Package Planning with Graphical Models

Y. Xiang and **M. Janzen**
University of Guelph, Canada

Abstract

We consider a novel class of applications where a set of activities conducted by a group of people over a time period needs to be planned, taking into account each member's preference. We refer to the decision process that leads to such a plan as *package planning*. The problem differs from a number of well-studied AI problems including standard AI planning and decision-theoretic planning. We present a computational framework using a combination of activity grammar, search, and graphical models. We show that the computation is tractable when the problem parameters are reasonably bounded.

Introduction

We consider a novel class of applications where a set of activities conducted by a group of people over a time period needs to be planned taking into account the preference of each member. We refer to the decision process that leads to such a plan as *package planning*. Examples of package planning include the following:

A family or a group of college students living in a dormitory shares a kitchen and shops for grocery collectively. The items to buy depend on a meal plan between consecutive shopping trips. Some meals are individual, such as a snack, but others are collective, such as a family supper. Each member has his or her preference of what and when to eat or drink. There are also budget constraints. Automated meal planning plans meals that satisfies these preferences and generates a shopping list.

Cable and satellite TV channels supply many programs daily. A program may be broadcast when a viewer is busy. Even when the viewer is free, he or she may prefer to record the program through a VCR and view it later so that uninteresting commercials can be fast forwarded. Members may prefer different viewing times during the day, but may view a program together sometimes. Each member has different tastes for different programs. A package planning system can select programs to record for all members taking into account individual preference in content and time, and to plan who views what and when.

A third example is to plan a group tour. It has a main travel path (e.g., via a cruise ship) but at different stops subgroups may choose to tour different places (e.g., some go shopping and others visit museums). Planning ahead and prearrangement of transportation will ensure a smooth and more cost-effective tour. A package planning system can select both the main tour route and stops for the whole group, as well as individual tour diversions for subgroups.

We are unaware of existing work that addresses package planning computationally. In this paper, we present a formal framework for solving this problem.

Package Planning Domains

We refer to the group of people involved as a *family* and each member of the family as a *user*. Formally, we denote the family by a set F of users.

The activities of users to be planned occur over a finite time period. The period is represented by *non-overlapping* intervals (*not necessarily adjacent*) of some granularity appropriate to the problem domain. For any given user, no more than one activity occurs in each interval. Formally, we denote the planning time period by a sequence of intervals $T = \{t_0, t_1, ...\}$. In meal planning, we can have

$$T = \{Mon\ morning, Mon\ noon, Mon\ afternoon, Mon\ evening, ...\}.$$

Possible activities of users are denoted by a set A. Each element of A is referred to as an *activity*, such as having a turkey meal, viewing a given TV program, or touring a given place.

A *package* is a set K where each element is a triplet (a, t, r) with $a \in A$, $t \in T$ and $r \subseteq F$. We refer to (a, t, r) as a *planned activity* for users r. Note that r could be a single user, a subset of the family, or the whole family. For instance, in meal planning, *(turkey meal, Sunday evening, F)* represents a turkey meal as the supper for the family on Sunday. In TV planning, *(Titanic, Saturday afternoon, Lisa)* stands for a Saturday afternoon activity for Lisa, where we have written $\{Lisa\}$ as $Lisa$ for simplicity. Similarly, *(Great Wall, Monday, David)* has the obvious meaning in tour planning. Note that it is not necessary for K to contain a planned activity for each $t \in T$ and for each user $r \in F$ (a user r may prefer a *null* activity in a given interval t). The collection of all possible packages for a family is denoted by $\mathcal{K}$ and called the *package space*.

We assume that each user has his/her own preference about different packages. For each user r, a *utility* function μ_r is defined over the package space $\mathcal{K}$ and in the range $[0, 1]$. We denote the set of utility functions, one for each user in F, by μ_F.

Def 1 summarizes the above to define a package planning domain and a package planning problem.

Definition 1 *A* `package planning domain` *consists of a tuple* $(F, T, A, \mathcal{K}, \mu_F)$. *$F$ is a set of users. T is a set of nonoverlapping time intervals. A is a set of activities. $\mathcal{K}$ is the collection of all possible packages over F, T and A, and μ_F is a set of user utility functions. A* `package planning problem` *is to find a package K^* in $\mathcal{K}$ such that*

$$\frac{1}{|F|}\sum_{r\in F}\mu_r(K^*) = \max_{\mathcal{K}}\frac{1}{|F|}\sum_{r\in F}\mu_r(K), \qquad (1)$$

where maximization is over K of length $|T|$.

