Artificial Intelligence 190 (2012) 1–31

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Generating diverse plans to handle unknown and partially known user preferences [☆]

Tuan Anh Nguyen [a,∗,1], Minh Do [b,2], Alfonso Emilio Gerevini [c], Ivan Serina [d], Biplav Srivastava [e], Subbarao Kambhampati [a]

[a] School of Computing, Informatics, and Decision System Engineering, Arizona State University, Brickyard Suite 501, 699 South Mill Avenue, Tempe, AZ 85281, USA
[b] NASA Ames Research Center, Mail Stop 269-3, Moffett Field, CA 94035-0001, USA
[c] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Via Branze 38, I-25123 Brescia, Italy
[d] Free University of Bozen–Bolzano, Viale Ratisbona, 16, I-39042 Bressanone, Italy
[e] IBM India Research Laboratory, New Delhi and Bangalore, India

A R T I C L E I N F O

Article history:
Received 12 January 2011
Received in revised form 16 May 2012
Accepted 16 May 2012
Available online 18 June 2012

Keywords:
Planning
Partial preferences
Diverse plans
Heuristics
Search

A B S T R A C T

Current work in planning with preferences assumes that user preferences are completely specified, and aims to search for a single solution plan to satisfy these. In many real world planning scenarios, however, the user may provide no knowledge or at best partial knowledge of her preferences with respect to a desired plan. In such situations, rather than presenting a single plan as the solution, the planner must instead provide a *set of plans* containing one or more plans that are similar to the one that the user really prefers. In this paper, we first propose the usage of different measures to capture the quality of such plan sets. These are domain-independent distance measures based on plan elements (such as actions, states, or causal links) if no knowledge of the user preferences is given, or the *Integrated Convex Preference* (ICP) measure in case incomplete knowledge of such preferences is provided. We then investigate various heuristic approaches to generate sets of plans in accordance with these measures, and present empirical results that demonstrate the promise of our methods.

1. Introduction

In many real world planning scenarios, user preferences on plans are either unknown or at best partially specified (cf. [33]). In such cases, the planner's task changes from finding a *single* optimal plan to finding a *set* of representative solutions or options. The user must then be presented with this set in the hope that she will find at least one of the constituent plans desirable and in accordance with her preferences. Most work in automated planning ignores this reality, and assumes instead that user preferences (when expressed) will be provided in terms of a completely specified objective function.

☆ This work is an extension of the work presented in Srivastava et al. (2007) [51] and Nguyen et al. (2009) [42].
∗ Corresponding author.
E-mail addresses: natuan@asu.edu (T.A. Nguyen), minh.b.do@nasa.gov (M. Do), gerevini@ing.unibs.it (A.E. Gerevini), ivan.serina@unibz.it (I. Serina), sbiplav@in.ibm.com (B. Srivastava), rao@asu.edu (S. Kambhampati).
1 Authors listed in alphabetical order, with the exception of the first and the last.
2 Company affiliation: Stinger Ghaffarian Technologies (SGT) Inc.

0004-3702/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.artint.2012.05.005

In this article, we study the problem of generating a *set of plans* using partial knowledge of the user preferences. This set is generated in the hope that the user will find at least one desirable according to her preferences. Specifically, we consider two qualitatively distinct scenarios:

- The planner is aware that the user has some preferences on the solution plan, but it is not provided with any knowledge on those preferences.
- The planner is provided with incomplete knowledge of the user preferences in the form of plan *attributes* (such as the duration or cost of a flight, or the importance of delivering all priority packages on time in a logistics problem). Each of these plan attributes has a different and unknown degree of importance, represented by *weights* or *trade-off* values. In general, users find it hard to indicate the exact value of a trade-off, but are more likely to indicate that one attribute is more (or less) important than another. For instance, a business executive may consider the duration of a flight as a more important factor than its cost. Incompletely specified preferences such as these can be modeled with a probability distribution on weight values,[3] and can therefore be assumed as input to the planner (together with the attributes themselves).

In both of the cases above, our focus is on returning a set of plans. In principle, a larger plan set implies that the user has a better chance of finding the plan that she desires; however, there are two problems — one computational, and the other comprehensional. Plan synthesis, even for a single plan, is costly in terms of computational resources used; for a large set of plans, this cost only increases. The comprehensional problem, moreover, is that it is unclear if the user will be able to completely inspect a set of plans in order to find the plan she prefers. What is clearly needed, therefore, is the ability to generate a set of plans with the highest chance of including the user's preferred plan among all sets of *bounded* (small) number of plans. An immediate challenge in this direction is formalizing what it means for a *meaningful* set of plans — in other words, we want to define a *quality measure* for plan sets given an incomplete preference specification.

We propose different quality measures for the two scenarios listed above. In the extreme case where the user is unable to provide any knowledge of her preferences, we define a spectrum of distance measures between two plans based on their syntactic features in order to define the *diversity* measure of plan sets. These measures can be used regardless of the user preferences, and by maximizing the diversity of a plan set we increase the chance that the set is uniformly distributed in the unknown preference space. This makes it more likely that the set contains a plan that is close to the one desired by the user.

The quality measure can be refined further when some knowledge of the user preferences is provided. We assume that it is specified as a convex combination of the plan attributes mentioned above, and incomplete in the sense that a distribution of trade-off weights, not their exact values, is available. The complete set of best plans (plans with the best value function) can then be pictured as the lower convex hull of the Pareto set on the attribute space. To measure the quality of any (bounded) set of plans on the complete optimal set, we adapt the idea of *Integrated Preference Function* (IPF) [14], and in particular its special case, the *Integrated Convex Preference* (ICP). This measure was developed in the Operations Research (OR) community in the context of multi-criteria scheduling, and is able to associate a robust measure of representativeness with any set of solution schedules [21].

Armed with these quality measures, we can formulate the problem of planning with partial user preferences as finding a bounded set of the plans that has the best quality value. Our next contribution therefore is to investigate effective approaches for using quality measures to search for a high quality plan set efficiently. For the first scenario — when the preference specification is not provided — two representative planning approaches are considered. The first, GP-CSP [19], typifies the issues involved in generating diverse plans in bounded horizon compilation approaches; while the second, LPG [27], typifies the issues involved in modifying the heuristic search planners. Our investigations with GP-CSP allow us to compare the relative difficulties of enforcing diversity with each of the three different distance measures defined in the forthcoming sections. With LPG, we find that the proposed quality measure makes it more effective in generating plan sets over large problem instances. For the second case — when part of the user preferences is provided — we also present a spectrum of approaches that can solve this problem efficiently. We implement these approaches on top of Metric-LPG [28]. Our empirical evaluation compares these approaches both among themselves as well as against the methods for generating diverse plans ignoring the partial preference information, and the results demonstrate the promise of our proposed solutions.

The rest of this paper is organized as follows. We start with a comprehensive discussion of related work in Section 2. Section 3 reviews fundamental concepts in preferences and introduces formal notations. In Section 4, we formalize the proposed quality measures for plan sets for the two cases of unknown and partially known user preferences. Sections 5 and 6 present and experimentally evaluate heuristic approaches for generating plan sets with respect to the introduced quality measures. Finally, Section 7 characterizes the limitations of our approaches, and Section 8 presents conclusions and outlines future directions.

[3] If there is no prior information about this probability distribution, one option is to initialize it with the uniform distribution and gradually improve it based on interaction with the user.

2. Related work

There are currently very few research efforts in the planning literature that explicitly consider incompletely specified user preferences during planning. The most common approach for handling multiple objectives is to assume that a specific way of combining the objectives is available [44,20], and then search for an optimal plan with respect to this function. In a sense, such work can be considered as assuming a complete specification of user preferences. Other relevant work includes [13], in which the authors devise a variant of the LAO* algorithm to search for a conditional plan with multiple execution options for each observation branch, such that each of these options is non-dominated with respect to objectives like probability and cost to reach the goal.

