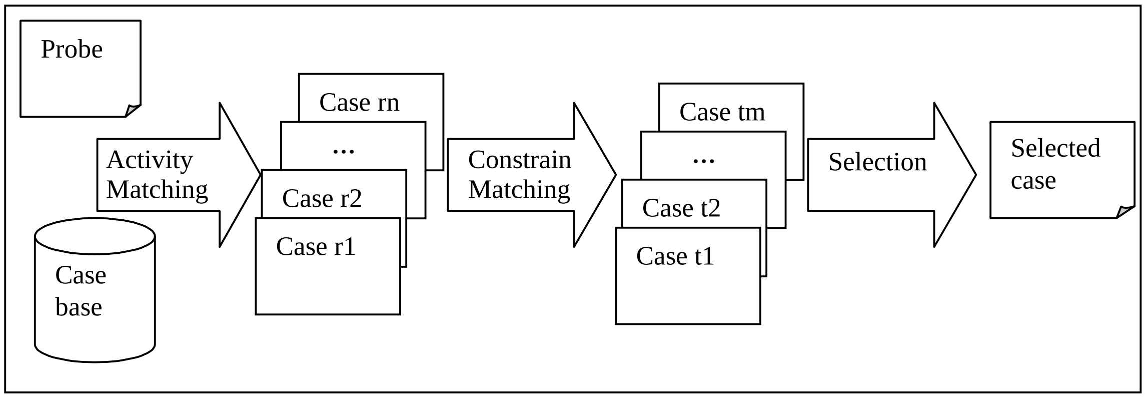

Fig. 2. Retrieval phase

The remaining definitions are related to the constraint similarity functions, one for each constraint: budged, duration, season and starting days (year and week).

$$sim(budged(P), budged(C_i)) = \begin{cases} \dfrac{1}{|budged(P) - budged(C_i)|} & budged(P) \neq budged(C_i) \\ 1 & otherwise \end{cases} \tag{4}$$

$$sim(duration(P), duration(C_i)) = \begin{cases} 1 - \dfrac{duration(C_i) - duration(P)}{7} & duration(P) \leq duration(C_{ii}) \\ 0 & otherwise \end{cases} \tag{5}$$

$$sim(season(P), season(C_i)) = 1 - \frac{|season(P) - season(C_i)|}{12} \tag{6}$$

$$sim(day\_year(P), day\_year(C_i)) = \begin{cases} 1 & day\_year(P) = day\_year(C_i) \\ 0 & otherwise \end{cases} \tag{7}$$

$$sim(day\_week(P), day\_week(C_i)) = \begin{cases} 1 & day\_week(P) = day\_week(C_i) \\ 0 & otherwise \end{cases} \tag{8}$$

Finally, the selection method is simple: we choose the case with the higher similarity degree.

3.3 Case Reuse, Revise and Retain

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

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 could not be the optimal. Moreover, the user can not agree with the filling activities proposed, or the solution in general. Then, he/she can accept it or modify it. That is to say, the user is who evaluates the outcomes.

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. most than one case has been used to solve the case), then the new case is stored. It will probably be useful in other situations, avoiding wasting time in accessing to the case base. Otherwise, the case will not be stored, since the adaptation process is simply and not time consuming. Avoiding the storage of all cases, we are trying to tackle on of the main drawbacks of CBR, case base maintenance [24], keeping the case base in a reasonable size.

4. The CSP component

To adapt the retrieved solution to the current problem, we propose the following procedure:

1. Copy the past solution to the new case
2. Remove activities not selected by the user and that does not hold in the corresponding season
3. Check inconsistencies
   3.a. Duration exceeded
   3.b. Budged 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 filling activities according to user profile

The remaining of this section is devoted to explain each of these steps.

Solving duration inconsistencies. This situation can occur when, for example, the user has specified 1 day duration and the retrieved case is 2 day duration. In such a situation, the only activities copied to the solution are the ones required by the user, together with the minimal set of components (lunch, dinner, hotel).

Solving budged inconsistencies. Budged violation can happens as a consequence of the constraint matching process (see equations 2 and 4). However, it does not mean that the case has no solution. Constraint C2 has been checked at the beginning, so the case should be solved within the current budged. Imagine, for example, than the retrieved case corresponds to a visit of two days duration long, and the current user is interested in a scheduling for a scheduling of single day duration. Obviously, since in the retrieved case the user has to spend money on a hotel, the budged is higher. In such a situation, a duration inconsistency has been previously detected, then, the solution of the case has been not completely transferred. As a consequence, what it is necessary to check is if the final cost of the activities remains under the budged (C2).