That is, the goal of package planning is to find a package that best suits all users in a family. The average utility among users is assumed, although other alternatives are possible (e.g., weighted average).

Individual Preference

A package consists of a set of planned activities. Some are planned for only one user and some for several users. In order to evaluate alternative packages effectively, we assume that a user's preference towards a given package is independent of other users' preferences towards the same package. That is, we do not allow the preference such as "If Lisa likes it, then I like it too." We refer to this assumption as *independence of individual preference*. We present its implication below.

Given a planned activity and a user, we define the user's perspective for such an activity as a *projection* of the triplet. For example, the projection of *(turkey meal, Sunday evening, F)* onto the user Lisa is *(turkey meal, Sunday evening,* $\{Lisa\}$*)*. Each planned activity can be projected to each user with the convention that (a, t, r) projected to r', where $r \cap r' = \emptyset$, yields a *null* triplet. Given a package K and a user r, the package obtained by projecting each planned activity in K to r is called the *projection* of K to r and denoted by K_r. Note that the projection of K to r is a sequence of planned activities for r only. Given $\mathcal{K}$ and a user r, the projection of $\mathcal{K}$ to r can be similarly defined, which we refer to as the *projected package space* and denote by $\mathcal{K}_r$.

Each user in a family has his/her own preference over different activities and how they are sequenced. Lisa may like fried chicken but David does not. Both may like turkey. David may have turkey three days in a row but Lisa only wants to have it once in a while. Given independence of individual preference, for each user r, its utility function μ_r is defined over the projected package space $\mathcal{K}_r$. Therefore, the package planning problem can be reformulated as to find a package K^* over T such that

$$\frac{1}{|F|}\sum_{r\in F}\mu_r(K_r^*) = \max_{\mathcal{K}}\frac{1}{|F|}\sum_{r\in F}\mu_r(K_r). \qquad (2)$$

In summary, independence of individual preference allows more focused and efficient computation.

Distributed Package Planning

The package planning task is carried out by multiple agents[1] consisting of one *user agent* for each user and a *manager* agent. We denote the agent for user r as g_r and the manager as g.

The manager g is responsible for generating alternative packages from $\mathcal{K}$. For each package K, g sends K_r to each g_r. User agent g_r maintains the utility function μ_r and uses it to evaluate K_r. The result $\mu_r(K_r)$ is then sent back to g. When g receives the reply from each g_r, it integrates replies into a final evaluation of K. The best K is selected after enough alternatives are evaluated.

These agents work for different principals. A meal manager is an agent in a *smart* house. A TV program manager is another agent in the smart house. Each user agent g_r serves one user r in the family. The meal planning agent for user r may be the same agent who does also TV planning for r, since the tasks are quite similar. This same agent may also do tour planning for r. On the other hand, a tour manager agent g works for a travel agency and provides service to users from many different households. The assumption that agents work for different principals implies a distributed (versus centralized) implementation of package planning.

[1]We use the term *agent* in a broad sense.

Space Reduction with Grammar

To select the best package, each projected package needs to be evaluated by a corresponding user agent. The desirability of a (projected) package to a user depends partially on the desirability of individual activities in the package (and partially on how the activities are sequenced, which we consider in Sections and). We consider the complexity of the activity space here:

In meal planning, a meal may consist of a number of foods. For example, a dinner may consist of steak as the main dish, baked potato as one side and steamed broccoli as another, Caesar salad, wine as a drink, and ice cream as desert. Denote the set of all foods by Fd and its cardinality by $|Fd|$, the number of possible meals is $O(2^{|Fd|})$, the cardinality of the *activity space* (consisting of both *primitive activities* such as a food and *composite activities* such as a multi-food meal). As $|Fd|$ can be well over a hundred, this is clearly intractable to process. Similar situations arise in TV planning and tour planning as well. In general, we need to consider composite activities in package planning.

