Allotment problem in travel industry: A solution based on ASP*

Carmine Dodaro, Nicola Leone, Barbara Nardi, and Francesco Ricca

Department of Mathematics and Computer Science, University of Calabria, Italy
{dodaro,leone,b.nardi,ricca}@mat.unical.it

Abstract. In the travel industry it is common for tour operators to pre-book from service suppliers blocks of package tours, which are called allotments in jargon. The selection of package tours is done according to several preference criteria aimed at maximizing the expected earnings given a budget. In this paper we formalize an allotment problem that abstracts the requirements of a real travel agent, and we solve it using Answer Set Programming. The obtained specification is executable, and it implements an advanced feature of the iTravel+ system.

1 Introduction

In the travel industry it is common for tour operators to pre-book for the next season blocks of package tours, which are called allotments in jargon [11, 12]. This practice is of help for both tour operators and service suppliers. Indeed, the first have to handle possible market demand changes, whereas the seconds are subject to the possibility that some package tours remain unsold, e.g., the rooms of a hotel can remain empty in a given season. Therefore, service suppliers and tour operators agree on sharing the economic risk of a potential low market demand by signing allotment contracts [12]. The effectiveness of this form of supplying has been studied in the economics literature under a number of assumptions on the behavior of the contractors [11, 21]. These studies, however, do not approach the problem of providing a tool that helps travel agents in the act of selecting package tours to be traded with service suppliers in the future market. Basically, given a set of requirements on the properties of packages to be brought, budget limits, and an offer of packages from several suppliers, the problem from the perspective of the travel agent is to select a set of offers to be brought (or pre-booked) for the next season so that the expected earnings are maximized [11]. Despite allotment is –de facto– one of the most commonly-used supplying practices in the tourism industry, the final selection of packages offered by travel suppliers is often done in travel agencies more or less manually.

In this paper we approach the problem of automatic allotment of package tours, and we formalize and solve it by using Answer Set Programming (ASP) [4, 15, 19]. ASP is a declarative rule-based programming paradigm for knowledge representation and declarative problem-solving. The idea of ASP is to represent a given computational problem by using a logic program, i.e., a set of logic ruler, such that its answer sets

* This work has been partially supported by the Calabrian Region under PIA project iTravelPlus POR FESR Calabria 2007-2013

correspond to solutions, and then, use an answer set solver to find such solutions. The high knowledge-modeling power [4, 15], and the availability of efficient systems [1, 2, 18], make ASP suitable for implementing complex knowledge-based applications. Indeed, ASP has been applied in several fields ranging from Artificial Intelligence [3, 5] to Knowledge Management [4], and Information Integration [24]. In particular ASP has already been exploited in the tourism domain for identification of package tours that best suit the customers of an e-tourism system [27]. In this paper we follow this trend and we describe a new application of ASP in the touristic domain. The contributions of this paper can be summarized as follows:

- We abstract the requirements of a real travel agent that needs to solve an allotment problem, and we solve it by using an ASP program.
- We model in ASP a number of additional preference criteria on packages to be selected that, according to a travel agent advise, allow one to further optimize the selection process by taking into account additional knowledge of the domain.
- We report on the results of a preliminary experimental analysis using real-word data that validates our approach.

A tool based on the ASP encoding described in this paper will be included as an advanced service of the e-tourism platform developed under the iTravelPlus project by the Tour Operator Top Class s.r.l. and the University of Calabria.

The paper is structured as follows: Section 2 overviews ASP syntax and semantics; Section 3 exemplifies the requirements of an automatic allotment tool; Section 4 presents a formalization of the allotment problem in ASP; Section 5 presents the results of an empirical evaluation of our approach; Section 6 discusses related work, and Section 7 concludes the paper summarizing the obtained results.

2 Answer Set Programming

Answer Set Programming (ASP) [4, 15, 19] is a programming paradigm developed in the field of nonmonotonic reasoning and logic programming. In this section we overview the language of ASP, and we recall a methodology for solving complex problems with ASP. More detailed descriptions and a more formal account of ASP, including the features of the language employed in this paper, can be found in [8, 16, 19], whereas a nice introduction to ASP can be found in [4]. Hereafter, we assume the reader is familiar with logic programming conventions.

Syntax. The syntax of ASP is similar to the one of Prolog. Variables are strings starting with uppercase letter and constants are non-negative integers or strings starting with lowercase letters. A *term* is either a variable or a constant. A *standard atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity n and $t_1, \ldots, t_n$ are terms. An atom $p(t_1, \ldots, t_n)$ is ground if $t_1, \ldots, t_n$ are constants. A *ground set* is a set of pairs of the form $\langle consts : conj \rangle$, where $consts$ is a list of constants and $conj$ is a conjunction of ground standard atoms. A *symbolic set* is a set specified syntactically as $\{Terms_1 : Conj_1; \cdots ; Terms_t : Conj_t\}$, where $t > 0$, and for all $i \in [1, t]$, each

$Terms_i$ is a list of terms such that $|Terms_i| = k > 0$, and each $Conj_i$ is a conjunction of standard atoms. A *set term* is either a symbolic set or a ground set. Intuitively, a set term $\{X : a(X, c), p(X); Y : b(Y, m)\}$ stands for the union of two sets: The first one contains the X-values making the conjunction $a(X, c), p(X)$ true, and the second one contains the Y-values making the conjunction $b(Y, m)$ true. An *aggregate function* is of the form $f(S)$, where S is a set term, and f is an *aggregate function symbol*. Basically, aggregate functions map multisets of constants to a constant. The most common functions implemented in ASP systems are the following:

- `#min`, minimal term, undefined for the empty set;
- `#max`, maximal term, undefined for the empty set;
- `#count`, number of terms;
- `#sum`, sum of integers.

