A Computational Framework for Package Planning

Y. Xiang and M. Janzen
Department of Computing and Information Science
University of Guelph, Canada

Corresponding author:
Prof. Yang Xiang
Department of Computing and Information Science
University of Guelph, Canada N1G 2W1
Tel: (519) 824-4120 Ext. 52824
Fax: (519) 837-0323
Email: yxiang@cis.uoguelph.ca

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, heuristic search, decision theoretic graphical models, and dynamic preference aggregation. We show that the computation is tractable when the problem parameters are reasonably bounded.

`Keywords:` Graphical models, uncertain reasoning, decision theory, utility networks, package planning.

1 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 plan for meals 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 constraints, and generates a shopping list.

Cable and satellite TV channels supply many programs daily. A program may be broadcasted 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 pre-arrangement 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.

The general problem of package planning has an intractable search space, as we will elaborate in the paper. We are unaware of existing work that addresses the general problem of package planning computationally, except [9] which studies a special case on meal planning. In this paper, we present a formal framework for solving this problem tractably in average cases. In Section 11, the novelty of this contribution is highlighted by comparing with work from several related AI areas, including standard AI *planning*, problems solving by state-space search such as A^* and branch-and-bound, decision-theoretic planning, scheduling, and constraint satisfaction.

2 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 finite, nonempty set F of users, for instance, $F = \{Lisa, David, ...\}$.

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. Each user is engaged in no more than one activity in each interval. Formally, we denote the planning time period by a finite, nonempty sequence of intervals $T = \{t_0, t_1, ...\}$. In meal planning, for instance, we have

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

Possible activities of users are represented by elements of a finite 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. The set A contains a special element called a *null* activity. The null activity is intended to capture any activity outside those in A.

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 $\{Lisa\}$, a subset of the family $\{Lisa, David\}$, or the whole family F. 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. Similarly, *(Great Wall, Monday,* $\{David\}$*)* has the obvious meaning in tour planning. Note that it is not necessary for K to contain a normal planned activity for each $t \in T$ and for each user $r \in F$. A user r may prefer the *null* activity in a given interval t and hence the user will have $(null, t, r)$ for t, called a null planned activity. The collection of all possible packages for a family is denoted by $\mathcal{K} = \{K_0, K_1, ...\}$ and is 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.*

The goal of the `package planning problem` *in the given domain is to find a package K^* in $\mathcal{K}$ such that*

$$\sum_{r \in F} k_r \mu_r(K^*) = \max_{K \in \mathcal{K}} [\sum_{r \in F} k_r \mu_r(K)], \tag{1}$$

where $k_r > 0$ for each $r \in F$, $\sum_{r \in F} k_r = 1$, and maximization is over all K of length $|T|$.

That is, the goal of package planning is to find a package that best suits all users in a family over the planning period.

In the problem definition, we have used additive utility combination to aggregate individual user's utility about a package. This is based on the classical work [10] on decision with multiple objectives. Additive utility combination is sufficiently expressive in representation without being intractable in acquisition. It provides a reasonable tradeoff between expressiveness in representation and efficiency in utility acquisition.

3 Individual Preference

A package consists of a set of planned activities. Some activity is planned for only one user and some for several users. Given a user r and a planned activity (a, t, R), we define the user's perspective of the activity as the *projection* of the activity onto r. For example, the projection of

(turkey meal, Sunday evening, F)

onto user Lisa is

(turkey meal, Sunday evening, {Lisa}).

Each planned activity can be projected to each user r with the convention that (a, t, R) projected to $\{r\}$, where $r \notin R$, yields a *null* planned activity $(null, t, r)$.

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 $\mathcal{K}_r$.

With the notion of projected package, we consider the acquisition of each user's utility function $\mu_r(K)$. In general, we have

$$\mu_r(K) = f(K_r, \mu_{r_0}(K), \mu_{r_1}(K), ...),$$

where $f()$ is some function and $r \neq r_i$. Note that the condition $r \neq r_i$ ensures that arguments of $f()$ are relative to distinct users. That is, the utility of a package K to user r depends not only on how much r likes the projected package K_r, but also how much other users like the package. Such generality, however, introduces a circular preference dependence, because each user's preference depends on each other's preference. The circular preference makes acquisition of $\mu_r(K)$ very difficult.