Solving timetables inconsistencies. Timetable violation happens when we recover a scheduling that does not exactly match either the day of the year or the weekly day. Then, some activities planned for one day cannot be performed. To solve this inconsistency we propose to recover past cases that provide an alternative scheduling for the inconsistent activities.

Solving overlapping activities. This situation can happens when either the duration of the retrieved case is not the same, and the activities scheduled for different days have been transferred to the new case as a single day; or when the current scheduling contains parts of solutions coming from different cases (as a consequence, for example, of solving timetable inconsistencies). Then, the solution we propose is to use the Dynamic Minimum Conflicts Algorithm developed by Lisa Purvis [9] to tackle inconsistencies due to overlapping activities (see figure 3).

```
When a dead-end is reached (i.e. no valid value can be found)
        Until encountering a choice point with an unexplored alternative
              1. Withdraw the most recently made choice
              2. Undo all consequences of the withdrawn choice by adding or
                 deleting all appropriate variables and updating problem
                 constraints accordingly
                 Move forward again making choices that conflict the
                 least with choices made so far
```

Fig. 3. Dynamic Minimum Conflicts Algorithm

In our approach:

1) We choose among the conflicting activities the one that is less constrained.
2) We search in the case-base for alternative schedulings for that activity
3) If some one is found, we suggest the new scheduling.
4) Otherwise, we search for available time gaps in current scheduled compatible with the activity. If some one is found, then we re-schedule it.

5) Otherwise, we leave unchanged the conflicting activity chosen in step 1 and we proceed by selecting the next less conflicting activity for re-scheduling. Then, we continue on step 2 until all overlapping activities are solved.

Complete the set of selected activities. Once have been solved the main inconsistencies, the problem is completed when all activities proposed by the user are present in the solution. Adding activities can be done by swapping etiher removing filling activities or scheduled activities not proposed by the user or by placing in the slot the pending activities.

Adding filling activities. The addition of filling activities is performed according to the user profile, following recommender system approaches as the ones in [20,25].

5. Examples

This section is devoted to illustrate our methodology by means of several examples. For this purpose, consider than in the case base are the cases shown in figure 4.

5.1 Example 1

Assume that we have a new case (let's call it P_1) with the following selected activities: Art Museum, Cinema Museum, City Museum. At the first step of the retrieval phase, activity matching for every case in memory (equation 1) is the following:

$$Sim_a(P_1, C_1) = \frac{1}{3} \qquad Sim_a(P_1, C_2) = 1$$

From the activity matching, case C_2 is retrieved as the best case. No further similarity computations are performed and then the adaptation process starts.

1. The past solution is transferred to the new case.
2. 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.
3. Check inconsistencies

   3.a. Current case is 1 day duration and the retrieved case is 2 days duration.
   Then, we just leave the activities selected by the user:
   (Lunch,13h,1h,7.00),(Art Museum, 10h, 3h,3),
   (Cinema Museum, 10h, 3h, 2),(City Museum, 16h, 2h, 1.80)

   3.b. Budged is OK.

   3.c. According to the calendar starting day of C_2 is Sunday while on P_1 is Sunday. Such difference is ignored since Museums have the same timetable on Saturday than on Sunday.

   3.d. The following activities are found to be overlapped:
   (Art Museum, 10h, 3h,3) overlaps (Cinema Museum, 10h, 3h, 2)

Then we proceed solving the inconsistency:

1) Art Museum is more constrained than Cinema Museum (more restrictive timetable). Then we leave the slot time for Art Museum unchanged and we try to find a new one for the Cinema Museum. According to the methodology, an alternative case is searched where Cinema Museum was planned in a different time slot. But no case is found in the current case base.
2) Available values in the current scheduling are 14-16h, 18-24. Minimal staying period for Cinema Museum is 3h, so it is not possible to collocate it between 14 and 16h. The museum closes at 18, so it can be neither placed after 18h. So no solution is found for changing Cinema Museum.
3) Then we try to find a new schedule for Art Museum. In this case, there is a previous case, C_1, where Art Museum was scheduled at 14h. Minimal staying period for the Art Museum is 3h, so the end of the activity is 17. This new scheduling for the Art Museum does not violate any constrain.