Our work can be seen as complementing the current research in planning with preferences. Under the umbrella of planning with preferences, most current work in planning focuses on synthesizing either a single solution plan under the assumption that the user has no preferences, or a single best solution assuming that a complete knowledge of the preferences is given to the planner. We, on the other hand, address the problem of synthesizing a set of plans when the knowledge of user preferences is either completely unknown,[4] or partially specified.

In the context of decision-theoretic planning, some work has considered Markov decision processes with *imprecise* reward functions, which are used to represent user preferences on the visited states during execution. These methods however assume that the true reward function is revealed only during the execution of policies, whereas in our setting the incomplete knowledge about user preferences is resolved after the synthesis of plans but before plan execution (with some required effort from the user). Many different notions of optimality for policies have been defined with respect to the incomplete reward function, and the aim is to search for an optimal policy. The *minimax regret* criterion [45,46,55] has been defined for the quality of policies when the *true* reward function is deterministic but unknown in a given set of functions. This criterion seeks an optimal policy that minimizes the loss (in terms of the expected discounted reward) assuming the presence of an adversary who selects a reward function, among all possible ones, to maximize the loss should a policy be chosen. Another criterion, called *maximin*, maximizes the worst-case expected reward also assuming an adversary acting optimally against the agent [38].

Incomplete knowledge of user preferences can also be resolved with some effort from the user *during* plan generation. This idea unfortunately has not been considered in previous work on automated planning with preferences; there is however some work in two related areas, decision theory and preference elicitation. In [17], the user is provided with a sequence of queries, one at a time, until an optimal strategy with respect to the refined preference model meets a stopping criterion, which is then output to the user. That work ignores the user's difficulty in answering questions that are posted, and instead emphasizes the construction of those which will give the best value of information at every step. This issue is overcome by Boutilier [6] which takes into account the cost of answering future elicitation questions in order to reduce the user's effort. Boutilier et al. [9] consider the preference elicitation problem in which the incompleteness in user preferences is specified on both the set of features and the utility function. In systems implementing the example-critiquing interaction mechanism (e.g., [54,37]), a user critiques examples or options presented by the system, and this information is then used to revise the preference model. The process continues until the user can pick a final choice from the k examples presented. There is one important difference between these methods and ours: the "outcomes" or "configurations" in these scenarios are considered given upfront (or can be obtained with low cost), whereas a feasible solution in many planning domains is computationally expensive to synthesize. As a result, an interactive method in which *a sequence of plans or sets of plans* needs to be generated for critiquing may not be suitable for our applications. Our approach, which presents a set of plans to the user to select, requires less effort from the user and at the same time avoids presenting a single optimal plan according to pessimistic or optimistic assumptions, such as those used in the minimax regret and maximin criteria.

The problem of reasoning with partially specified preferences has also long been studied in multi-attribute utility theory, though this work is also different from ours when ignoring the computation cost of "alternatives". Given prior preference statements on how the user compares two alternatives, Hazen [31] considers additive and multiplicative utility functions with unknown scaling coefficients, which represents the user partial preferences, and proposes algorithms for the consistency problem (i.e., if there exists a complete utility function consistent with the prior preferences), the dominance problem (i.e., whether the prior information implies that one alternative is preferred to another), and the potential optimality problem (i.e., if there exists a complete utility function consistent with the prior preferences under which a particular alternative is preference optimal). Ha and Haddawy [30] addressed the last two problems for multi-linear utility functions with unknown coefficients. These efforts are similar to ours in how the user preferences are partially represented. However, similar to the example-critiquing work mentioned above, they assume that the user is able to provide pairwise comparison between alternatives, which is then used to further constrain the set of complete utility functions representing user preferences.

Our approach to generating diverse plan sets to cope with planning scenarios without knowledge of user preferences is in the same spirit as [52] and [40,41], though for different purposes. Myers, in particular, presents an approach to generate diverse plans in the context of an HTN planner by requiring the metatheory of the domain to be available and by using bias on the metatheoretic elements to control search [41]. The metatheory of the domain is defined in terms of pre-defined attributes and their possible values covering roles, features and measures. Our work differs from this in two respects. First,

[4] Note that not knowing anything about the user's preferences is different from assuming that the user has no preferences.

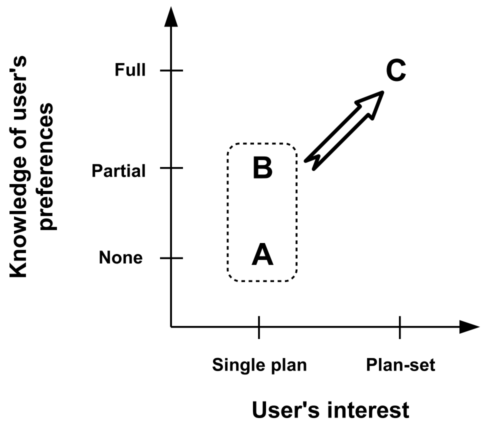


Fig. 1. The planning problem with unknown (**A**) and partially known (**B**) user preferences can be reformulated as the problem of synthesizing plan-sets with complete preferences over plan-sets (**C**).

we focus on domain-independent distance measures. Second, we consider the computation of diverse plans in the context of domain-independent planners.

The problem of finding multiple but *similar* plans has been considered in the context of replanning [22]. Our work focuses on the problem of finding *diverse* plans by a variety of measures when the user preferences exist but are either completely unknown or partially specified.

Outside the planning literature, our closest connection is first to the work by Gelain et al. [25], who consider *soft* constraint satisfaction problems (CSPs) with incomplete preferences. These are problems where quantitative values of some constraints that represent their preferences are unspecified. Given such incomplete preferences, the authors are interested in finding a single solution that is "necessarily" optimal (possibly with some effort from the user), i.e. an assignment of variables that is optimal in all possible ways that the currently unspecified preferences can be revealed. In a sense, this notion of optimality is very similar to the maximin criterion when seeking a solution that is optimal even with the "worst" selection of the unspecified preferences. Hebrard et al. [32] use a model closer to ours that focuses on the problem of finding similar/dissimilar solutions for CSPs, assuming that a domain-specific distance measure between two solutions is already defined. It is instructive to note that unlike CSPs with finite variable domains, where the number of potential solutions is finite (albeit exponential), the number of distinct plans for a given problem can be infinite. Thus, effective approaches for generating a good quality set of plans are even more critical.

The challenges in finding a set of interrelated plans also bear some tangential similarities to the work in other research areas and applications. In information retrieval, Zhang et al. [56] describe how to return relevant as well as novel (non-redundant) documents from a stream of documents; their approach is to first find relevant docs and then find non-redundant ones. In adaptive web services composition, the causal dependencies among some web services might change at execution time, and as a result the web service engine wants to have a set of diverse plans/compositions such that if there is a failure while executing one composition, an alternative may be used which is less likely to be failing simultaneously [15]. However, if a user is helping in selecting the compositions, the planner could be first asked for a set of plans that may take into account the user's trust in some particular sources and when she selects one of them, it is next asked to find plans that are similar to the selected one. Another example of the use of diverse plans can be found in [39] in which test cases for graphical user interfaces (GUIs) are generated as a set of distinct plans, each corresponding to a sequence of actions that a user could perform, given the user's unknown preferences on how to interact with the GUI to achieve her goals. The capability of synthesizing multiple plans would also have potential application in case-based planning (e.g., [48]) where it is important to have a plan set satisfying a case instance. These plans can be different in terms of criteria such as resources, makespan and cost that can only be specified in the retrieval phase.