To make preference acquisition practical, we assume that a user's preference towards a given package is *self-sufficient*. That is, we assume

$$\mu_r(K) = f(K_r).$$

It allows each user to express 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 self-sufficiency of individual preference, for each

user r, his/her 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

$$\sum_{r \in F} k_r \mu_r(K_r^*) = \max_{K \in \mathcal{K}} [\sum_{r \in F} k_r \mu_r(K_r)]. \quad (2)$$

We emphasize that self-sufficiency of individual preference only requires $\mu_r(K)$ to be self-defined on K_r. It does not prevent interaction of user preferences. For example, "I like whatever Lisa likes" can be expressed as follows:

"Let K be any package where, for each $t \in T$, if the projection of the activity (a, t, R) to Lisa is $(a', t, \{Lisa\})$, the projection to me is $(a', t, \{I\})$. Then, for every such package, my preference satisfies $\mu_I(K) = \mu_{Lisa}(K)$."

Similarly, "I don't like anything Lisa likes and I like anything she doesn't" can be expressed by the following:

"Let K be any package where, for each $t \in T$, if the projection of the activity (a, t, R) to Lisa is $(a', t, \{Lisa\})$, the projection to me is $(a', t, \{I\})$. Then, for every such package, my preference satisfies $\mu_I(K) = 1 - \mu_{Lisa}(K)$."

In summary, self-sufficiency of individual preference allows more focused utility acquisition and package evaluation without significantly limiting expressiveness.

4 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 in $\mathcal{K}$. Details on the generation are presented in Sections 5 and 6. For each package K, g sends K_r to each g_r for evaluation. When g receives the evaluation from each g_r, it integrates them into a final evaluation of K according to Eqn (2). The best K is selected after enough alternatives have been evaluated.

Each user agent g_r is responsible for acquiring the utility function μ_r from its principal and maintaining it. In this paper, we do not deal with in details the issue of preference elicitation. Useful techniques can be found in [10, 12]. User agent g_r is also in charge of evaluating each K_r received from g. The result $\mu_r(K_r)$ is then sent back to g. Details on evaluation are presented in Sections 7 and 8.

The 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. A given user r may be served by different user agents specialized in particular tasks, for instance, a meal planning user agent, a TV planning user agent, and a tour planning user agent. On the other hand, a tour manager agent g works for a travel agency and provides service to users from many different households.

Note that user agents working for different principals may interact directly with each other in addition to indirect interaction through the manager agent. For instance, when David claims that "I like anything that Lisa likes", his meal planning user agent will interact with Lisa's counterpart to acquire the utility function.

The assumption that agents work for different principals implies a distributed (versus centralized) implementation of package planning.

5 Space Reduction with Grammar

To select the best package, each projected package needs to be evaluated by the 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 6 and 7). We consider the complexity of the activity space here:

An activity may be *primitive* or *composite*. A primitive activity is one that cannot be further decomposed. A composite activity is composed of multiple primitive and composite activities. 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

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

as one side and steamed broccoli as another, Caesar salad, wine as a drink, and ice cream as desert. Backed potato is a primitive activity and the meal itself is a composite activity.

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 space is clearly intractable to process. Similar situations arise in TV planning and tour planning as well. In general, we need to consider the complexity due to 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 Figure 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.

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 only AND nodes, Y defines a single composite activity of all leaves.

When Y contains both AND and OR nodes, there is no alternative at an AND node and a composite activity is a subset 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 a pair of bounds, depending on how close the AND nodes are from the root (at the top). The lower bound is $O(b^{d-m})$ when all AND nodes are at the bottom of Y, which gives the complexity of a reduced OR tree. The upper bound is $O(b^{d-m} * c^m)$ when all AND nodes are at the top of Y, which gives the complexity of c^m OR trees. When d and m are small, $O(b^{d-m} * c^m)$ is tractable. For the example of Figure 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.