Current solution is then: (Cinema Museum, 10h, 3h, 2), (Lunch,13h,1h,7.00), (Art Museum, 14h, 3h,3), (City Museum, 16h, 2h, 1.80). In this solution, a new inconsistency rises 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 Art Museum, then the next activity to re-schedule is City Museum.

There is no previous case in the case base regarding an alternative schedule for City Museum. Available values in the current schedule are from 17-24h. Then, the City Museum can be scheduled from

17 to 19, two hours duration, and leaving just at closing time. The final schedule is the following: (Cinema Museum, 10h, 3h, 2), (Lunch,13h,1h,7.00), (Art Museum, 14h, 3h,3), (City Museum, 17h, 2h, 1.80)

Case 1
Duration: 3
Budged: 200
Temporal: (Season June), (Year day 22-June-2002), (Week day Saturday)
Selected activities: Jewish Museum, Cathedral Museum, City Museum
Profile: (Age 23), (Sex Female), (Civil state Single), (Profession Computer Science), (Hobbies Travel, Sports, Photography)
Solution: (Price 190.85),(Initial time10h),(Final time 20h)((Jewish Museum, 10h, 3h,1.8),(Lunch,13h,1h,9.00),(Beach,14h,6h,0),(Dinner, 20h,2h,15), (Pub,22h,2h,20),(Disco,24h,3h,20),(Hotel, 03h,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
Budged: 100
Temporal: (Season February),(Year day 2-Feb-2002),(Weekly day Saturday)
Selected activities: Art Museum, Cathedral Museum, Cinema Museum, City Museum
Profile: (Age 23), (Sex Female), (Civil state Single), (Profession Computer Science), (Hobbies Travel, Sports)
Solution:(Price 90.05),Initial time 10h),Final time 19h),((Art Museum, 10h, 3h,3) , (Lunch,13h,1h,7.00),(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.4. Case base used in the examples

5.2 Example 2

This second example is devoted to illustrate how calendar inconsistencies are handled. The new case description (probe P_2) consist on the following selected activities: Jewish Museum, Cathedral Museum, City Museum, Cinema Museum and Art Museum. Constrains made explicit by the user are: (Duration 3), (Budged 200), (Season December), (Year day 30-Dec-2002), (Week_day Sunday). According to the current case base, we recover from matching activities cases C1 and C2 with, $Sim_a(P_2,C_1)=Sim_a(P_2,C_2)=1$. Then, the constraint matching step provides C1 as the best case (see details on table 1).

When copying the solution to the new case, activities not corresponding to the current season are removed (i.e. Beach). Then, checking inconsistencies reveals that timetables are incompatible due to calendar reasons. Differences are shown on table 2. Activities planned on Monday in the past, being Sunday on June 23th, cannot be copied to a Monday on December, since on Monday most of Museums are closed. Then (Cathedral Museum) must be rescheduled.

Searching in the case-based, we found that case 2 have scheduled the Cathedral Museum in the past. So we adapt the scheduling for Cathedral Museum to the current problem. The current scheduling is now: (Jewish Museum, 10h, 3h,3) ,(Lunch,13h,1h,9.00),(Devessa, 14h, 2h, 0),(Cathedral Museum, 17h, 2h, 1.8),(Sightseeing, 19h, 1h, 0),(Dinner, 20h, 2h, 15),(Pub,22h,2h,20),(Disco,24h,3h,20),(Hotel, 03h,7h,45),(Sightseeing-outcity, 10h, 3h, 0),(Lunch, 13h, 1h, 7),(Firanadal, 14h, 4h, 0),(Fontajau sportive installations, 18h, 2h, 5),(Dinner, 20h, 2h, 15),(Pub,22h,2h,20),(Disco,24h,3h,20),(Hotel, 03h,7h,45),(City Museum, 10h, 2h, 1.80),(Lleona Market, 12h, 1h, 0),(Lunch, 13h, 1h, 7),(Lleuresport, 14h, 2h, 0).

Finally, activity chosen by the user and not yet included should be added. Particularly,),(Sightseeing-outcity, 10h, 3h, 0), is swapped by (Art Museum, 10h, 3h, 3); and (Firanadal, 14h, 4h, 0) by (Cinema Museum, 14h, 4h, 2).

Table 1. Calendar differences between P_2 and C_1.