To reduce the number of composite activities to be considered, we observe that composite activities that users prefer follow certain structures. By requiring a composite activity to follow the structural constraint, we can focus the computation on preferable activities and reduce the number of activities to be considered significantly. We explore the use of a grammar to specify such structures. An example meal grammar is shown in Fig 1 in Backus-Naur form (BNF) and the above steak dinner is a possible meal generated from the grammar. We restrict the grammar to disallow the head of a rule to appear also at the tail.

$< start > \longrightarrow < breakfast > | < dinner >$
$< breakfast > \longrightarrow$ breakfast cereal
$< dinner > \longrightarrow < steak\ dinner > | < stir\ fry\ dinner >$
$< steak\ dinner > \longrightarrow$ steak $< side > < side > < drink > < salad > < desert >$
$< stir\ fry\ dinner > \longrightarrow$ stir fry $< drink > < desert >$
$< side > \longrightarrow$ steamed broccoli | baked potato | french fries
$< drink > \longrightarrow$ wine | Coca Cola | water
$< salad > \longrightarrow$ garden salad | Caesar salad | Greek salad
$< desert > \longrightarrow$ chocolate cake | ice cream | fresh fruit

Figure 1: An example meal grammar

Such a grammar can be equivalently encoded into an AND-OR tree Y. Denote the maximum depth of Y by d. When Y contains only OR nodes, a composite activity is a leaf in Y. If the maximum branching factor for OR nodes is b, the number of leaves is $O(b^d)$. When Y contains both AND and OR nodes, there is no alternative at an AND node and a composite activity is a set of leaves. If the maximum branching factor for AND nodes is c and the number of AND nodes in a maximum path from the root to a leaf is m, then the number of composite activities is between $O(b^{d-m})$ and $O(b^{d-m} * c^m)$ depending on how close the AND nodes are from the root. When d and m are small, $O(b^{d-m} * c^m)$ is tractable. For the example of Fig 1, $b = 4$, $c = 6$, $d = 4$, $m = 1$, and $|Fd| = 15$. The

grammar thus reduces the complexity from the order of 32768 to 384.

Package Search

The desirability of a (projected) package to a user also depends on how activities in the package are sequenced. For many, eating the same foods in a row is undesirable. Hence, package selection must consider the dependence among activities at different time intervals. In other words, we need to consider the desirability of both individual activities in a package and them as a whole. However, given A and T, the number of packages is $O(|A|^{|T|})$: an intractable search space.

We assume that the desirability of an activity depends only on the activities in the past but not those in the future. We refer to this assumption as *future independence of activities*. In other words, the contribution of a partial package (between t_0 and t_k) to the utility of the package cannot be changed by the activities from t_k onwards. In Section , we present a graphical model representation of the dependence between an activity at consideration and activities in the history. As will be seen, this assumption simplifies the representation.

Although the desirability of an activity does not depend on future activities, once the activity is determined, it does affect the desirability of future activities. Therefore, the future independence assumption does not reduce the complexity of the package search space. Greedy search is commonly used in learning probabilistic graphical models, e.g., (Cooper & Herskovits 1992; Heckerman, Geiger, & Chickering 1995). To make the computation tractable, we apply greedy search as an approximation method. We search for the best activity in each time interval, considerating dependence only on the past activities. For example, to plan the fifth breakfast of a week, we only evaluate each activity in A given the meals that have been planned so far for the first four days (and the preceding meal history). This allows us to reduce the package search space to the order of $O(|A|\ |T|)$. The effect of this choice is the replacement of the package search space $\mathcal{K}$ by a subspace $\mathcal{K}' \subset \mathcal{K}$ in Eqn (2). Preliminary experiments showed no significant impact of this replacement on the quality of the planning outcome.

Cost and Desirability

The utility of an activity to a user depends on a number of factors. We group these factors grossly into *desirability* and *cost*, where desirability concerns nothing about cost. We denote the cost related utility function of a user r by c_r and its desirability related utility function by u_r. A meal that is delicious and expensive has both high desirability and high cost. Its overall utility depends on the relative importance of desirability and cost to the user(s), i.e., $\mu_r = \alpha_r\ c_r + (1 - \alpha_r)\ u_r$, where a constant $(0 < \alpha_r < 1)$ encodes the relative importance. A package planning problem can then be stated as finding a package K^* such that

$$\frac{1}{|F|}\sum_{r\in F}\mu_r(K_r^*) = \max_{\mathcal{K}'}\frac{1}{|F|}\sum_{r\in F}[\alpha_r c_r(K) + (1-\alpha_r)u_r(K_r)], \qquad (3)$$

Note that the package search is over subspace $\mathcal{K}'$. We have assumed linear utility combination, although other combinations are possible (Keeney & Raiffa 1976).