An *aggregate atom* is of the form $f(S) \prec T$, where $f(S)$ is an aggregate function, $\prec \in \{<, \leq, >, \geq\}$ is a comparison operator, and T is a term called guard. An aggregate atom $f(S) \prec T$ is ground if T is a constant and S is a ground set. An *atom* is either a standard atom or an aggregate atom. A *rule* r has the following form:

$$a_1 \mid \ldots \mid a_n \text{ :- } b_1, \ldots, b_k, \texttt{not } b_{k+1}, \ldots, \texttt{not } b_m.$$

where $a_1, \ldots, a_n$ are standard atoms, $b_1, \ldots, b_k$ are atoms, $b_{k+1}, \ldots, b_m$ are standard atoms, and $n, k, m \geq 0$. A literal is either a standard atom a or its negation `not` a. The disjunction $a_1 | \ldots | a_n$ is the *head* of r, while the conjunction $b_1, \ldots, b_k, \texttt{not } b_{k+1}, \ldots, \texttt{not } b_m$ is its *body*. Rules with empty body are called *facts*. Rules with empty head are called *constraints*. A variable that appears uniquely in set terms of a rule r is said to be *local* in r, otherwise it is a *global* variable of r. An ASP program is a set of *safe* rules. A rule r is *safe* if both the following conditions hold: *(i)* for each global variable X of r there is a positive standard atom ℓ in the body of r such that X appears in ℓ; *(ii)* each local variable of r appearing in a symbolic set $\{Terms : Conj\}$ also appears in $Conj$.

A *weak constraint* [8] ω is of the form:

$$:\sim b_1, \ldots, b_k, \texttt{not } b_{k+1}, \ldots, \texttt{not } b_m.[w@l]$$

where w and l are the weight and level of ω. (Intuitively, $[w@l]$ is read "as weight w at level l", where weight is the "cost" of violating the condition in the body of w, whereas levels can be specified for defining a priority among preference criteria). An ASP program with weak constraints is $\Pi = \langle P, W \rangle$, where P is a program and W is a set of weak constraints.

A standard atom, a literal, a rule, a program or a weak constraint is *ground* if no variables appear in it.

Semantics. Let P be an ASP program. The *Herbrand universe* U_P and the *Herbrand base* B_P of P are defined as usual (see e.g.,[4]). The ground instantiation G_P of P is the set of all the ground instances of rules of P that can be obtained by substituting variables with constants from U_P.

An *interpretation* I for P is a subset I of B_P. A ground literal ℓ (resp., $\texttt{not}\ \ell$) is true w.r.t. I if $\ell \in I$ (resp., $\ell \notin I$), and false (resp., true) otherwise. An aggregate atom is true w.r.t. I if the evaluation of its aggregate function (i.e., the result of the application of f on the multiset S) with respect to I satisfies the guard; otherwise, it is false.

A ground rule r is *satisfied* by I if at least one atom in the head is true w.r.t. I whenever all conjuncts of the body of r are true w.r.t. I.

A model is an interpretation that satisfies all the rules of a program. Given a ground program G_P and an interpretation I, the *reduct* [16] of G_P w.r.t. I is the subset G_P^I of G_P obtained by deleting from G_P the rules in which a body literal is false w.r.t. I. An interpretation I for P is an *answer set* (or stable model [19]) for P if I is a minimal model (under subset inclusion) of G_P^I (i.e., I is a minimal model for G_P^I) [16].

Given a program with weak constraints $\Pi = \langle P, W\rangle$, the semantics of Π extends from the basic case defined above. Thus, let $G_\Pi = \langle G_P, G_W\rangle$ be the instantiation of Π; a constraint $\omega \in G_W$ is violated by an interpretation I if all the literals in ω are true w.r.t. I. An *optimum answer set* O for Π is an answer set of G_P that minimizes the sum of the weights of the violated weak constraints in G_W as a prioritized way.