6 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.

This assumption is clearly motivated by the computational efficiency. Its implications include the following: First, it does not eliminate the dependence between historical and future activities. Although the desirability of an activity does not depend on future activities when it is being considered, once the activity is determined, it does affect the desirability of future activities. Therefore, the future independence assumption does not by itself reduce the complexity of the package search space. Second, it allows the user utility function $\mu_r(K_r)$ to be defined over a partial package K_r (over a subset of intervals of T). Without this assumption, utility over partial package is not well-defined.

Third, the availability of utility over partial packages enables greedy search as an approximation method, which directly contributes to the reduction of computational complexity. Greedy search is commonly used in learning probabilistic graphical models, e.g., [5, 8]. To apply greedy search to package evaluation, we search for the best activity in each time interval, considering only dependence 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 [9].

Fourth, in Section 7, we present a graphical model representation of the dependence between an activity under consideration and activities in the history. It would not be possible without this assumption.

7 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. A meal that is delicious and expensive has both high desirability and high cost. We aggregate user utility function μ_r from a cost related utility denoted by c_r and a desirability related utility denoted by u_r through additive utility combination [10]:

$$\mu_r = \alpha_r \; c_r + (1 - \alpha_r) \; u_r,$$

where a constant $(0 < \alpha_r < 1)$ encodes the relative importance of cost and desirability to the user. The package planning problem can then be solved by finding a package K^* such that

$$\sum_{r \in F} k_r \mu_r(K_r^*) = \max_{K \in \mathcal{K}'} \; [\sum_{r \in F} k_r [\alpha_r c_r(K_r) + (1 - \alpha_r) u_r(K_r)]]. \tag{3}$$

Note that the package search is over subspace $\mathcal{K}'$ due to greedy search.

Assessment of cost utility c_r is relatively straightforward and the cost of a package can be aggregated from the costs of primitive activities (e.g., based on amount of money or amount of time required).

On the other hand, assessment of u_r is more difficult because there is no simple metrics for desirability. Again, we rely on decomposition and additive utility combination. 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 $u_r(a', t)$. Furthermore, $u_r(a', t)$ can be aggregated from the utility of each primitive activity a contained in the composite activity a'. 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. Aggregation weight is assigned to each activity according to the time of the activity. For instance, a dinner may be treated as more important than a breakfast. See Section 8 for details on this weight. We emphasize that the additive aggregation does not prevent counting utility dependence among activities, as will be clear from the presentation below and in Section 8.

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 as context factors, the time of the day and the season.

Without confusion and for simplicity, we refer to a user's desirability related utility as just desirability. 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 the number of parameters in $u_r^i(a^i|\pi^i)$ 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 once in the last two days, then the variable $just - had - a^i$ has the value *some* (versus *none* and *a lot*). 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.

The computational agent g_r maintains $u_r^i(a^i|\pi^i)$ but may not know its parameters with certainty because they are subjective quantities that belong to its principal. 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 8), 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 a utility function $u_r^i(a^i|x_0, y_0)$, where (x_0, y_0) is a configuration of π^i, may then be expressed as

$$\begin{aligned}
&P(u_r^i(a^i|x_0, y_0) = 0) = 0,\\
&P(u_r^i(a^i|x_0, y_0) = 0.25) = 0.05,\\
&P(u_r^i(a^i|x_0, y_0) = 0.5) = 0.2,\\
&P(u_r^i(a^i|x_0, y_0) = 0.75) = 0.6,\\
&P(u_r^i(a^i|x_0, y_0) = 1.0) = 0.15.
\end{aligned}$$

We simplify the above notation $P(u_r^i(a^i|\pi^i))$ as $P(u_r^i|\pi^i)$, i.e.,

$$\begin{aligned}
&P(u_r^i = u^{i_0}|x_0, y_0) = 0,\\
&P(u_r^i = u^{i_1}|x_0, y_0) = 0.05,\\
&P(u_r^i = u^{i_2}|x_0, y_0) = 0.2,\\
&P(u_r^i = u^{i_3}|x_0, y_0) = 0.6,\\
&P(u_r^i = u^{i_4}|x_0, y_0) = 0.15,
\end{aligned}$$

where u^{i_1} denotes the utility value 0.25.