We assume that $u_r(K_r)$ can be aggregated from the utility of each planed activity (a', t, r) in K_r, which we denote as $u_r(a', t, r)$. Furthermore, $u_r(a', t, r)$ can be aggregated from the utility of each primitive activity a contained in the composite activity a'. We refer to this assumption as *utility decomposition* (see (Keeney & Raiffa 1976; Bacchus & Grove 1995) for more general discussion). For example, the utility of a meal package K_r can be aggregated from the utility of each meal for r and the utility of each meal can be aggregated from the utility of each food contained. Our framework does not dictate the form of aggregation, although averaging can be a simple option. Note also that the aggregation does not hinder representation of utility dependence among activities, as will be clear below.

For each user r, a user agent g_r maintains u_r. Based on the above method, g_r must maintain user r's desirability for each primitive activity a. The desirability of a primitive activity can depend on how much the user likes the activity in general, what other primitive activities are contained in the same composite activity, what activities were conducted in the recent past, and other context factors. For example, the desirability of a food in a particular meal depends on the user's taste about the food, other foods in the same meal, what foods the user consumed recently, and the time of the day and the season as context factors.

We represent each user's desirability about activity a as a utility function from a set of factors to $[0, 1]$. Let u_r^i be the desirability of user r for the ith primitive activity a^i. Let π^i be the set of factors that u_r^i depends on. Then u_r^i can be written as $u_r^i(a^i|\pi^i)$. For example, suppose $a^i = french\ fries\ (ff)$ and $\pi^i = \{backed\ potato\ (bp), steak\ (s), wine\ (w)\}$. Then "never serve backed potato and french fries together" can be expressed by $u_r^i(ff = y|bp = y, s, w) = 0$. "French fries are better with steak when wine is served as a drink" can be represented by $u_r^i(ff = y|bp = n, s = y, w = n) = 0.6$ and $u_r^i(ff = y|bp = n, s = y, w = y) = 0.8$.

Desirability of an activity partially depends on whether it was performed recently. A naive representation of this dependence would include in π^i a variable corresponding to the activity in each past time interval. As a result, $|\pi^i|$ grows linearly as the desirability of a^i is evaluated for later time intervals, and $u_r^i(a^i|\pi^i)$ grows exponentially.

We present an alternative representation with the following characteristics: First, we consider dependence of an activity on its own historic performance as well as that of other activities. For instance, whether *chocolate ice cream* is desirable for an afternoon snack depends not only on whether it was the dessert in lunch, but also on whether the user had *hot chocolate* recently. Second, we group historic performance of a relevant activity into, say, *just*, *recently*, and *a while ago*. The exact time period covered under each group depends on the domain. In meal planning, for instance, *just* may cover the last two days and *recently* may cover days beyond the last two and within the last seven days. Hence, if a food a^i has been planned for r once in the last two days, then the variable $just - had - a^i$ has the value $some$. This representation allows historical dependence of each primitive activity to be encoded in a compact and stable form. Although the representation is an approximation relative to the naive alternative, it corresponds intuitively to human preference patterns.

Agent g_r maintains $u_r^i(a^i|\pi^i)$ but may not know its parameters with certainty. We encode g_r's uncertain knowledge about r's preference as a probability distribution over possible utility functions: $P(u_r^i(a^i|\pi^i))$. To facilitate package evaluation (see Section), we approximate the range of values of function u_r by a set U of discrete utility values, e.g., $U = \{0, 0.25, 0.5, 0.75, 1\}$. The uncertainty about the above $u_r^i(a^i|x_0, y_0)$ may then be expressed as

$$P(u_r^i(a^i|x_0, y_0) = 0) = 0, \quad P(u_r^i(a^i|x_0, y_0) = .25) = 0.05,$$
$$P(u_r^i(a^i|x_0, y_0) = .5) = 0.2, \quad P(u_r^i(a^i|x_0, y_0) = .75) = 0.6,$$
$$P(u_r^i(a^i|x_0, y_0) = 1) = 0.15.$$