Problem Solving in ASP. ASP can be used to encode problems in a declarative way. The disjunctive rules allow for expressing combinatorial problems which are in the second level of the polynomial hierarchy, and the (optional) separation of a fixed, non-ground program from an input database allows one to obtain uniform solutions over varying instances. More in detail, many problems of comparatively high computational complexity can be solved in a natural manner by following a *Guess&Check&Optimize* programming methodology [22]. This method requires that a database of facts is used to specify an instance of the problem; a set of (usually) disjunctive rules, called "guessing part", is used to define the search space; admissible solutions are then identified by other rules, called the "checking part", which impose some admissibility constraints; finally weak constraints are used to single out solutions that are optimal with respect to some criteria, the "optimize part". As an example, consider the well-known NP-hard problem called the Traveling Salesman Problem (TSP). Given a weighted graph $G = \langle V, E\rangle$, where V is the set of nodes and E is the set of edges with integer labels, the problem is to find a path of minimum length containing all the nodes of G. TSP can be encoded as follows:

$$vertex(v). \qquad \forall \mathtt{v} \in V \tag{1}$$

$$edge(i, j, w). \quad \forall(\mathtt{i}, \mathtt{j}, \mathtt{w}) \in E \tag{2}$$

$$inPath(X, Y) \mid outPath(X, Y) \text{:-} edge(X, Y, \_). \tag{3}$$

$$\text{:-} node(X), \#count\{I : inPath(I, X)\} \neq 1. \tag{4}$$

$$\text{:-} node(X), \#count\{O : inPath(X, O)\} \neq 1. \tag{5}$$

$$\text{:-} node(X), \texttt{not}\ \ reached(X). \tag{6}$$

$$reached(X) \text{:-} inPath(M, X), \#min\{N : node(N)\} = M. \tag{7}$$

$$reached(X) \text{:-} reached(Y), inPath(Y, X). \tag{8}$$

$$\text{:}\sim inPath(X, Y), edge(X, Y, W).[W@0] \tag{9}$$

The first two lines introduce suitable facts, representing the input graph G. Then, rule (3), which can be read as "each edge may or may not be part of the path", guesses a solution (a set of `inPath` atoms). Lines 4-6 select admissible paths. In particular, rule in line (4) (resp., (5)) is satisfied if exactly one arc in the solution enters in (resp., exits from) each node, indeed the body is satisfied by solutions not having a count of one. Moreover, line (6) ensures that the path traverses (say, reaches) all the nodes of G. Actually, this latter condition is obtained by checking that there exists a path reaching all the nodes of G and starting from the first node of V, say `M`. In particular, a node `X` is reached either if there is an arc connecting `M` to `X` (rule (7)), or if there is an arc connecting a reached node `Y` to `X` (rule (8)). The last line selects the solutions of minimal weight. Indeed the weak constraint in rule (9) is violated with cost `W` each time an arc of cost `W` is in the candidate solution.

3 Travel Agent Requirements

In this section we informally describe the requirements of a common problem in the tourism industry, i.e. the problem of booking in advance blocks of package tours for the next season. A travel agency usually selects a block of package tours from several travel suppliers, which may apply several discounts if predetermined amounts of their package tours are bought. In general, a critical requirement is that *the sum of prices of selected package tours must not exceed a limited budget*. This means that travel agencies are not allowed to buy all the package tours they wish. Thus, their goal is to *select package tours in order to maximize the expected earnings*. Moreover, depending on the specific needs, travel agencies might specify other preferences among the selected package tours. Those preferences are not general. On the contrary, two different travel agencies usually have different priorities among selected packages according to their experience and customer base. In the following we detail several preferences that travel agencies might specify according to their needs.

Preference of suppliers according to destination. Travel agencies might specify a preference of suppliers for package tours involving particular destinations. For instance, a supplier can be considered highly reliable for travels in Europe and unreliable for travels in other countries.

Preference of suppliers according to the type of holidays. A supplier can be considered also preferable for particular types of holidays. For instance, some suppliers are specialized in holidays involving cruises, while others are specialized in holidays involving sports activities.

Preference on package tours with the highest rating. After a trip, travelers usually evaluate their holidays by assigning a numerical score. A package tour is evaluated by looking at the rating assigned by the travelers. Thus, travel agencies give priority to the package tours with the highest ratings.

Preferences on the number of package tours to buy. According to their typology of customers travel agencies also express a preference on the number of package tours to buy. In particular, in case travel agencies obtain the same expected earnings from two or more package tours then they can maximize or minimize the number of bought package tours according to their customer base. For instance, travel agencies working with wealthy customers may prefer to buy few package tours with highest earnings, while travel agencies working with many customers may prefer to maximize the number of package tours to buy.

Preference on the amount of money to pay. Another important preference concerns the amount of money to pay. In particular, in case travel agencies obtain the same expected earnings from two or more package tours it is preferable to select package tours with the lowest prices.

4 Specification in ASP

This section illustrates the ASP program which solves the allotment problem specified in the previous section. First, the input data is described, then, the ASP rules solving the allotment problem are presented. Finally, preferences that can be specified by travel agencies are described.