	C_1	C_2
Budged	1	0.01
Duration	1	0.87
Season	0.41	0.75
Year day	0	0

Week day	0	0
$Sim_c(P_1, C_i)$	0.35*1+0.25*1+0.17*0.41=0.67	0.35*0.87+0.25*0.75+0.17*0.01=0.49

6. Related work

There are previous works in which the integration of CBR and CSP has been tackled. Purvis [9,28] has developed COMPOSER, a system that combines both techniques to solve configuration problems. The approach implemented in COMPOSER is based on an unification mechanism in order to map current problems to past solutions and follows the Dynamic Minimum Conflicts Algorithm. The integration of CBR and CSP does not allow backtracking, in the methodology proposed by Purvis. Her practical experiments shown that the approach is efficient.

Sqalli and Freuder also combine CBR and CSP in the problem of testing protocol interoperability [16]. In their system, CBR is used when CSP fails. Scott and Simpson [26] also proposes the use of CBR to solve the nurse rostering problem. They propose to store in a case base patterns of nurse preferences, that is, soft constraints of problems. Cases are then used when hard constraints are satisfied. Avesani et al.[17] and Ricci et al. [8] propose the integration of CBR and CSP to build plans for fire fighting. They propose a two part methodology in which first, case-based planning is used to recover similar plans for different zones, and then a CSP approach to build the final plan, allocating the corresponding resources. Bilgic and Fox [13] use a case library to store design problems indexed by constraints. Cases are recovered upon new descriptions. However, no adaptation process is performed on the solutions.

Table 2. Calendar differences between P_2 and C_1.

C_1	Activities	P_2
Saturday	(Jewish Museum, 10h, 3h,1.8), (Lunch,13h,1h,9.00), (Dinner, 20h, 2h, 15),(Pub,22h,2h,20) ,(Disco,24h,3h,20),(Hotel, 03h,7h,45)	Sunday
Sunday	(Cathedral Museum, 10h, 2h, 4.25), (Sightseeing, 12h, 1h, 0), (Lunch, 13h, 1h, 7), (Dinner, 20h, 2h, 15), (Pub,22h,2h,20), (Disco,24h,3h,20), (Hotel, 03h,7h,45)	Monday
Monday (Holiday)	(City Museum, 10h, 2h, 1.80), (Sightseeing, 12h, 1h, 0), (Lunch, 13h, 1h, 7)	Tuesday (Holiday)

Regarding Holiday planning, Huang [27] proposes also the use of CBR and CSP techniques to configure packages to users according to pre-established catalogue. His main goal is making reservations. We think that our approach focus scheduling at another level of detail; so both approaches can be complementary. Other approaches to holiday planning use CBR alone, as in Blanzieri et al. [19], or in Haig et al 97 [14]. The later uses several cases to build a whole route, and then a A* algorithm to combine the results. Soo and Liang[17] and Ricci et al [18] works also at the level of travel packages configured from catalogues. We think that our approach is closed to the citizens and provides a service at the low level of holiday planning, that is, when visiting a city. This is an innovative approach.

7. Conclusions

In this report we have formulated the problem of holiday scheduling as a CSP and then provided a methodology to solve it with the integration of Case-Based Reasoning and Constraint Satisfaction techniques. The synergy of the integration improves CSP as well as the adaptation phase of CBR. The approach must be refuted with experimental work. It is specially interesting the evaluation of 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 non quite constrained domain.

It is important to note, also, that the problem formulated is not critical: constraints must not be violated, but activities proposed by the user are not too much. In other domains, as for example in emergency situations, activities and duration have a more critical relation.

Another issue of our approach is that no capacity constraints on resources have been considered. This is the case of the major activities contemplated in the model: Museums. However in further developments capacity should be taken into account since, for example, special museums accept a maximum number of visitors per day, theatres have a limited forum, hotels can be full in festival seasons, etc. We will continue extending the model to cope with them.