The representation admits estimated preference and allows its uncertainty to be reduced gradually by learning (see Section). We simplify notation $P(u_r^i(a^i|\pi^i))$ as $P(u_r^i|\pi^i)$, i.e.,

$$P(u_r^i = u^{i0}|x_0, y_0) = 0, \quad P(u_r^i = u^{i1}|x_0, y_0) = 0.05,$$
$$P(u_r^i = u^{i2}|x_0, y_0) = 0.2, \quad P(u_r^i = u^{i3}|x_0, y_0) = 0.6,$$
$$P(u_r^i = u^{i4}|x_0, y_0) = 0.15.$$

where u^{i1} denotes the utility value 0.25.

Evaluation of $c_r(K)$ is performed by first evaluating the cost of K relative to r and then mapping the cost to utility by c_r. User preference on quantity and cost information can be attached to the grammar in Section for cost evaluation in a straightforward way. The cost evaluation is then a simple summation.

Package Evaluation

By Eqn (3), evaluating desirability of a projected package amounts to computing $u_r(K_r)$, as defined below:

Definition 2 *Let K_r be a package projected to user r. Its desirability based utility of K_r is*

$$u_r(K_r) = \frac{1}{\sum_{a^i@t} 1} \sum_{a^i@t} [\sum_j u^{ij} P(u_r^i = u^{ij}|K_r)], \quad (4)$$

where $a^i@t$ is a primitive activity in K_r at time t, u^{ij} is a discrete utility value.

The summation $\sum_{a^i@t}$ is over each primitive activity at each time interval and $\sum_j$ is over each discrete utility value. Hence, $\sum_{a^i@t} 1$ denotes the number of primitive activities at time t.

The evaluation can be performed using a graphical model which we term as a *package evaluation net* (PEN). It is a utility network (Bayesian network augmented with multiple utility nodes) applied to package planning. A PEN is created by g_r in order to evaluate a given package.

Definition 3 *Let K_r be a package projected to user r and $\{P(u_r^i|\pi^i)\}$ be the uncertain desirability of r defined over each primitive activity a^i. A* package evaluation net *for K_r is a triplet (V, G, P). V is a set of discrete variables consisting of only the following:*

- *For each planned activity (a, t, r) and each primitive activity a^i in a, there are two variables in V: desirability $u^i@t \in \{u^{i0}, u^{i1}, ...\}$ and expected utility $w^i@t \in \{y, n\}$.*[2]
- *For each $u^i@t$, there is a subset $\pi^i@t \subset V$ that is a copy of π^i (as defined by $u_r^i(a^i|\pi^i)$) for time t.*

[2]The space $\{y, n\}$ can be alternatively expressed as $\{1, 0\}$. It's a technique to allow utility to be handled in the same form as probability.

G is a DAG whose nodes map one-to-one to variables in V and are labeled accordingly. Its topology admits semantics that d-separation *(Pearl 1988) implies conditional independence. Its arcs are defined as follows:*

- *For each pair of $u^i@t$ and $w^i@t$, there is an arc from $u^i@t$ to $w^i@t$.*
- *For each $x \in \pi^i@t$, there is an arc from x to $u^i@t$.*

P is a set of probability distributions:

- *Each $u^i@t$ is assigned $P(u^i@t|\pi^i@t) = P(u_r^i|\pi^i)$.*
- *Each node without parents is assigned a uniform distribution.*
- *Each $w^i@t$ is assigned*

$$P(w^i@t = y|u^i@t = u^{ij}) = u^{ij},$$

$$P(w^i@t = n|u^i@t = u^{ij}) = 1 - u^{ij}.$$

Consider a trivial projected package for Lisa's lunch and afternoon snack:

$$K_{Lisa} = \{(hotdog\ meal, noon, Lisa),$$

$$(chocolate\ ice\ cream, afternoon, Lisa)\}.$$

Suppose that the hotdog meal is composed of *sausage*, *hotdog bun* and *hot chocolate* drink. Fig 2 shows the graph structure G of its PEN. It is trivial because a package in practice involves many time intervals and hundreds of primitive activities.