4.1 Data Model

The input of the process is specified by means of the predicates described in this section. The predicates representing the facts of our encoding are the following:

- Instances of the predicate *availablePackage(pkId, supplier, destination, type, sellingPrice, purchasePrice, rating, availableQuantity)* represent stocks of available package tours in the market, where *pkId* is the identifier of the tour package, *supplier* is the identifier of the supplier selling the package tour, *destination* is the destination of the package tour, *type* is the type of holiday, *sellingPrice* is the price applied by the travel agency to their customers, *purchasePrice* is the price applied by the supplier to the travel agency, *rating* is a numerical score associated to the package tour representing the appreciation of customers for this package tour, and *availableQuantity* corresponds to the quantity of available package tours of this kind in the market.
- Instances of the predicate *requiredPackage(destination, type, minPrice, maxPrice, requiredQuantity)* represent package tours required by a travel agency, where *destination* is destination of the package tour required, *type* is the type of holiday required, *minPrice* and *maxPrice* represents the range of prices the travel agency is willing to pay for a given destination and type of holiday, and *requiredQuantity* corresponds to the quantity of required package tours of this kind.
- Instances of the predicate *discount(supplier, quantity, percentageDiscount)* represent the discount applied by suppliers if a given amount of their package tours is bought, where *supplier* represents the identifier of the supplier, *quantity* is the minimum quantity of bought package tours for applying the discount, and *percentageDiscount* is the percentage discount applied.

– The only instance of the predicate *budget(b)* represents the maximum amount of money the travel agency is willing to pay.
– Instances of the predicate *evalSupplierDestination(supplier, destination, score)* represent the evaluations of suppliers according to destination, where *supplier* is the identifier of the supplier, *destination* is the destination of the package tour, and *score* is a numerical score representing the reliability of the supplier for package tours involving the destination.
– Instances of the predicate *evalSupplierType(supplier, type, score)* represent the evaluations of suppliers according to type of holiday, where *supplier* is the identifier of the supplier, *type* is the type of holiday, and *score* is a numerical score representing the reliability of the supplier for package tours concerning the type of holiday.

4.2 Allotment Problem

In this section we describe the ASP rules used for solving the allotment problem. We follow the *Guess&Check&Optimize* programming methodology [22]. In particular, the following disjunctive rule guesses a quantity to buy for each required package:

$$\begin{aligned} buy(P,Q) \mid nBuy(P,Q) \coloneq\ & availablePackages(P,\_,D,T,SP,PP,\_,AvQ), \\ & requiredPackages(D,T,MinP,MaxP,ReqQ), \\ & 0 \leq Q \leq ReqQ, \\ & Q \leq AvQ, \\ & MinP \leq SP \leq MaxP. \end{aligned} \tag{1}$$

The guess of the quantity is limited to available package tours which are requested and their selling price is in the requested range. Then, assignments buying different quantities of the same package tour are filtered out by the following constraint:

$$\coloneq \#count\{Q,P : buy(P,Q)\} > 1, availablePackages(P,\_,\_,\_,\_,\_,\_,\_). \tag{2}$$

Suppliers may apply one or more discounts if predetermined amounts of their package tours are bought. In general several discounts are offered depending on the volume of booked packages. In this case the maximum applicable discount among them must be applied. All applicable discounts and the maximum discount among them are computed by the following rules:

$$\begin{aligned} allDiscounts(S,D) \coloneq\ & discount(S,Q1,D), \\ & \#sum\{Q,P : buy(P,Q)\} \geq Q1. \\ \\ maxDiscount(S,Disc) \coloneq\ & discount(S,\_,\_), \\ & \#max\{D : allDiscounts(S,D)\} = Disc. \end{aligned} \tag{3}$$

The predicate *allDiscounts(supplier, discount)* stores the association between the supplier and all the applicable discounts, while *maxDiscount(supplier, discount)* stores the association between the supplier and the corresponding maximum applicable discount.

Then, the prices of the package tours are updated according to the above-computed discounts. This behavior is achieved by employing the following rule:

$$\begin{aligned} discountPrices(P, SP, PPD) :- & \, availablePackages(P, S, \_, \_, SP, PP, \_, \_), \\ & maxDiscount(S, MD), \\ & PPD = PP - (PP * MD)/100. \end{aligned} \tag{4}$$

The predicate *discountPrices* stores the original selling price and the purchase price after the application of the discount for each package tour. This predicate is then used to handle a critical requirement on the budget, i.e. the sum of prices of selected package tours must not exceed a limited budget. This is expressed in ASP by the following rule:

$$\begin{aligned} :- & \#sum\{PP * Q, P : buy(P, Q), discountPrices(P, \_, PP)\} > B, \\ & budget(B). \end{aligned} \tag{5}$$

Finally, the last requirement is to maximize the earnings. This is obtained in our encoding by means of the following weak constraint:

$$\begin{aligned} :\sim & \, discountPrices(P, SP, PP), buy(P, Q), \\ & E = (SP - PP) * Q.[-E@\ell] \end{aligned} \tag{6}$$

Intuitively, when a stock of package tours is bought the solution is associated with a cost depending on the earnings obtained by buying those packages. The weight of weak constraint is negative since weak constraints expresses the minimization of the cost associated to a solution.[1] The choice of the level ℓ is explained in the following section.