Evaluation of $c_r(K_r)$ is performed by first evaluating the cost of K_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 5 for cost evaluation in a straightforward way. The cost evaluation is then a simple summation.

8 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*

$$u_r(K_r) = \sum_t \beta_t \sum_{a^i@t} \gamma_i \sum_j [u^{i_j} P(u_r^i = u^{i_j}|K_r)], \tag{4}$$

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

The summation $\sum_{a^i@t}$ is over each primitive activity a^i at time t. The weight γ_i is best set to a constant for simplicity. The summation $\sum_j$ is over each discrete utility value for activity a^i at time t. The weight β_t expresses the relative importance of the activity at time t, as discussed in the last section. The summation $\sum_t$ combines activities at all time intervals of K_r.

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 $(a, t, \{r\})$, decompose a to primitive activities. For each primitive activity a^i in a, there are three variables in V, activity $a^i@t \in \{t, f\}$, desirability $u^i@t \in \{u^{i_0}, u^{i_1}, ...\}$ 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.*

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 *[13] 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*

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

Consider a planned lunch for Lisa in a projected package. The corresponding composite activity is

$$(hotdog\ meal, Friday\ noon, \{Lisa\}).$$

Suppose that the hotdog meal is composed of *hotdog*, *hot chocolate* drink and *chocolate ice cream* desert. Figure 2 shows a partial PEN graph structure for the package. For each primitive activity, hotdog, hot chocolate and chocolate ice cream, there is an activity node (the a-node), a desirability node (the u-node) and the expected utility node (the w-node). The a-node for hotdog is not shown as no other node is linked to it in the partial graph. The time 'Friday noon' has been labeled as 'fnoon'.

The desirability of hot chocolate depends on several factors shown as the parents of the u-node. It depends on whether Lisa has had hot chocolate within the last two days. It depends on whether there is a chocolate ice cream as presence of such a desert will make hot chocolate drink less desirable (Lisa does not like to have chocolate in both drink and ice cream). Note that even if Lisa likes hot chocolate drink and chocolate ice cream individually, if she does not like to have chocolate in both drink and ice cream, the desirability of each of them will be reduced by the dependence from a_HotChocolate@fnoon to u_ChocolateIceCream@fnoon and that from a_ChocolateIceCream@fnoon to u_HotChocolate@fnoon. The consequence is that the package containing such a meal will be given less favorable evaluation by Lisa's agent. The desirability of hot chocolate also depends on the meal type since although the time of the meal is noon, Lisa could have returned from an oversea trip. Due to jetlag, she is having breakfast at noon for Friday, which will make a hotdog meal undesirable. Finally, the graph shows that it depends on the season as a hot drink is more preferable in the winter than in the summer.

Syntactically, a PEN is equivalent to a Bayesian net [13] 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 [6]:

[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.

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 $E = \sum_t \beta_t \sum_{w^i@t} \gamma_i \; P'(w^i@t = y)$;

The algorithm starts by instantiating parent variables of $u^i@t$, which corresponds to entering observations in standard probabilistic reasoning with graphical models [6]. 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

$$E = \sum_t \beta_t \sum_{w^i@t} \gamma_i \; P'(w^i@t = y)$$

$$= \sum_t \beta_t \sum_{a^i@t} \gamma_i \sum_j [u^{i_j} P(u^i_r = u^{i_j} | K_r)] = u_r(K_r).$$