The primary focus of our paper are scenarios where the end user is interested in single plans, but her preferences on that single plan are either unknown or partially known to the planner. Our work shows that an effective technique for handling these scenarios is to generate a set of diverse plans and present them to the user (so she can select the single plan she is most interested in). While we came to sets of plans as an *intermediate step* for handling lack of preference knowledge about single plans, there are also applications where the end user is in fact interested in *sets of plans* (aka "plan-sets"), and has preferences over these plan-sets. Techniques for handling this latter problem do overlap with the techniques we develop in this paper, but it is important to remember their distinct motivations. Fig. 1 makes these distinctions clear by considering two orthogonal dimensions. The X-axis is concerned with whether the end user is interested in single plans or plan-sets. The Y-axis is concerned with the degree of the knowledge of user preferences.

In this space, traditional planning with preferences corresponds to (`single-plan`, `full-knowledge`). The problems we are considering in this paper are (`single-plan`, `no-knowledge`) and (`single-plan`, `partial-knowledge`), respectively. A contribution of our work is to show that these two latter problems can be reformulated as (`plan-set`, `full-knowledge`), where the quality of plan-sets is evaluated by the internal diversity measures we will develop. There are also compelling motivations to study the (`plan-set`, `full-knowledge`) problem in its own right if the end user is explicitly interested in plan-sets. This is the case, for example, in applications such as intrusion detection [5], where the objective is to come up with a set of plans that can inhibit system breaches, or option generation in mission planning, where the commander wants a set of options not to immediately commit to one of them, but rather to study their trade-offs.

The techniques we develop in this paper are related but not equivalent to the techniques and inputs for solving that plan-set generation problem. In particular, when the end users are interested in plan sets, they may have preferences on plan-sets, not on single plans.[5] This means that (i) we need a language support for expressing preferences on plan sets such as the work on DD-PREF language [18], and (ii) our planner has to take as input and support a wide variety of plan-set preferences (in contrast to our current system where the plan-set preference is decided internally — in terms of distance measures for unknown (single plan) preference case, and in terms of IPF measure for partially known preference cases).[6]

3. Background and notation

Given a planning problem with the set of solution plans $\mathcal{S}$, a user preference *model* is a transitive, reflexive relation in $\mathcal{S} \times \mathcal{S}$, which defines an ordering between two plans p and p' in $\mathcal{S}$. Intuitively, $p \preceq p'$ means that the user prefers p at least as much as p'. Note that this ordering can be either partial (i.e. it is possible that neither $p \preceq p'$ nor $p' \preceq p$ holds — in other words, they are incomparable), or total (i.e. either $p \preceq p'$ or $p' \preceq p$ holds). A plan p is considered (strictly) more preferred than a plan p', denoted by $p \prec p'$, if $p \preceq p'$, $p' \not\preceq p$, and they are equally preferred if $p \preceq p'$ and $p' \preceq p$. A plan p is an optimal (i.e., most preferred) plan if $p \preceq p'$ for any other plan p'. A plan set $\mathcal{P} \subseteq \mathcal{S}$ is considered more preferred than $\mathcal{P}' \subseteq \mathcal{S}$, denoted by $\mathcal{P} \ll \mathcal{P}'$, if $p \prec p'$ for any $p \in \mathcal{P}$ and $p' \in \mathcal{P}'$, and they are incomparable if there exists $p \in \mathcal{P}$ and $p' \in \mathcal{P}'$ such that p and p' are incomparable.

The ordering $\preceq$ implies a partition of $\mathcal{S}$ into disjoint plan sets (or *classes*) $\mathcal{S}_0, \mathcal{S}_1, \ldots$ ($\mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots = \mathcal{S}$, $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$) such that plans in the same set are equally preferred, and for any set $\mathcal{S}_i$, $\mathcal{S}_j$, either $\mathcal{S}_i \ll \mathcal{S}_j$, $\mathcal{S}_j \ll \mathcal{S}_i$, or they are incomparable. The partial ordering between these sets can be represented as a Hasse diagram [3] where the sets are vertices, and there is an (upward) edge from $\mathcal{S}_j$ to $\mathcal{S}_i$ if $\mathcal{S}_i \ll \mathcal{S}_j$ and there is not any $\mathcal{S}_k$ in the partition such that $\mathcal{S}_i \ll \mathcal{S}_k \ll \mathcal{S}_j$. We denote $l(\mathcal{S}_i)$ as the "layer" of the set $\mathcal{S}_i$ in the diagram, assuming that the most preferred sets are placed at the layer 0, and $l(\mathcal{S}_j) = l(\mathcal{S}_i) + 1$ if there is an edge from $\mathcal{S}_j$ to $\mathcal{S}_i$. A plan in a set at a layer of smaller value, in general, is either more preferred than or incomparable with ones at layers of higher values.[7] Fig. 2 shows two examples of Hasse diagrams representing a total and partial preference ordering between plans. We will use this representation of plan sets in Section 4 to justify the design of our quality measures for plan sets when no knowledge of user preferences is available.

The preference model of a user can be explicitly specified by iterating the set of plans and providing the ordering between any two of them, and in this case answering queries such as comparing two plans, finding a most preferred (optimal) plan becomes an easy task. This is, however, practically infeasible since synthesizing a plan in itself is hard, and the solution space of a planning problem can be infinite. Many preference *languages*, therefore, have been proposed to represent the relation $\preceq$ in a more compact way, and serve as starting points for *algorithms* to answer queries. Most preference languages fall into the following two categories:

- Quantitative languages define a *value function* $V : \mathcal{S} \rightarrow R$ which assigns a real number to each plan, with a precise interpretation that $p \preceq p' \Leftrightarrow V(p) \leqslant V(p')$. Although this function is defined differently in many languages, at a high level it combines the user preferences on various aspects of plan that can be measured quantitatively. For instance, in the context of decision-theoretic planning [8], the value function of a policy is defined as the expected rewards of states that are visited when the policy executes. In partial satisfaction (over-subscription) planning (PSP) [49,53], the quality of plans is defined as its total rewards of soft goals achieved minus its total action costs. In PDDL2.1 [23], the value function is an arithmetic function of numerical fluents such as plan makespans, fuel used etc., and in PDDL3 [26] it is enhanced with individual preference specifications over state trajectory constraints, defined as formulae with modal operators having their semantics consistent with that used for modal operators in linear temporal logic [43] and other modal temporal logics.

[5] This is akin to a college having explicit preferences on its freshman classes — such as student body diversity — over and above their preferences on individual students.

[6] As an analogy, partial order planning was originally invented to speed up plan generation in classical planning — where the end solutions are all action sequences. Of course, the *technique* of partial order planning is also useful when the end user is interested not in action sequences but *concurrent plans*. In this case however, we need a preference language that allows the user to express preferences over concurrent plans, and we will also have to relax some specific simplifications in normal partial order planners — such as single contributor causal link semantics. Another interesting analogy is between MDP with discrete state space which becomes POMDPS in the context of partial observability. The POMDPS can be handled by compiling them back to MDPs but with continuous state spaces (specifically, MDPs in the space of belief states). It is also possible for an end user to be interested in (fully observable) MDPs in continuous state spaces. While this problem is related to the problem of solving POMDPs as MDPs in belief-space, it also has important differences. In particular, the reward function in the continuous MDP will be in terms of continuous states, while in the case of POMDPs is still in terms of the underlying discrete states. Further, some of the efficiency tricks that the techniques for POMDPs employ based on the fact that the value function has to be convex in the belief-space will no longer hold in general continuous MDPs.

[7] If $\preceq$ is a total ordering, then plans at a layer of smaller value are strictly more preferred than ones at a layer of higher value.