4.3 Preferences (optional)

In this section, we describe preferences travel agencies might specify among the selected package tours depending on their specific needs. Different travel agencies usually have different priorities among selected package tours which are expressed in our framework by means of weak constraints. In the weak constraints we use numerical values $\ell_1, \ldots, \ell_5$ representing the levels of weak constraints. Then, an order on the preferences can be specified by properly assigning a value to those levels. The only requirement is that the level ℓ of the constraint that maximizes earnings (6) is greater than all the other weak constraints that are specified in the following.

Preference of suppliers according to destination. A travel agency might specify a preference of suppliers according to the destination of a travel. The following weak constraint expresses this preference:

$$\begin{aligned} :\sim & \, evalSupplierDestination(S, D, SC), \\ & availablePackages(P, S, D, \_, \_, \_, \_), \\ & nBuy(P, Q).[SC * Q@\ell_1] \end{aligned} \tag{7}$$

[1] ASP solvers may have undefined behaviors in presence of negative weights. A work-around is to augment the weight of the weak constraint by the maximum possible earnings.

Intuitively, when a stock of package tours is not bought a numerical penalty is associated to the solution. For each package tour which is not selected the cost of the solution is increased by the score associated to the corresponding supplier for the destination.

Preference of suppliers according to the type of holidays. Similarly, the following weak constraint expresses a preference among suppliers according to the type of holidays:

$$\begin{aligned} :\sim\ & evalSupplierType(S, T, SC), \\ & availablePackages(P, S, \_, T, \_, \_, \_, \_), \\ & nBuy(P, Q).[SC * Q@\ell_2] \end{aligned} \tag{8}$$

For each package that is not selected the solution cost is increased by the score associated to the corresponding supplier for the type of holiday. The effect is to maximize the number of package tours in the solution that are provided by preferred suppliers.

Preference on package tours with the highest rating. Travel agencies give priority to the package tours with the highest ratings. This preference is expressed by the following weak constraint:

$$:\sim availablePackages(P, \_, \_, \_, \_, \_, R, \_), nBuy(P, Q).[R * Q@\ell_3] \tag{9}$$

Here, the cost of the solution is given by the sum of ratings of package tours which are not bought. Thus we maximize the ratings of selected package tours.

Preferences on the number of package tours to buy. In case a travel agency is willing to minimize the number of packages to buy we apply the following weak constraint:

$$:\sim buy(P, Q).[Q@\ell_4] \tag{10}$$

The cost of the solution is increased by the quantity of package tours which are not bought. Otherwise, if a travel agency is willing to maximize the number of packages to buy we apply the following weak constraint:

$$:\sim nBuy(P, Q).[Q@\ell_4] \tag{11}$$

Here, the cost of the solution is increased by the quantity of package tours which are bought. Note that weak constraints (10) and (11) are never applied together since travel agencies either maximize or minimize the number of package tours to buy.

Preference on the amount of money to pay. Finally, travel agencies may also want to minimize the amount of money to pay. Note that this is different from the earnings, since in this case travel agency minimizes the purchase prices without considering their selling prices. This behavior is employed by the following weak constraint:

$$:\sim discountPrices(P, \_, PP), buy(P, Q).[PP * Q@\ell_5] \tag{12}$$

Intuitively, the cost of the solution depends on sum of prices of package tours which are bought. Hence, this has the effect to minimize the price of package tours in the solution.

Specification of preferences. As stated in Section 3, the preferences depend on the specific needs of travel agencies, and can be applied selectively by simply adding or ignoring some of the weak constraints described in Section 4.3. Moreover, a travel agent must also specify a layering of preferences by properly assigning values to $\ell_1, \ldots, \ell_5$. As an example, consider a travel agent that wants to give highest priorities on package tours with the highest ratings; and then maximize the number of packages to buy. In the encoding those preferences are specified by considering weak constraints (9) and (11) and by assigning integer values to the levels such that $\ell_3 > \ell_5$, e.g., $\ell_3 = 2$, and $\ell_5 = 1$.

5 Empirical Validation

We validated our ASP-based solution running a preliminary experiment on real-world data provided by the partners of the iTravelPlus project. In particular, we obtained an instance of a database of package tours querying the database of the iTravel+ system and properly encoding it by means of ASP facts. Moreover, we generated a specification of the requested package tours by running a mining services of the same system that generates a prediction based on the package tours sold in the past. Finally, we randomly generated a number of additional requirements to test the effects of the optional preferences of our solution. Concerning the ASP solver we used WASP [2]. (For the sake of completeness, we report that we also tried the ASP solver CLASP [18] obtaining similar performance.) The system was run on a four core Intel Xeon CPU X3430 2.4 GHz, with 16 GB of physical RAM, each execution was limited to 600 seconds.