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 $w^i@t$:

$$P'(w^i@t = y) = \sum_j u^{i_j} P(u^i_r = u^{i_j} | K_r).$$

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

$$P'(u^i@t = u^{i_j}) = P(u^i_r = u^{i_j} | K_r). \tag{5}$$

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

$$\begin{aligned} P'(w^i@t) &= P(w^i@t | K_r) && (6)\\ &= \sum_{u^i@t} P(w^i@t, u^i@t | K_r) && (7)\\ &= \sum_{u^i@t} P(w^i@t | u^i@t, K_r) P(u^i@t | K_r) && (8)\\ &= \sum_{u^i@t} P(w^i@t | u^i@t) P(u^i@t | K_r). && (9) \end{aligned}$$

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).

$$\begin{aligned} & P'(w^i@t = y)\\ = & \sum_j P(w^i@t = y | u^i@t = u^{i_j}) * P(u^i@t = u^{i_j} | K_r) && (10)\\ = & \sum_j u^{i_j} P(u^i_r = u^{i_j} | K_r). && (11) \end{aligned}$$

□

Def 3 forms the basic mechanism to represent and evaluate $u_r(K_r)$. It can be extended to incorporate other general features of uncertain reasoning using graphical models. For instance, at the time of planning, whether Lisa will have an oversea trip later in the week may be pending. As a result, the meal type at Friday noon may be a normal lunch or a late breakfast. The node mealType@fnoon in Figure 2 will then be substituted by a network segment that encodes the additional variables and dependence, which can be used to infer the meal type at Friday noon. To deal with such more general cases of PEN, Algorithm 5 has 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.

9 Minority Users

In Eqn (2), k_r is used to determine how significant the preference of user r will be relative to those of other users. Although a constant k_r seems to be a simple choice, it may lead to unhappy minority users. This may occur due to the existence of a majority of users who share common preference patterns which may be in conflict with those of the minority users. For instance, activities favored by the majority may be disliked by the minority. As a result, the package filled with activities that suit the majority will always be selected, leaving the minority unhappy.

We propose a computational mechanism to first detect the existence of unhappy users and then adjust the package selection process such that no user will be left unhappy forever.

Definition 7 *A selected partial package K^* is an* unhappy package *for user r, if*

$$1 - \frac{\mu_r(K_r^*)}{\max_{K'} \mu_r(K_r')} > \omega,$$

where maximization is over all partial packages K' searched that lead to the selection of K^*, and $\omega > 0$ is a threashold.

Intuitively,

$$\frac{\mu_r(K_r^*)}{\max_{K'} \mu_r(K_r')}$$

represents the degree of happiness of r with the selected package K^* and its difference from 1 represents the degree of unhappiness. If r is sufficiently unhappy (difference $> \omega$), the choice is deemed an unhappy package to r.

An occasional unhappy package to a user is unavoidable in a family. It is the continuous unhappiness of some users that should be avoided. We denote the percentage of unhappy partial packages of user r in the last x time intervals by θ_r, where the value of x is a predetermined parameter. Based on θ_r, the weight k_r is adjusted every x time intervals by the manager agent g according to Algorithm 8. In the algorithm, $f()$ is a non-negative and monotonically increasing function with $f(0) = 0$.

Algorithm 8 *At the end of package search for each x time intervals, agent g does the following.*

1 for each user agent g_r, do
2 receive θ_r from g_r;
3 adjust weight k_r to $k_r' = k_r(1 + f(\theta_r))$;
4 for each user agent g_r, do
5 $k_r'' = \frac{k_r'}{\sum_{s \in F} k_s'}$;
6 save k_r'' for package evaluation for the next x intervals;

The method is efficient as θ_r involves simple local computation at g_r and adjustment involves simple local computation at g. It has the desirable property that as long as a user r is sufficiently unhappy in the last x intervals, its weight k_r will be increased for the next x intervals, as shown in the following Proposition.