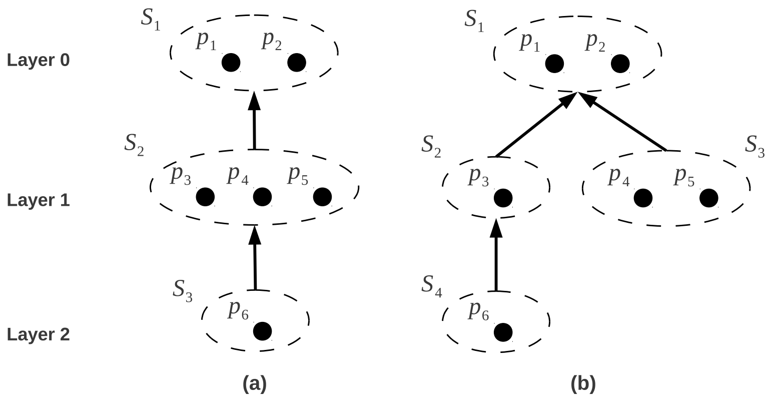


Fig. 2. The Hasse diagrams and layers of plan sets implied by two preference models. In (a), $\mathcal{S}_1 \ll \mathcal{S}_2 \ll \mathcal{S}_3$, and any two plans are comparable. In (b), on the other hand, $\mathcal{S}_1 \ll \mathcal{S}_2 \ll \mathcal{S}_4$, $\mathcal{S}_1 \ll \mathcal{S}_3$, and each plan in $\mathcal{S}_3$ is incomparable with plans in $\mathcal{S}_2$ and $\mathcal{S}_4$.

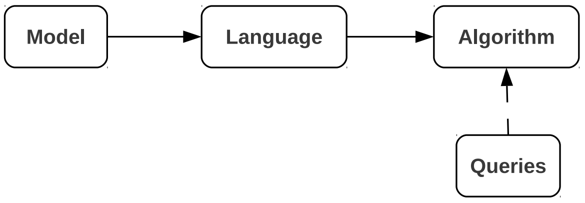


Fig. 3. The metamodel [11]. The user preference model is compactly represented by a preference language, on which algorithms perform tasks of answering queries.

- Qualitative languages provide qualitative statements that are more intuitive for lay users to specify. A commonly used language of this type is *CP-networks* [7], where the user can specify her preference statements on values of plan attributes, possibly given specification of others (for instance, “Among tickets with the same prices, I prefer airline A to airline B.”). Another example is *LPP* [2] in which the statements can be specified using LTL formulae, and possibly being aggregated in different ways.

Fig. 3 shows the conceptual relation of preference models, languages and algorithms. We refer the reader to the work by Brafman and Domshlak [11] for a more detailed discussion on this metamodel, and by Baier and McIlraith [1] for an overview of different preference languages used in planning with preferences.

From the modeling point of view, in order to design a suitable language capturing the user preference model, the modeler should be provided with some knowledge of the user’s interest that affects the way she evaluates plans (for instance, flight duration and ticket cost in a travel planning scenario). Such knowledge in many cases, however, cannot be completely specified. Our purpose therefore is to present a bounded set of plans to the user in the hope that it will increase the chance that she can find a desired plan. In the next section, we formalize the quality measures for plan sets in two situations where either no knowledge of the user preferences or only part of them is given.

4. Quality measures for plan sets

4.1. Syntactic distance measures for unknown preference cases

We first consider the situation in which the user has some preferences for solution plans, but the planner is not provided with any knowledge of such preferences. It is therefore impossible for the planner to assume any particular form of preference language representing the hidden preference model. There are two issues that need to be considered in formalizing a quality measure for plan sets:

- What are the elements of plans that can be involved in a quality measure?
- How should a quality measure be defined using those elements?

For the first question, we observe that even though users are normally interested in some *high level* features of plans that are relevant to them, many of those features can be considered as “functions” of *base level* elements of plans. For instance, the set of actions in the plan determines the makespan of a (sequential) plan, and the sequence of states when the plan executes gives the total reward of goals achieved. We consider the following three types of base level features of plans which could be used in defining quality measure, independently of the domain semantics:

Table 1
The pros and cons of different base level elements of plan.

Basis	Pros	Cons
Actions	Does not require problem information	No problem information is used
States	Not dependent on any specific plan representation	Needs an execution simulator to identify states
Causal links	Considers causal proximity of state transitions (action) rather than positional (physical) proximity	Requires domain theory

- *Actions that are present in plans*, which define various high level features of the plans such as its makespan, execution cost, etc. that are of interest to the user whose preference model could be represented with preference languages such as in PSP and PDDL2.1.
- *Sequence of states that the agent goes through, which captures the behaviors resulting from the execution of plans.* In many preference languages defined using high level features of plans such as the reward of goals collected (e.g., PSP), of the whole state (e.g., MDP), or the temporal relation between propositions occurring in states (e.g. PDDL3, $\mathcal{PP}$ [50] and *LPP* [24]), the sequence of states can affect the quality of plan evaluated by the user.
- *The causal links representing how actions contribute to the goals being achieved, which measures the causal structures of plans.*[8] These plan elements can affect the quality of plans with respect to the languages mentioned above, as the causal links capture both the actions appearing in a plan and the temporal relation between actions and variables.

A similar conceptual separation of features has also been considered recently in the context of case-based planning by Serina [48], in which planning problems were assumed to be well classified, in terms of costs to adapt plans of one problem to solve another, in some *unknown* high level feature space. The similarity between problems in the space was implicitly defined using kernel functions of their domain-independent graph representations. In our situation, we aim to approximate quality of plan sets on the space of features that the user is interested by using distance between plans with respect to the base level features mentioned above.

Table 1 gives the pros and cons of using the different base level elements of plan. We note that if actions in the plans are used in defining quality measure of plan sets, no additional problem or domain theory information is needed. If plan behaviors are used as base level elements, the representation of the plans that bring about state transition becomes irrelevant since only the actual states that an execution of the plan will go through are considered. Hence, we can now compare plans of different representations, e.g., four plans where the first is a deterministic plan, the second is a contingent plan, the third is a hierarchical plan and the fourth is a policy encoding probabilistic behavior. If causal links are used, then the causal proximity among actions is now considered rather than just physical proximity in the plan.

Given those base level elements, the next question is how to define a quality measure of plan sets using them. Recall that without any knowledge of the user preferences, there is no way for the planner to assume any particular preference language, thus the motivation behind the choice of quality measure should come from the hidden user preference model. Given a Hasse diagram induced from the user preference model, a k-plan set that will be presented to the user can be considered to be randomly selected from the diagram. The probability of having one plan in the set classified in a class at the optimal layer of the Hasse diagram would increase when the individual plans are more likely to be at different layers, and this chance in turn will increase if they are less likely to be equally preferred by the user.[9] On the other hand, the effect of base level elements of a plan on high level features relevant to the user suggests that *plans similar with respect to base level features are more likely to be close to each other on the high level feature space determining the user preference model.*

In order to define a quality measure using base level features of plans, we proceed with the following assumption: *plans that are different from each other with respect to the base level features are less likely to be equally preferred by the user, in other words they are more likely to be at different layers of the Hasse diagram*. With the purpose of increasing the chance of having a plan that the user prefers, we propose the quality measure of plan sets as its *diversity* measure, defined using the distance between two plans in the set with respect to a base level element. More formally, the quality measure $\zeta : 2^{\mathcal{S}} \rightarrow \mathbb{R}$ of a plan set $\mathcal{P}$ can be defined as either the minimal, maximal, or average distance between plans:

- minimal distance:

$$\zeta_{min}(\mathcal{P}) = \min_{p,p' \in \mathcal{P}} \delta\big(p, p'\big), \tag{1}$$

[8] A causal link $a_1 \rightarrow p - a_2$ records that a proposition p is produced by a_1 and consumed by a_2.

[9] To see this, consider a diagram with $\mathcal{S}_1 = \{p_1, p_2\}$ at layer 0, $\mathcal{S}_2 = \{p_3\}$ and $\mathcal{S}_3 = \{p_4\}$ at layer 1, and $\mathcal{S}_4 = \{p_5\}$ at layer 2. Assuming that we randomly select a set of 2 plans. If those plans are known to be at the same layer, then the chance of having one plan at layer 0 is $\frac{1}{2}$. However, if they are forced to be at different layers, then the probability will be $\frac{3}{4}$.