The performance of the system for different sizes of the available packages is reported in Table 1. In particular, the first column reports the sizes of the considered DBs. We considered the original database (labeled DBx1) and then we consider two more settings containing the first half of the same database (labeled DBx0.5), a generated instance (labeled DBx2) having twice of the facts from DBx1 obtained adding more suppliers. The second column reports the minimum and the maximum available package tours among the instances considered. The number of considered instances is reported in the third column together with the number of instances in which the system found a (sub-optimal) solution (fourth column) and the number of instances in which the system found the optimum solution (fifth column). The sixth column reports the sum of execution times (in seconds) elapsed for finding the optimum solution of the allotment problem without optional preferences. The seventh column reports the sum of time in seconds of finding the optimum solution of the allotment problem where all preferences are enabled. As first observation we note that the system provides a solution for all the instances considered within 10 minutes. The provided solutions are usually

Table 1. Performance of the system for different available package tours.

	Available Pkgs (Min-Max)	**#inst**	**#solved**	**#optima**	**Time (no pref.)**	**Time (all pref.)**
DBx0.5	216-291	30	30	30	0.6	0.8
DBx1	445-584	30	30	26	48.5	147.8
DBx2	963-1093	30	30	14	11.7	33.41

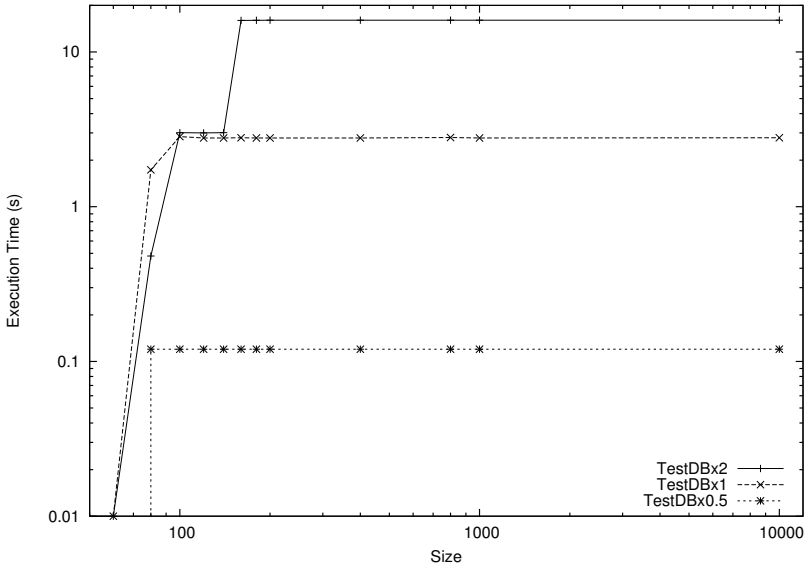

Fig. 1. Scalability w.r.t. the number of available package tours.

either close to the optimum ones or are optimal but the system is not able to prove their optimality within the timeout. Moreover, when we consider half size of the DB size the system finds always the optimum solution, while for the original size of the DB this is the case in the 87% of the instances, which is a performance considered fairly acceptable by our project partners. The performance is still good in the case we double the size of the original DB, since the system is able to find the optimum for about half of the instances. In addition, we also observe that adding the preference does not reduce either the number of solved instances nor the number of instances in which the optimum solution is found. As expected, we observe a constant slow down in the solving time, which is approximately three times higher than the one measured with no preferences.

It is worth pointing out that the performance of the system does not heavily depend on the quantity of available packages. In fact, rule (1) in Section 4.2 filters out all the package tours which are not required. In order to confirm this observation, we increased the available quantities for each package tours in the database by several factors. The result is reported in Figure 1, in which, for a particular size of the DB, a point (x, y)

Table 2. Performance of the system for different periods (in months).

Period	Required Pkgs (Min-Max)	#inst	#solved	#optima	Search Space (avg)
2m	79-94	30	30	30	2^{27}
4m	174-182	30	30	24	2^{54}
6m	345-365	30	30	16	2^{110}

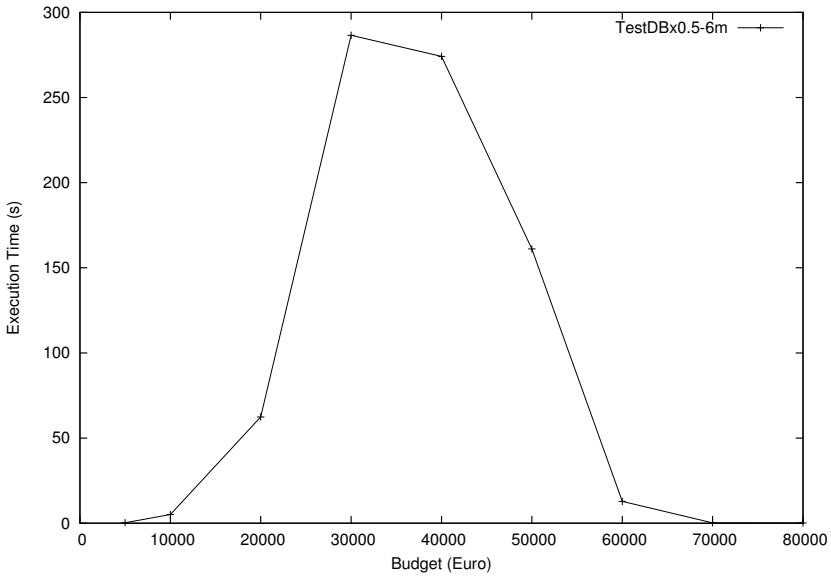

Fig. 2. Performance of the system depending on the budget.

represents the solving time y where the availability of package tours is x percent of the original size. The graph shows that the execution times grow until the offered packages are 120% more than in the original DB, and then performance has a constant trend that is not dependent on the quantity of available packages.