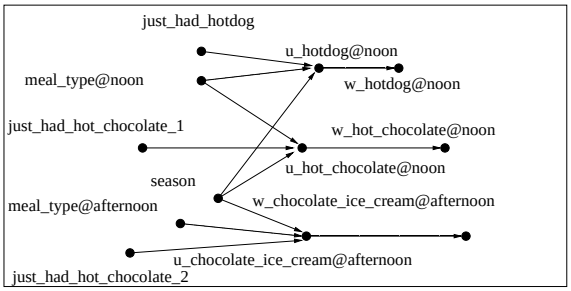

Figure 2: A trivial PEN where $meal\_type@noon$ indicates the meal type for an activity at noon and $just\_had\_hotdog$ indicates immediate eating history with hotdog.

Syntactically, a PEN is equivalent to a Bayesian net (Pearl 1988) as shown below:

Proposition 4 *A package evaluation net specified by Def 3 is a syntactically valid Bayesian net.*

Proof: By Def 3, G is a DAG. Each node in G corresponds to a variable. Each node x with parents $\pi(x)$ in G is assigned a conditional probability distribution in the form $P(x|\pi(x))$. □

Due to Proposition 4, belief propagation can be performed in a PEN. The following Algorithm 5 allows utility $u_r(K_r)$ to be computed using a PEN based on any belief propagation algorithm (D'Ambrosio 1999):

Algorithm 5 *Input: A projected package K_r and its corresponding PEN $S = (V, G, P)$.*

1 for each node $u^i@t$ in S
2 for each parent node x of $u^i@t$
3 instantiate x according to K_r;
4 perform belief propagation in the resultant S;
5 for each node $w^i@t$ in S
6 retrieve updated probability $P'(w^i@t = y)$;
7 return $E = \frac{1}{\sum_{w^i@t} 1} \sum_{w^i@t} P'(w^i@t = y)$;

The algorithm starts by instantiating parent variables of $u^i@t$, which corresponds to entering observations in standard probabilistic reasoning with graphical models (D'Ambrosio 1999). Belief propagation is then performed using any inference methods. Afterwards, the updated probability in each node $w^i@t$ is retrieved and integrated as the result. Theorem 6 establishes the correctness.

Theorem 6 *Let K_r be a package projected to user r and $S = (V, G, P)$ be the PEN of K_r. After Algorithm 5 halts, its return value satisfies $E = u_r(K_r)$ as specified by Def 2.*

Proof: From Def 2, we need to show

$$E = \frac{1}{\sum_{w^i@t} 1} \sum_{w^i@t} P'(w^i@t = y)$$

$$= \frac{1}{\sum_{a^i@t} 1} \sum_{a^i@t} [\sum_j u^{ij} P(u_r^i = u^{ij}|K_r)] = u_r(K_r).$$

By Def 3, for every $a^i@t$ in K_r, there is a corresponding $w^i@t$. Hence, it suffices to show that the following holds for each $w^i@t$: $P'(w^i@t = y) = \sum_j u^{ij} P(u_r^i = u^{ij}|K_r)$. By Def 3, each node $u^i@t$ has a single child node $w^i@t$ that itself has no child. In Algorithm 5, parent variables of each $u^i@t$ are instantiated at step 3. Neither any $u^i@t$ nor any $w^i@t$ is instantiated. Hence, after instantiation and belief propagation (step 4), the updated probability at each node $u^i@t$ is

$$P'(u^i@t = u^{ij}) = P(u_r^i = u^{ij}|K_r). \quad (5)$$

Since each node $w^i@t$ is not instantiated and has no child, after belief propagation, we have

$$P'(w^i@t) = P(w^i@t|K_r) \quad (6)$$

$$= \sum_{u^i@t} P(w^i@t, u^i@t|K_r) \quad (7)$$

$$= \sum_{u^i@t} P(w^i@t|u^i@t, K_r) P(u^i@t|K_r) \quad (8)$$

$$= \sum_{u^i@t} P(w^i@t|u^i@t) P(u^i@t|K_r). \quad (9)$$

Eqn (6) holds by belief propagation. Eqn (7) follows from marginalization and Eqn (8) from product. Eqn (9) is due to semantics of PEN. By restricting the value of $w^i@t$ to y, we derive Eqn (10), and from Def 3 on the distribution assigned to $w^i@t$, we derive Eqn (11).