- maximal distance:

$$\zeta_{max}(\mathcal{P}) = \max_{p,p' \in \mathcal{P}} \delta(p, p'), \tag{2}$$

- average distance:

$$\zeta_{avg}(\mathcal{P}) = \binom{|\mathcal{P}|}{2}^{-1} \times \sum_{p,p' \in \mathcal{P}} \delta(p, p'), \tag{3}$$

where $\delta : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is the distance measures between two plans.

4.1.1. *Distance measures between plans*

There are various choices on how to define the distance measure $\delta(p, p')$ between two plans using plan actions, sequence of states or causal links, and each way can have different impact on the diversity of plan set on the Hasse diagram. In the following, we propose distance measures in which a plan is considered as (i) a set of actions and causal links, or (ii) sequence of states the agent goes through, which could be used independently of plan representation (e.g. total order, partial order plans).

- *Plan as a set of actions or causal links*: given a plan p, let $A(p)$ and $C(p)$ be the set of actions or causal links of p. The distance between two plans p and p' can be defined as the ratio of the number of actions (causal links) that do not appear in both plans to the total number of actions (causal links) appearing in one of them:

$$\delta_a(p, p') = 1 - \frac{|A(p) \cap A(p')|}{|A(p) \cup A(p')|}, \tag{4}$$

$$\delta_{cl}(p, p') = 1 - \frac{|C(p) \cap C(p')|}{|C(p) \cup C(p')|}. \tag{5}$$

- *Plan as a sequence of states*: given two sequences of states $(s_0, s_1, \ldots, s_k)$ and $(s'_0, s'_1, \ldots, s'_{k'})$ resulting from executing two plans p and p', and assume that $k' \leqslant k$. Since the two sequences of states may have different lengths, there are various options in defining distance measure between p and p', and we consider here two options. In the first one, it can be defined as the average of the distances between state pairs (s_i, s'_i) $(0 \leqslant i \leqslant k')$, and each state $s_{k'+1}, \ldots, s_k$ is considered to contribute maximally (i.e., one unit) into the difference between two plans:

$$\delta_s(p, p') = \frac{1}{k} \times \left[\sum_{i=1}^{k'} \Delta(s_i, s'_i) + k - k' \right]. \tag{6}$$

On the other hand, we can assume that the agent continues to stay at the goal state $s'_{k'}$ in the next $(k - k')$ time steps after executing p', and the measure can be defined as follows:

$$\delta_s(p, p') = \frac{1}{k} \times \left[\sum_{i=1}^{k'} \Delta(s_i, s'_i) + \sum_{i=k'+1}^{k} \Delta(s_i, s'_{k'}) \right]. \tag{7}$$

The distance measure $\Delta(s, s')$ between two states s, s' used in those two measures is defined as:

$$\Delta(s, s') = 1 - \frac{|s \cap s'|}{|s \cup s'|}. \tag{8}$$

These distance metrics would consider long plans to be distant from short plans. In the absence of information about user preferences, we cannot rule out the possibility that the unknown preference might actually favor longer plans (e.g., it is possible that a longer plan has cheaper actions, making it attractive for the user). In the implementation of a system for computing diverse plans, while these distance measures affect which part of the (partial plan) search space a planner tends to focus on, in general the length of resulting plans especially depends on whether the search strategy of the planner attempts to minimize it. In our experiments, we will use two types of planners employing exhaustive search and local search, respectively. For the second, which does not attempt to minimize plan length, we will introduce additional constraints into the search mechanism that, by balancing the differences in the generated diverse plans, also attempts to control the relative size of resulting plans.

Example. Fig. 4 shows three plans p_1, p_2 and p_3 for a planning problem where the initial state is $\{r_1\}$ and the goal propositions are $\{r_3, r_4\}$. The specification of actions are shown in the table. The action sets of the first two plans ($\{a_1, a_2, a_3\}$ and $\{a_1, a_2, a_4\}$) are quite similar ($\delta_a(p_1, p_2) = 0.5$), but the causal links which involve a_3 ($a_2 \to r_3 - a_3$, $a_3 \to r_4 - a_G$)

Initial state: $\{r_1\}$

Goals: $\{r_3, r_4\}$

Action	Pre	Add	Delete
a_1	r_1	r_2	r_1
a_2	r_2	r_3	
a_3	r_3	r_4	
a_4	r_1	r_4	
a_5	r_1	r_2	
a_6	r_2	r_3, r_4	r_1

State sequence:

$p_1: (\{r_1\}, \{r_2\}, \{r_2, r_3\}, \{r_2, r_3, r_4\})$

$p_2: (\{r_1\}, \{r_2, r_4\}, \{r_2, r_3, r_4\})$

$p_3: (\{r_1\}, \{r_1, r_2\}, \{r_3, r_4\})$

Causal links:

$p_1: \{a_I \to r_1 - a_1, a_1 \to r_2 - a_2, a_2 \to r_3 - a_3, a_2 \to r_3 - a_G, a_3 \to r_4 - a_G\}$

$p_2: \{a_I \to r_1 - a_1, a_I \to r_1 - a_4, a_1 \to r_2 - a_2, a_2 \to r_3 - a_G, a_4 \to r_4 - a_G\}$

$p_3: \{a_I \to r_1 - a_5, a_5 \to r_2 - a_6, a_6 \to r_3 - a_G, a_6 \to r_4 - a_G\}$

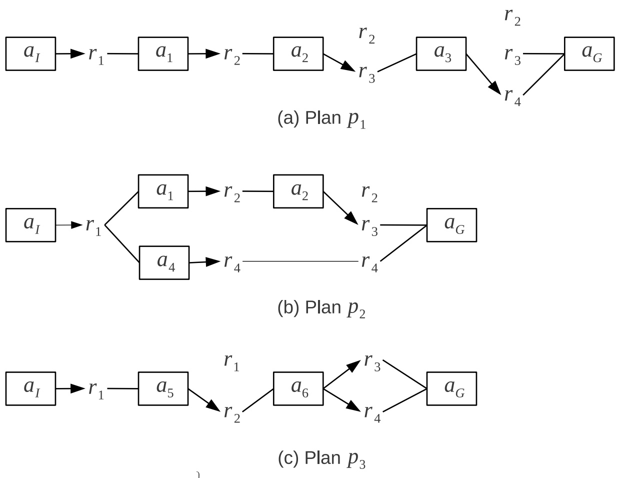


Fig. 4. Example illustrating plans with base-level elements. a_I and a_G denote dummy actions producing the initial state and consuming the goal propositions, respectively (see text for more details).

and a_4 ($a_I \to r_1 - a_4$, $a_4 \to r_4 - a_G$) make their difference more significant with respect to causal-link based distance ($\delta_{cl}(p_1, p_2) = \frac{4}{7}$). Two other plans p_1 and p_3, on the other hand, are very different in terms of action sets (and therefore the sets of causal links): $\delta_a(p_1, p_3) = 1$, but they are closer in term of state-based distance ($\frac{13}{18}$ as defined in Eq. (6), and 0.5 if defined in Eq. (7)).

4.2. Integrated Preference Function (IPF) for partial preference cases

We now discuss a quality measure for plan sets in the case when the user preference is partially expressed. In particular, we consider scenarios in which the preference model can be represented by some quantitative language with an incompletely specified value function of high level features. As an example, the quality of plans in PDDL2.1 [23] and PDDL3 [26] are represented by a metric function combining metric fluents and preference statements on state trajectory with parameters representing their relative importance. While providing a convenient way to represent preference models, such parameterized value functions present an issue of obtaining reasonable values for the relative importance of the features. A common approach to model this type of incomplete knowledge is to consider those parameters as a vector of random variables, whose values are assumed to be drawn from a distribution. This is the representation that we will follow.