Table 2 reports on the performance of the system for different periods of request packages. In particular, the first column reports the considered period expressed in months. The second column reports the minimum and the maximum required package tours among the instances considered. The number of considered instances is reported in the third column together with the number of instances in which the system found a solution (fourth column) and the number of instances in which the system found the optimum solution (fifth column). The last column reports the average search space for the considered instances. Also in this case, the system provides a solution for all the instances considered. Moreover, when we consider a period of 2 months the system also finds always the optimum solution, while if we considered a period of 4 months this is the case in the 80% of the instances. It is worth pointing out that, travel agencies usually book package tours for one season, thus they consider a period of at most 3-4 months. Nonetheless, the performance is still good in the case we consider a period of 6 months, since the system is able to find the optimum for about half of the instances.

Finally, another observation concerns the budget allowed by the travel agency, since the hardness of the instances depends on this parameter. In fact, is it easy to see that instances with very low (resp. high) budgets w.r.t. to the one needed to fulfill the request are likely easy, since they correspond to over-constrained (resp. under-constrained) problems where the solution is to buy no package tours (resp. to buy all required pack-

age tours). Thus, we also analyzed the behavior of the system in case we consider different budgets. The result is reported for DBx0.5 and a request of 6 months in Figure 2, in which a point (x, y) represents the solving time y if the budget is limited to x euro. The trend of the system confirms our expectations, since the instance is trivially solved when the budget is enough either to buy nothing or to buy everything. The maximum hardness is reached when the allotted budget can cover about 40% of the request in our experiment, a setting that in real-world instances is not that common, since the budget is usually enough to cover most of the requests.

6 Related Work

In the literature there are solution to many e-tourism systems challenges including: package tours search and assemblage, automatic holiday advisors, modeling of general purpose ontologies of the touristic domain [23, 9, 10, 25, 26, 14, 6], etc. These studies do not focus –to the best of our knowledge– on helping travel agents in the act of selecting package tours to be traded with service suppliers in the future market.

Concerning the applications of ASP, we mention that it has been used to develop several industrial applications [20, 28] and, in particular, it has already been exploited in an e-tourism system [27]. Nonetheless, the problem considered in [27] was to identify the package tours that best suit the needs of a customer of an e-tourism platform; thus, [27] approaches a different problem of the one considered here. This paper presents the first attempt to exploit ASP for assisting tour operators in the allotment of packages.

It is important pointing out that other approaches, such as Constraint Programming (CP) [29] and Mixed-Integer Programming (MIP) [13], could also be good candidates for solving the problem considered in this paper. Despite it would be interesting to investigate whether other solving technologies such as CP or MIP are also successfully applicable, the goal of this research was to demonstrate that ASP can be used for solving the allotment problem in practice, and our ASP-based solution satisfied the stakeholder partners of the iTravelPlus project. Comparison with other approaches can be an interesting research goal to be developed in a separate work.

For the sake of completeness, we also mention a different way of dealing with the problem of allotment [30]. This approach aims at acquiring directly and on-demand from hotel management services the information about the hotel rooms and facilities that suit the request of a tour operator, so to avoid the allotment problem using agent technologies [30]. This is clearly a radically different approach from ours that aims at optimizing the pre-booking of allotments for an entire period of time.

7 Conclusion

In this paper we described an application of Answer Set Programming to the problem of allotment in travel industry. We have formalized an allotment problem that abstracts the requirements of a real travel agent, by means of an ASP programs. Since ASP programs are executable specifications we also obtained a prototypical implementation of a tool for supporting a travel agent in selecting the packages to be traded for next season. We experimented with our implementation on instances of the problem made of real-word

data provided by the travel agency Top Class s.r.l. The preliminary results that we obtained are promising in terms of performance. Our ASP program will be included as an advanced reasoning service of the e-tourism platform developed under the iTravelPlus project by the Tour Operator Top Class s.r.l. and the University of Calabria.

As far as future work is concerned, we plan to study the computational properties of the allotment problem, and to extend our formulation by modeling additional preference criteria on the solutions. In this respect we will investigate the adoption of more general frameworks for expressing preferences [17, 7]. The automatic allotment tool will be integrated in the system developed under the iTravelPlus project that aims at developing several services for tour operators.

Acknowledgments. We are grateful to DLV SYSTEM s.r.l. for its support in the development of the system and to Denise Angilica, Gianluigi Greco, and Gianni Laboccetta for fruitful discussions on the specification of the problem.