Proposition 9 *Let r be the only user where $\theta_r > 0$ for the last x intervals. Then $k''_r > k_r$, where k''_r is defined by Algorithm 8.*

Proof:

Since r is the only unhappy user, for each other user s, we have $\theta_s = 0$ and $k'_s = k_s(1 + f(\theta_s)) = k_s$. Therefore,

$$k''_r = \frac{k_r(1 + f(\theta_r))}{(\sum_{s \in F} k_s) + k_r f(\theta_r)} = \frac{k_r(1 + f(\theta_r))}{1 + k_r f(\theta_r)}.$$

Because $0 < k_r < 1$ and $f(\theta_r) > 0$, we have $1 + k_r f(\theta_r) < 1 + f(\theta_r)$. Hence, $k''_r > k_r$.

□

Furthermore, Algorithm 8 allows simultaneous weight adjustment for multiple unhappy users, as shown by Theorem 10.

Theorem 10 *Let R be the proper subset of all users such that for each $r \in R$, $\theta_r > 0$ for the last x intervals. If*

$$f(\theta_r) > \sum_{s \in R} k_s f(\theta_s),$$

then $k''_r > k_r$ for each $r \in R$, where k''_r is defined by Algorithm 8.

Proof:

For each user $r \in R$, we have

$$k''_r = \frac{k_r(1 + f(\theta_r))}{(\sum_{s \in F} k_s) + \sum_{s \in R} k_s f(\theta_s)} = \frac{k_r(1 + f(\theta_r))}{1 + \sum_{s \in R} k_s f(\theta_s)}.$$

From the assumption, the result follows. □

Note that, the condition

$$f(\theta_r) > \sum_{s \in R} k_s f(\theta_s)$$

in Theorem 10 can be intuitively stated as that R is a minority group. To see the equivalence, suppose $f(\theta_s)$ is identical to all users in R. Then

$$\sum_{s \in R} k_s f(\theta_s) = f(\theta_r) \sum_{s \in R} k_s < f(\theta_r).$$

Algorithm 8 ensures that r's preference will be given more dominant consideration. If the resultant partial packages still do not make r happy, an even larger weight will be assigned to r for the next x intervals. Eventually, r's evaluation will be sufficiently dominant and the chosen family package will contain some of its favored activities.

10 Complexity Analysis

Package evaluation is dominated by the computation performed at each user agent. Using notations in Section 6 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. Experiments show satisfactory results which are reported in [9].

11 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 [15] 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 [3] 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 q local variables, then to make variables at t_2 independent of those at t_0 given those at t_1, $2q$ variables are needed at t_1 in the worst case. In general, $(h+1)q$ variables are needed for time interval t_h in the worst case. Parallel to this linear increase of number of variables in each time interval, the number of parent variables at t_h also increases linearly with h. As a result, the number of parameters needed to specify the conditional probability distribution at each variable grows exponentially with h. The representation using historic performance grouping (Section 7) as a reasonable approximation results in no more than $4q$ 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 [1].

Package planning differs significantly from *scheduling* [16]. 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 representation and computation for activity composition appears to be more complex than what is needed using the BNF grammar. Valued CSPs [14] 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 [13, 6] and by situation specific model construction (e.g., [7, 11]). Each PEN is such a situation specific model. PEN evaluation using probabilistic reasoning is inspired by [4]. Utility decomposition has been explored by Bacchus and Grove [2] in a generic context although they did not address uncertainty on utility functions.

12 Conclusion

We identify *package planning* as a class of novel applications for knowledge based systems. Solution of this problem by exhaustive search is intractable. To solve the problem effectively, we assumed *(1) self-sufficiency 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.* These assumptions greatly reduce computational complexity, but also introduce approximations. As we argued in the paper, they do not appear to be severely limiting. 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.

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. For instance, we have defined, in

Eqn (1), the goal of package planning as to find a package that maximizes the weighted sum of user utilities. Alternatively, the goal could be defined to find a package where the minimal user utility is maximized.