To measure the quality of plan sets, we propose the usage of *Integrated Preference Function* (IPF) [14], which has been used to measure the quality of a solution set in a wide range of multi-objective optimization problems. The IPF measure assumes that the user preference model can be represented by two factors: (1) a probability distribution $h(\alpha)$ of parameter vector α, whose domain is denoted by Λ, such that $\int_{\alpha \in \Lambda} h(\alpha)\, d\alpha = 1$ (in the absence of any special information about the distribution, $h(\alpha)$ can be assumed to be uniform), and (2) a value function $V(p, \alpha) : \mathcal{S} \times \Lambda \to \mathbb{R}$ combines different objective functions into a single real-valued quality measure for plan p. We assume that such objective functions represent aspects of plans that have to be minimized, such as makespan and execution cost. This incomplete specification of the value function represents a set of candidate preference models, for each of which the user will select a different plan, the one with the best value, from a given plan set $\mathcal{P} \subseteq \mathcal{S}$. The IPF value of solution set $\mathcal{P}$ is defined as:

$$IPF(\mathcal{P}) = \int_{\alpha \in \Lambda} h(\alpha) V(p_\alpha, \alpha)\, d\alpha, \tag{9}$$

with $p_\alpha = \operatorname{argmin}_{p \in \mathcal{P}} V(p, \alpha)$, i.e., the best solution in $\mathcal{P}$ according to $V(p, \alpha)$ for each given α value. Let p_α^{-1} be a range of α values for which p is an optimal solution according to $V(p, \alpha)$, i.e., $V(p, \alpha) \leqslant V(p', \alpha)$ for all $\alpha \in p_\alpha^{-1}$, $p' \in \mathcal{P} \setminus \{p\}$.

As p_α is piecewise constant, the $IPF(\mathcal{P})$ value can be computed as:

$$IPF(\mathcal{P}) = \sum_{p \in \mathcal{P}} \left[\int_{\alpha \in p_\alpha^{-1}} h(\alpha) V(p, \alpha)\, d\alpha \right]. \tag{10}$$

Let $\mathcal{P}^* = \{p \in \mathcal{P}:\ p_\alpha^{-1} \neq \emptyset\}$; then we have:

$$IPF(\mathcal{P}) = IPF(\mathcal{P}^*) = \sum_{p \in \mathcal{P}^*} \left[\int_{\alpha \in p_\alpha^{-1}} h(\alpha) V(p, \alpha)\, d\alpha \right]. \tag{11}$$

Since $\mathcal{P}^*$ is the set of plans that are optimal for some specific parameter vector, $IPF(\mathcal{P})$ now can be interpreted as the expected value that the user can get by selecting the best plan in $\mathcal{P}$. Therefore, the set $\mathcal{P}^*$ of solutions (known as *lower convex hull* of $\mathcal{P}$) with the minimal IPF value is most likely to contain the desired solutions that the user wants, and in essence it is a good representative of the plan set $\mathcal{P}$.

The requirement for $IPF(\mathcal{P})$ to exist is that the function $h(\alpha)V(p,\alpha)$ needs to be integrable over the p_α^{-1} domains.[10] The complication in computing the $IPF(\mathcal{P})$ value is in deriving a partition of Λ, the domain of α, into the ranges p_α^{-1} for $p \in \mathcal{P}^*$, and the computation of integration over those ranges of the parameter vector. As we will describe, the computational effort to obtain $IPF(\mathcal{P})$ is negligible in our work with two objectives. Although it is beyond the scope of this article, we refer the readers to Kim et al. [35] for the calculation of the measure when the value function is a convex combination of high number of objectives, and to Bozkurt et al. [10] for the weighted Tchebycheff value function with two and three criteria.

In this work, in order to make our discussion on generating plan sets concrete, we will concentrate on metric temporal planning where each action $a \in A$ has a duration d_a and an execution cost c_a. The planner needs to find a plan $p = \langle a_1, \ldots, a_n \rangle$, which is a sequence of actions that is executable and achieves all goals. The two most common plan quality measures are: *makespan*, the total execution time of p, *plan cost*, the total execution cost of all actions in p. Both of them are high level features that can be affected by the actions in the plan. In most real-world applications, these two criteria compete with each other in that shorter plans usually have higher cost and vice versa. We use the following assumptions:

- The desired objective function involves minimizing both components: $time(p)$ measures the makespan of the plan p and $cost(p)$ measures its execution cost.
- The quality of a plan p is a convex combination: $V(p, w) = w \times time(p) + (1 - w) \times cost(p)$, where weight $w \in [0, 1]$ represents the trade-off between the two competing objective functions.
- The belief distribution of w over the range $[0, 1]$ is known. If the user does not provide any information or we have not learned anything about the preference on the trade-off between *time* and *cost* of the plan, then the planner can assume a uniform distribution (and improve it later using techniques such as preference elicitation).

Given that the exact value of w is unknown, our purpose is to find a bounded representative set of non-dominated[11] plans minimizing the expected value of $V(p, w)$ with regard to the given distribution of w over $[0, 1]$.

IPF for metric temporal planning: The user preference model in our target domain of temporal planning is represented by a convex combination of the *time* and *cost* quality measures, and the IPF measure now is called *Integrated Convex Preference* (ICP). Given a set of plans $\mathcal{P}^*$, let $t_p = time(p)$ and $c_p = cost(p)$ be the makespan and total execution cost of plan $p \in \mathcal{P}^*$, the ICP value of $\mathcal{P}^*$ with regard to the objective function $V(p, w) = w \times t_p + (1 - w) \times c_p$ and the parameter vector $\alpha = (w, 1 - w)$ $(w \in [0, 1])$ is defined as:

$$ICP(\mathcal{P}^*) = \sum_{i=1}^{k} \int_{w_{i-1}}^{w_i} h(w)\big(w \times t_{p_i} + (1 - w) \times c_{p_i}\big)\, dw, \tag{12}$$

where $w_0 = 0$, $w_k = 1$ and $V(p_i, w) \leqslant V(p, w)$ for all $p \in \mathcal{P}^* \setminus \{p_i\}$ and every $w \in [w_{i-1}, w_i]$. In other words, we divide $[0, 1]$ into k non-overlapping regions such that in each region (w_{i-1}, w_i) there is an optimal solution $p_i \in \mathcal{P}^*$ according to the value function.

We select the IPF/ICP measure to evaluate our solution set for the following reasons:

- From the perspective of *decision theory*, presenting a plan set $\mathcal{P} \subseteq \mathcal{S}$ to the user, among all possible subsets of $\mathcal{S}$, can be considered as an "action" with possible "outcomes" $p \in \mathcal{P}$ that can occur (i.e., being selected by the user) with probability $\int_{\alpha \in p_\alpha^{-1}} h(\alpha)\, d\alpha$. Since the $IPF(\mathcal{P})$ measures the expected utility of $\mathcal{P}$, presenting a set of plans with an optimal IPF value is a rational action consistent with the current knowledge of the user preferences.
- If $\mathcal{P}_1$ dominates $\mathcal{P}_2$ in the *set Pareto dominance* sense, then $IPF(\mathcal{P}_1) \leqslant IPF(\mathcal{P}_2)$ for any type of weight density function $h(\alpha)$ [14], and this property also holds with any scaling of the objective values for ICP measure [21]. Intuitively, this means that if we "merge" those two plan sets, all non-dominated plans "extracted" from the resulting set are those in $\mathcal{P}_1$.

[10] Although a value function can take any form satisfying axioms about preferences, the user preferences in many real-world scenarios can be represented or approximated with an *additive value function* [47], including the setting in our application, which is integrable over the parameter domain. Since $h(\alpha)$ is integrable, so is the product $h(\alpha)V(p,\alpha)$ in those situations.