References

1. Pedro Meseguer: Constraint Satisfaction Problems: An Overview. AICOM, Vol 2., # 1, March 1989, pp. 3-17.
2. S. L. Epstein, E. Freuder: Collaborative Learning for Constraint Solving. CP'01
3. W. Zhang, T.G. Dietterich: Solving Combinatorial Optimization Tasks by Reinforcement Learning: A General Methodology Applied to Resource-Constraint Scheduling. Technical Report, Department of Computer Science, Oregon State University, 1997.
4. W. Zhang, T.G. Dietterich: A Reinforcement Learning Approach to Job-shop Scheduling.IJCAI'95.
6. M.H. Sqalli, L. Purvis, E.C. Freuder: Survey of Applications Integrating Constraint Satisfaction and Case-Based Reasoning. PACLP99.
7. P. Avesani, A. Perini, F. Ricci: Combining CBR and Constraint Reasoning in Planning Forest Fire Fighting. Proc. EWCBR, pp. 235-239, Kaiserslautern, 1993.
8. F. Ricci, S. Mam, P. Marti, V. Normand, P. Olmo: CHARADE: a Platform for Emergencies Management Systems. IRST Technical Report #9404-07, 1994
9. Lisa Purvis: Intelligent Design Problem Solving Using Case Based and Constraint Based Techniques. PhD Thesis, Department of Computer Science and Engineering, The University of Connecticut.
10. Y. Huang and R. Miles. “A Case Based Method for Solving Relatively Stable Dynamic Constraing Satisfaction Problems. ICCBR-95, Case-Based Reasoning Research & Development. LNAI, 1010, Springer, 1995.
11. Y. Huang, R. Miles: Combining case based and constraint based techniques in travel reservation systems. Proc. 11th Conference on Artficial Intelligence for Applications (CAIA'95).
12. Kazuo Miyashita, Katia P. Sycara. “CABINS: A Framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair”. Artificial Intelligence Journal, vol 76, num. 1-2, pp. 377-426, 1995.
13. Tarner Bilgic, Mark. S. Fox. "Constraint-Based Retrieval of Engineering Design Cases: Context as constraints. Artificial Intelligence in Design'96, J. Gero and F. Sudweeks (eds.), Kluwer Academic Publishers, pp. 269-288.
14. Karen Zita Haigh, Jonathan Richard Shewchuk and Manuela M. Veloso: Exploiting Domain Geometry in Analogical Route Planning. Jorunal of Experimental and Theoretical Artificial Intelligence, 9:509-541, 1997.
15. S. Scott: Learning Behaviour in a Case-Based Workforce Scheduler. School of Computing and Mathematics, University of Huddersfield, UK, 1998.
16. M.H. Sqalli, E. C. Freuder: Diagnosing InterOperability Problems by Enhancing Constraint Satisfaction with Case-Based Reasoning. Ninth International Workshop on Principles of Diagnosis, Sea Creast Resort, Massachussetts, pp. 266-273, 1998.
17. V.-W. Soo, S.-H. Liang: Recommending a trip plan by negotiation with a software travel agent. CIA’01.
18. F. Ricci, H. Werthner: Case-based destination recommendations over and XML data repository. Proc. Enter 2001.
19. E. Blanzieri, A. Ebranati: COOL-TOUR: A Case Based Reasoning System for Tourism Culture Support. Technical Report #0002-05, Instituto per la Ricerca Scientifica e Tecnologica, Italy, 2000.
20. M. Montaner, B. López, J.L. de la Rosa: Improving Case Representation and Case Base Maintenance in Recommender Agents. Accepted to ECCBR’02.
21. A. Gavarró: SMA que proporciona serveis de restauració. B.Sc.Thesis, Rovira i Virgili University, Tarragona, Spain, September 2001.
22. R. Burke: “A Case-Based Reasoning Approach to Collaborative Filtering”. In: E. Blanzieri and L. Portinali (eds), Advances in Case-Based Reasoning, (EWCBR2000), pp. 370-379, Springer-Verlag, 2000.
23. R.R. Yager. : On Ordered Weighted Averaging Aggregation Operators in Multi-Creiteria Decision Making. In IEEE Transactions on SMC, volume 18, pages 183-190, 1988.
24. D. B. Leake, D.C. Wilson: “Guiding Case-Base Maintenance: Competence and Performance”. Online Proceeding of the ECAI’2000 Workshop on Flexible Strategies for Maintaining Knowledge Containers.
25. G. González, B. López, J.L. de la Rosa: “The Emotional Factor. An Innovative Approach to User Modelling for REcommender Systems”. RPeC02.
26. S. Scott, R. Simpson: Case-Bases Incorporating Scheduling Constraint Dimensions. Experiences in Nurse Rostering. EWCBR'98, 392-401
27. Y. Huang: Using Case-Based Techniques to Enhance Constraint Satisfaction Problem Solving. Applied Artificial Intelligence, Vol. 10, Num. 4, August, 1996, 307-328.
28. L. Purvis: Dynamic Constraing Satisfaction using Case-Based Reasoning Techniques. Constraint Programming’97, Workshop on Dynamic Constraint Satisfaction.