$$P'(w^i@t = y)$$

$$= \sum_j P(w^i@t = y|u^i@t = u^{ij}) P(u^i@t = u^{ij}|K_r) \quad (10)$$

$$= \sum_j u^{ij} P(u_r^i = u^{ij}|K_r). \quad \square \quad (11)$$

Def 3 can be extended to incorporate canonical dependence models such as noisy-or (Pearl 1988). Conditioning factors of an activity can be further decomposed, resulting in longer directed paths in PENs (than those in Fig 2). Algorithm 5 will then have to be modified accordingly. Such modification is straightforward. The validity of Theorem 6, however, remains.

An important advantage of PEN representation and Algorithm 5 is that they allow package evaluation to be performed based on existing belief propagation algorithms intended for Bayesian nets.

Complexity and Experiment

Package evaluation is dominated by the computation performed at each user agent. Using notations in Section for d (max depth of activity grammar tree), b (max branching factor), m (max number of AND nodes in a root-to-leaf path), and c (max branching factor of AND nodes), the number of composite activities to be considered at each greedy search is $O((b^{d-m})\ c^m)$. Hence, each user agent evaluates $O((b^{d-m})\ c^m\ |T|)$ packages.

Let n denote the number of primitive activities. Each PEN has $O(n)$ nodes. Let k denote the maximum number of values of a variable. Let q denote the maximum number of parents of a node. The evaluation computation is in the order $O(n\ k^q)$. Overall, the complexity for each user agent is $O((b^{d-m})\ c^m\ |T|\ n\ k^q)$. When d, m and q are reasonably bounded, the computation is tractable.

A prototype was implemented in meal planning. Its representation contains 49 foods and the meal grammar consists of 27 rules, from which 3 million meal plans per day could be generated. Preliminary experiments show satisfactory results. Details are omitted due to space limit.

Related Work

Package planning provides a class of useful practical applications. We are unaware of previous work that deals with this problem. The package planning problem differs significantly from standard AI *planning*. Standard planning (Weld 1999) is specified by an initial and a goal state, and possible actions. The objective is to find a sequence of actions to transform the initial state to the goal. Package planning has no equivalent goal states and has a significantly different objective.

Similar comparisons can be made between package planning and problems solvable by state-space search algorithms such as A^* and branch-and-bound. These problems have well-defined goal states. There are no natural goal states in package planning.

Decision-theoretic planning (Boutilier, Dean, & Hanks 1999) has been an active research area on planning using MDPs. Package planning shares some features of MDPs (e.g., an activity can be viewed as an action and its desirability can be viewed as a reward). However, the Markov property does not hold in package planning domain because desirability of an activity at a given time may depend strongly on the user's activity history (both recent and more remote). Although a non-Markovian model of a finite order can always be converted to an equivalent Markov model, the approach seems to be more complex than what we have presented: In trying to convert the non-Markovian model into Markovian, if each time interval has k local variables, then to make variables at t_2

independent of those at t_0 given those at t_1, $2k$ variables are needed at t_1 in the worst case. In general, $(m+1)k$ variables are needed for time interval t_m in the worst case. Parallel to this linear increase of number of variables in each time interval, the number of parent variables at t_m also increases linearly with m. As a result, the number of parameters needed to specify the conditional probability distribution at each variable grows exponentially with m. The representation using historic performance grouping (Section) as a reasonable approximation results in no more than $4k$ variables for every time interval in the worst case. For general proposals on solving non-Markovian problems using a temporal logic representation of historic dependence, see Bacchus et al (Bacchus, Boutilier, & Grove 1997).

Package planning differs significantly from *scheduling* (Zweben & Fox 1994). Although scheduling also selects among alternative plans for activities, it aims at optimizing resource allocation relative to some *hard* resource constraints and measures such as tardiness, inventory and makespan. Package planning, instead, aims at optimization subject to a set of desirability and cost preferences of *soft* nature.

More generally, package planning differs from constraint satisfaction problems (CSP). How a composite activity is made out of primitive activities may be alternatively represented by constraints. The corresponding computation for activity composition appears to be more complex than what is needed using the BNF grammar. Valued CSPs (Schiex, Fargier, & Verfaillie 1995) allow preferences to be handled in a CSP framework. However, composite activities in a package rarely follow any *hard* constraints. Hence, representation of package planning as a CSP appears to be unnatural and less effective than what we have been presented.