[11] A plan p_1 is dominated by p_2 if $time(p_1) \geqslant time(p_2)$ and $cost(p_1) \geqslant cost(p_2)$ and at least one of the inequalities is strict.

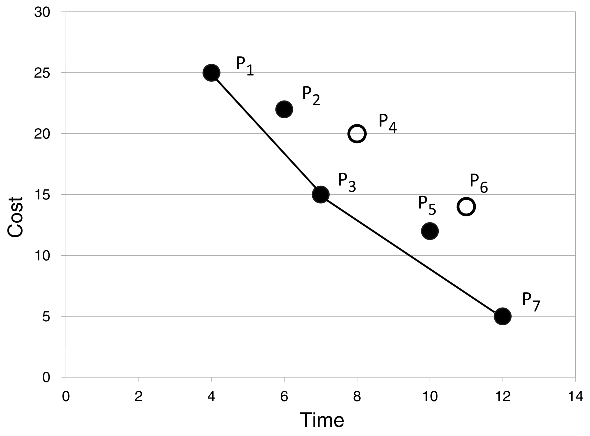


Fig. 5. Solid dots represent plans in the Pareto set (p_1, p_2, p_3, p_5, p_7). Connected dots represent plans in the lower convex hull (p_1, p_3, p_7) giving optimal ICP value for any distribution on trade-off between *cost* and *time*.

- The value of $IPF(\mathcal{P})$ is monotonically non-increasing over increasing sequences of solution sets, and the set of plans optimal according to the utility function $V(p,\alpha)$, i.e., the efficient frontier, has the minimal *IPF* value [14]. Thus, the measure can be used as an indicator for the quality of a plan set during the search towards the efficient frontier.

Empirically, extensive results on scheduling problems in [21] have shown that ICP measure "*evaluates the solution quality of approximation robustly* (*i.e., similar to visual comparison results*) *while other alternative measures can misjudge the solution quality*".

Example. Fig. 5 shows our running example in which there are a total of 7 plans with their *time*(p) and *cost*(p) values as follows: $p_1 = \{4, 25\}$, $p_2 = \{6, 22\}$, $p_3 = \{7, 15\}$, $p_4 = \{8, 20\}$, $p_5 = \{10, 12\}$, $p_6 = \{11, 14\}$, and $p_7 = \{12, 5\}$. Among these 7 plans, 5 of them belong to a Pareto optimal set of non-dominated plans: $\mathcal{P}_p = \{p_1, p_2, p_3, p_5, p_7\}$. The other two plans are dominated by some plans in $\mathcal{P}_p$: p_4 is dominated by p_3 and p_6 is dominated by p_5. Plans in $\mathcal{P}_p$ are depicted in solid dots, and the set of plans $\mathcal{P}^* = \{p_1, p_3, p_7\}$ that are optimal for some specific value of w is highlighted by connected dots. In particular, p_7 is optimal when $w \in [w_0 = 0, w_1 = \frac{2}{3}]$ where $w_1 = \frac{2}{3}$ can be derived from the satisfaction of the constraints $V(p_7, w) \leqslant V(p, w)$, $p \in \{p_1, p_3\}$. Similarly, p_3 and p_1 are respectively optimal for $w \in [w_1 = \frac{2}{3}, w_2 = \frac{10}{13}]$ and $w \in [w_2 = \frac{10}{13}, w_3 = 1]$. Assuming that $h(w)$ is a uniform distribution, the value of $ICP(\mathcal{P})$ can therefore be computed as follows:

$$\begin{aligned} ICP\big(\mathcal{P}^*\big) &= \int_0^{\frac{2}{3}} h(w)V(p_7, w)\,dw + \int_{\frac{2}{3}}^{\frac{10}{13}} h(w)V(p_3, w)\,dw + \int_{\frac{10}{13}}^{1} h(w)V(p_1, w)\,dw \\ &= \int_0^{\frac{2}{3}} \big[12w + 5(1-w)\big]\,dw + \int_{\frac{2}{3}}^{\frac{10}{13}} \big[7w + 15(1-w)\big]\,dw + \int_{\frac{10}{13}}^{1} \big[4w + 25(1-w)\big]\,dw \\ &\approx 7.32. \end{aligned}$$

In the next two Sections 5 and 6, we investigate the problem of generating high quality plan sets for two cases mentioned: when no knowledge about the user preferences is given, and when part of it is given as input to the planner.

5. Generating diverse plan set in the absence of preference knowledge

In this section, we describe approaches to searching for a set of diverse plans with respect to a measure defined with base level elements of plans as discussed in the previous section. In particular, we consider the quality measure of plan set as the minimal pairwise distance between any two plans, and generate a set of plans containing k plans with the quality of at least a predefined threshold d. As discussed earlier, by diversifying the set of plans on the space of base level features, it is likely that plans in the set would cover a wide range of space of unknown high level features, increasing the possibility that the user can select a plan close to the one that she prefers. The problem is formally defined as follows:

$$d\text{DISTANT}k\text{SET: Find } \mathcal{P} \text{ with } \mathcal{P} \subseteq \mathcal{S},\ |\mathcal{P}| = k \text{ and } \zeta(\mathcal{P}) = \min_{p,q \in \mathcal{P}} \delta(p,q) \geqslant d,$$

where any distance measure between two plans formalized in Section 4.1.1 can be used to implement $\delta(p, p')$.

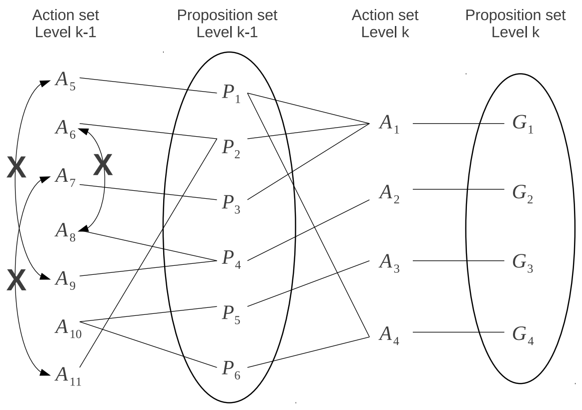


Fig. 6. An example of (a portion of) a planning graph. At each level, propositions presenting in a previous one and no-op actions are omitted, and at level k only the actions used to support the goals are shown for simplification.

We now consider two representative state-of-the-art planning approaches in generating diverse plan sets. The first one is GP-CSP [19] representing constraint-based planning approaches, and the second one is LPG [27] that uses an efficient local-search based approach. We use GP-CSP to compare the relation between different distance measures in diversifying plan sets. On the other hand, with LPG we stick to the action-based distance measure, which is shown experimentally to be the most difficult measure to enforce diversity (see below), and investigate the scalability of heuristic approaches in generating diverse plans.

5.1. *Finding diverse plan set with GP-CSP*

The GP-CSP planner [19] converts the planning graph of Graphplan [4] into a CSP encoding, and solves it using a standard CSP solver. A planning graph is a data structure consisting of alternating levels of proposition set and action set. The set of propositions present in the initial state is the proposition set at the zeroth level of the graph. Given a k-level planning graph, all actions whose preconditions are present in the proposition set of level k are introduced into the next level $k+1$. In addition, one "noop" action is also added for each proposition at level k, which are both the precondition and effect of the action. The set of propositions at level $k+1$ is then constructed by taking the union of additive effects of all actions at the same level. This expansion process also computes and propagates a set of "mutex" (i.e., mutually exclusive) constraints between pairs of propositions and actions at each level. At the first level, the computation starts by marking as mutex the actions that are statically interfering with each other (i.e., their preconditions and effects are inconsistent). The mutex constraints are then propagated as follows: (i) at level k, two propositions are mutually exclusive if any action at level k achieving one of them is mutually exclusive with all actions at the same level supporting the other one; (ii) two actions at level $k+1$ are mutex if they are statically interfering or if one of the precondition of the first action is mutually exclusive with one of the precondition of the second action.