Our framework is efficient under the above stated condition due significantly to a series of decompositions of the computation. It is important to note that our decompositions retain the interaction between individual user preferences. First of all, a user can define its utility function from other user's utility functions based on the similarity and dissimilarity (Section 3). Second, when a collective activity is being planned, each user can exert his/her influence based on its personal preference. This is accomplished from the individually constructed user preference structure (Section 8) and from the user evaluation aggregation (Section 4). The existence of a planed activity in the current partial package triggers different reactions at each user's PEN. If enough users do not like it, it will be rejected. If it is mostly favored by enough users, it will be selected. Finally, overall happiness of the family is maintained by dynamically modifying how much each user's voice counts (Section 9).

Acknowledgement

Financial support from NSERC of Canada, in terms of a Discovery Grant to the first author and in terms of a Postgraduate Scholarship to the second author, is acknowledged. We thank anonymous reviewers for their encouragement and helpful suggestions.

References

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

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

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

[4] G.F. Cooper. A method for using belief networks as influence diagrams. In R.D. Shachter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer, editors, *Proc. 4th Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.

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

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

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

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

[9] M. Janzen and Y. Xiang. Probabilistic reasoning for meal planning in intelligent fridges. In Y. Xiang and B. Chaib-draa, editors, *Advances in Artificial Intelligence, LNAI 2671*, pages 575–582. Springer, 2003.

[10] R.L. Keeney and H. Raiffa. *Decisions with Multiple Objectives*. Cambridge, 1976.

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

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

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

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

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

[16] M. Zweben and M.S. Fox, editors. *Intelligent Scheduling*. Morgan Kaufmann, 1994.

Figure 1 caption: An example meal grammar.

Figure 2 caption: Partial PEN for a projected package.

$< start > \longrightarrow < breakfast > | < dinner >$
$< breakfast > \longrightarrow$ 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.

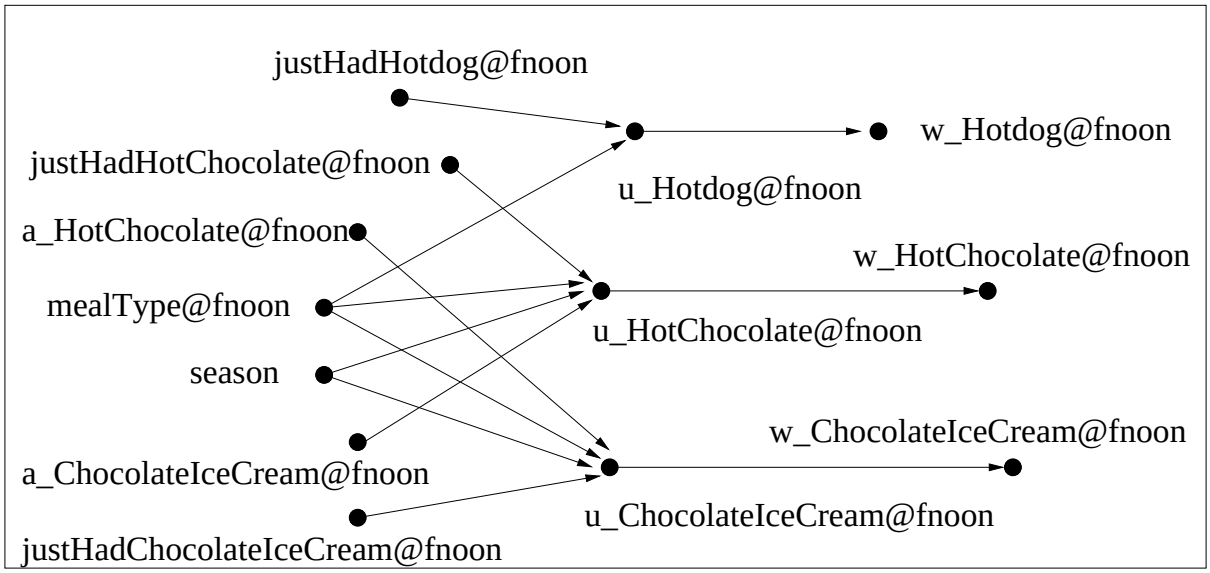

Figure 2: Partial PEN for a projected package.