Our work is influenced by work on graphical models (Pearl 1988; D'Ambrosio 1999) and by situation specific model construction (e.g., (Goldman & Breese 1992; Mahoney & Laskey 1998)). Each PEN is such a situation specific model. PEN evaluation using probabilistic reasoning is inspired by (Cooper 1988). Utility decomposition has been explored by Bacchus and Grove (Bacchus & Grove 1995) in a generic context although they did not address uncertainty on utility functions.

Limitations and Future Work

We identify *package planning* as a class of novel applications. Solution by exhaustive search is intractable. To solve the problem effectively, we assumed *(1) independence of individual preference, (2) future independence of activities, (3) grouping of historic dependence, (4) greedy search, (5) utility decomposition, (6) uncertain user preference, and (7) discrete utility.* Some assumptions, e.g., (3),(4) and (7), introduce approximations. Other assumptions prevent certain packages to be selected. For instance, (1) prevents certain types of preference toward joint activities, as illustrated in Section . Further research is needed to relax this assumption. Assumption (2) also appears to limit selected packages. Whether it really has any negative impact and to what degree need to be carefully assessed. Some specific representational choices were also made in the framework developed, such as the BNF grammar for composite activity and the encoding of historic performance.

Despite these limitations, these assumptions greatly reduced computational complexity. Based on these assumptions and representational choices, we developed the first computational framework where agents working for different principals cooperate to select a package that approximately optimally suits all users. The computation is tractable when problem parameters are reasonably bounded. The framework provides a basis for further generalization, elaboration and improvement.

Our presentation does not deal with acquisition of probability and utility parameters. Work on parameter learning, e.g., (Spiegelhalter & Lauritzen 1990), and preference elicitation, e.g., (Nguyen & Haddawy 1998), are relevant on this regard.

Acknowledgement Financial support from NSERC, Canada is acknowledged. We thank reviewers for their helpful comments.

References

Bacchus, F., and Grove, A. 1995. Graphical models for preference and utility. In *Proc. 11th Conf. on Uncertainty in Artificial Intelligence*, 3–10.

Bacchus, F.; Boutilier, C.; and Grove, A. 1997. Structured solution methods for non-Markovian decision process. In *Proc. 14th National Conf. on Artificial Intelligence*, 112–117.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision theoretic planning: structural assumptions and computational leverage. *J. Artificial Intelligence Research* 1–94.

Cooper, G., and Herskovits, E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9:309–347.

Cooper, G. 1988. A method for using belief networks as influence diagrams. In Shachter, R.; Levitt, T.; Kanal, L.; and Lemmer, J., eds., *Proc. 4th Workshop on Uncertainty in Artificial Intelligence*, 55–63.

D'Ambrosio, B. 1999. Inference in Bayesian networks. *AI Magazine* 20(2):21–36.

Goldman, R., and Breese, J. 1992. Integrating model construction and evaluation. In Dubois, D.; Wellman, M.; D'Ambrosio, B.; and Smets, P., eds., *Proc. 8th Conf. on Uncertainty in Artificial Intelligence*, 104–111. Stanford University: Morgan Kaufmann.

Heckerman, D.; Geiger, D.; and Chickering, D. 1995. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning* 20:197–243.

Keeney, R., and Raiffa, H. 1976. *Decisions with Multiple Objectives*. Cambridge.

Mahoney, S., and Laskey, K. 1998. Constructing situation specific belief networks. In *Proc. 14th Conf. on Uncertainty in Artificial Intelligence*, 370–378.

Nguyen, H., and Haddawy, P. 1998. The decision-theoretic video advisor. In *AAAI Workshop on Recommender Systems*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proc. Inter. Joint Conf. on Artificial Intelligence*, 631–637.

Spiegelhalter, D., and Lauritzen, S. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks* 20:579–605.

Weld, D. 1999. Recent advances in AI planning. *AI Magazine* 20(2):93–123.

Zweben, M., and Fox, M., eds. 1994. *Intelligent Scheduling*. Morgan Kaufmann.