References

1. Alviano, M., Calimeri, F., Charwat, G., Dao-Tran, M., Dodaro, C., Ianni, G., Krennwallner, T., Kronegger, M., Oetsch, J., Pfandler, A., Pührer, J., Redl, C., Ricca, F., Schneider, P., Schwengerer, M., Spendier, L.K., Wallner, J.P., Xiao, G.: The fourth answer set programming competition: Preliminary report. In: Cabalar, P., Son, T.C. (eds.) LPNMR. LNCS, vol. 8148, pp. 42–53. Springer (2013)
2. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Son, T.C. (eds.) LPNMR. LNCS, vol. 8148, pp. 54–66. Springer (2013)
3. Balduccini, M., Gelfond, M., Watson, R., Nogeira, M.: The USA-Advisor: A Case Study in Answer Set Planning. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR. LNCS, vol. 2173, pp. 439–442. Springer (2001)
4. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
5. Baral, C., Gelfond, M.: Reasoning Agents in Dynamic Domains. In: Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 257–279. Kluwer Academic Publishers (2000)
6. Barta, R., Feilmayr, C., Pröll, B., Grün, C., Werthner, H.: Covering the semantic space of tourism: an approach based on modularized ontologies. In: CIAO. pp. 1–8. ACM (2009)
7. Brewka, G., Delgrande, J.P., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: Bonet, B., Koenig, S. (eds.) AAAI. pp. 1467–1474. AAAI Press (2015)
8. Buccafurri, F., Leone, N., Rullo, P.: Enhancing Disjunctive Datalog by Constraints. IEEE Transactions on Knowledge and Data Engineering 12(5), 845–860 (2000)
9. Cardoso, J.: Combining the semantic web with dynamic packaging systems. In: AIKED. pp. 133–138. World Scientific and Engineering Academy and Society (2006)
10. Cardoso, J.: Developing an owl ontology for e-tourism. In: Semantic Web Services, Processes and Applications, pp. 247–282. Springer (2006)
11. Castellani, M., Mussoni, M.: An economic analysis of tourism contracts: Allotment and free sale*. In: Advances in Modern Tourism Research, pp. 51–85. Springer (2007)
12. Cooper, C., Fletcher, J., Fyall, A., Gilbert, D., Wanhill, S.: Tourism: Principles and Practice. Financial Times Management, 4 pap/pas edn.

13. Dakin, R.J.: A tree-search algorithm for mixed integer programming problems. The Computer Journal 8(3), 250–255 (1965)
14. Dogac, A., Kabak, Y., Laleci, G., Sinir, S., Yildiz, A., Kirbas, S., Gurcan, Y.: Semantically enriched web services for the travel industry. SIGMOD Rec. 33(3), 21–27 (2004)
15. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Trans. Database Syst. 22(3), 364–418 (Sep 1997)
16. Faber, W., Leone, N., Pfeifer, G.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence 175(1), 278–298 (2008)
17. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-criteria optimization in answer set programming. In: Gallagher, J.P., Gelfond, M. (eds.) Technical Communications of ICLP. LIPIcs, vol. 11, pp. 1–10. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
18. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. 187, 52–89 (2012)
19. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
20. Grasso, G., Leone, N., Manna, M., Ricca, F.: Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond, LNAI, vol. 6565. Springer (2011)
21. Gurcaylilar-Yenidogan, T., Yenidogan, A., Windspergerc, J.: Antecedents of contractual completeness: the case of tour operator-hotel allotment contracts. Procedia - Social and Behavioral Sciences 24(0), 1036 – 1048 (2011), International Strategic Management Conference
22. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM Trans. Comput. Log. 7(3), 499–562 (Jul 2006)
23. Maedche, A., Staab, S.: Applying semantic web technologies for tourism information systems. In: Weber, K., Frew, A., (eds.), M.H. (eds.) "International Conference for Information and Communication Technologies in Tourism". Springer (2002)
24. Manna, M., Ricca, F., Terracina, G.: Consistent query answering via ASP from different perspectives: Theory and practice. TPLP 13(2), 277–252 (2013)
25. Martin, H., Katharina, S., Daniel, B.: Towards the semantic web in e-tourism: can annotation do the trick? In: European Conference on Information System (2006)
26. Prantner, K., Ding, Y., Luger, M., Yan, Z., Herzog, C.: Tourism ontology and semantic management system: State-of-the-arts analysis. In: International Conference WWW/Internet. IADIS (2007)
27. Ricca, F., Dimasi, A., Grasso, G., Ielpa, S.M., Iiritano, S., Manna, M., Leone, N.: A logic-based system for e-tourism. Fundam. Inform. 105(1-2), 35–55 (2010)
28. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the gioia-tauro seaport. TPLP 12(3), 361–381 (2012)
29. Rossi, F., Beek, P.v., Walsh, T.: Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA (2006)
30. Withalm, J., Karl, E., Fasching, M.: Agents solving strategic problems in tourism. In: Fesenmaier, D., Klein, S., Buhalis, D. (eds.) Information and Communication Technologies in Tourism 2000, pp. 275–282. Springer (2000)