The planning graph construction stops at a level T at which one of the following conditions is satisfied: (i) all goal propositions are present in the proposition set of level T without any mutex constraints between them, or (ii) two consecutive levels of the graph have the same sets of actions, propositions and mutex constraints. In the first case, the Graphplan algorithm searches this graph backward (i.e., from level T) for a valid plan, and continuing the planning graph expansion before a new search if no solution exists. In the second condition, the problem is provably unsolvable. Fig. 6, which is taken from Do and Kambhampati [19], shows an example of two levels of a planning graph. The top-level goals are $G_1, \ldots, G_4$ supported by actions $A_1, \ldots, A_4$ at the same level k. Each of these actions has preconditions in the set $\{P_1, \ldots, P_6\}$ appearing at level $k-1$, which are in turn supported by actions $A_5, \ldots, A_{11}$ at that level. The action pairs $\{A_5, A_9\}$, $\{A_7, A_{11}\}$ and $\{A_6, A_8\}$ are mutually exclusive, however these mutex relations are not enough to make any pair of propositions at level $k-1$ mutually exclusive.

The GP-CSP planner replaces the search algorithm in Graphplan by first converting the planning graph data structure into a constraint satisfaction problem, and then invoking a solver to find an assignment of the encoding, which represents a valid plan for the original planning problem. In the encoding, the CSP variables correspond to the predicates that have to be achieved at different levels in the planning graph (different planning steps) and their possible values are the actions that can support the predicates. For each CSP variable representing a predicate p, there are two special values: (i) $\perp$: indicates that a predicate is not supported by any action and is *false* at a particular level/planning-step; (ii) "no-op": indicates that the predicate is true at a given level i because it was made true at some previous level $j < i$ and no other action deletes p between j and i. Constraints encode the relations between predicates and actions: (1) mutual exclusion relations between predicates and actions; and (2) the causal relationships between actions and their preconditions. Fig. 7 shows the CSP encoding corresponding the portion of the planning graph in Fig. 6.

Variables: $G_1, \dots, G_4, P_1, \dots, P_6$

Domains:

$G_1 : \{A_1, \perp\}, G_2 : \{A_2, \perp\}, G_3 : \{A_3, \perp\}, G_4 : \{A_4, \perp\}$
$P_1 : \{A_5, \perp\}, P_2 : \{A_6, A_{11}, \perp\}, P_3 : \{A_7, \perp\}$
$P_4 : \{A_8, A_9, \perp\}, P_5 : \{A_{10}, \perp\}, P_6 : \{A_{10}, \perp\}$

Constraints (Mutex):

$P_1 = A_5 \Rightarrow P_4 \neq A_9$
$P_2 = A_6 \Rightarrow P_4 \neq A_8$
$P_2 = A_{11} \Rightarrow P_3 \neq A_7$

Constraints (Activity):

$G_1 = A_1 \Rightarrow P_1 \neq \perp \wedge P_2 \neq \perp \wedge P_3 \neq \perp$
$G_2 = A_2 \Rightarrow P_4 \neq \perp$
$G_3 = A_3 \Rightarrow P_5 \neq \perp$
$G_4 = A_4 \Rightarrow P_1 \neq \perp \wedge P_6 \neq \perp$

Initial state: $G_1 \neq \perp \wedge G_2 \neq \perp \wedge G_3 \neq \perp \wedge G_4 \neq \perp$

Fig. 7. The CSP encoding for the example planning graph.

5.1.1. *Adapting GP-CSP to different distance metrics*

When the above planning encoding is solved by any standard CSP solver, it will return a solution containing $\langle var, value\rangle$ of the form $\{\langle x_1, y_1\rangle, \dots, \langle x_n, y_n\rangle\}$. The collection of x_i where $y_i \neq \perp$ represents the facts that are made true at different time steps (plan trajectory) and can be used as a basis for the *state-based* distance measure[12]; the set of $(y_i \neq \perp) \wedge (y_i \neq noop)$ represents the set of actions in the plan and can be used for *action-based* distance measure; lastly, the assignments $\langle x_i, y_i\rangle$ themselves represent the causal relations and can be used for the *causal-based* distance measure.

However, there are some technical difficulties we need to overcome before a specific distance measure between plans can be computed. First, the same action can be represented by different values in the domains of different variables. Consider a simple example in which there are two facts p and q, both supported by two actions a_1 and a_2. When setting up the CSP encoding, we assume that the CSP variables x_1 and x_2 are used to represent p and q. The domains for x_1 and x_2 are $\{v_{11}, v_{12}\}$ and $\{v_{21}, v_{22}\}$, both representing the two actions $\{a_1, a_2\}$ (in that order). The assignments $\{\langle x_1, v_{11}\rangle, \langle x_2, v_{21}\rangle\}$ and $\{\langle x_1, v_{12}\rangle, \langle x_2, v_{22}\rangle\}$ have a distance of 2 in traditional CSP because different values are assigned for each variable x_1 and x_2. However, they both represent the same action set $\{a_1, a_2\}$ and thus lead to the plan distance of 0 if we use the action-based distance in our plan comparison. Therefore, we first need to translate the set of values in all assignments back to the set of action instances before doing comparison using action-based distance. The second complication arises for the causal-based distance. A causal link $a_1 \rightarrow p - a_2$ between two actions a_1 and a_2 indicates that a_1 supports the precondition p of a_2. However, the CSP assignment $\langle p, a_1\rangle$ only provides the first half of each causal-link. To complete the causal-link, we need to look at the values of other CSP assignments to identify action a_2 that occurs at the later level in the planning graph and has p as its precondition. Note that there may be multiple "valid" sets of causal-links for a plan, and in the implementation we simply select causal-links based on the CSP assignments.

5.1.2. *Making GP-CSP return a set of plans*

To make GP-CSP return a set of plans satisfying the *d*DISTANT*k*SET constraint using one of the three distance measures, we add "global" constraints to each original encoding to enforce d-diversity between every pair of solutions. When each global constraint is called upon by the normal forward checking and arc-consistency checking procedures inside the default solver to check if the distance between two plans is over a predefined value d, we first map each set of assignments to an actual set of actions (action-based), predicates that are true at different plan-steps (state-based) or causal-links (causal-based) using the method discussed in the previous section. This process is done by mapping all $\langle var, value\rangle$ CSP assignments into action sets using a call to the planning graph, which is outside of the CSP solver, but works closely with the general purpose CSP solver in GP-CSP. The comparison is then done within the implementation of the global constraint to decide if two solutions are diverse enough.

We investigate two different ways to use the global constraints:

1. The *parallel* strategy to return the set of k plans all at once. In this approach, we create one encoding that contains k identical copies of each original planning encoding created using the GP-CSP planner. The k copies are connected together using $k(k-1)/2$ pairwise global constraints. Each global constraint between the ith and jth copies ensures that two plans represented by the solutions of those two copies will be at least d distant from each other. If each copy has n variables, then this constraint involves $2n$ variables.
2. The *greedy* strategy to return plans one after another. In this approach, the k copies are not setup in parallel up-front, but sequentially. We add to the ith copy one global constraint to enforce that the solution of the ith copy should be d-diverse from any of the earlier $i-1$ solutions. The advantage of the greedy approach is that each CSP encoding

[12] We implement the state-based distance between plans as defined in Eq. (6).

Table 2
Average solving time (in seconds) to find a plan using greedy (first 3 rows) and by random (last row) approaches.

	log-easy	rocket-a	log-a	log-b	log-c	log-d
δ_a	0.087	7.648	1.021	6.144	8.083	178.633
δ_s	0.077	9.354	1.845	6.312	8.667	232.475
δ_{cl}	0.190	6.542	1.063	6.314	8.437	209.287
Random	0.327	15.480	8.982	88.040	379.182	6105.510

Table 3
Comparison of the diversity in the plan sets returned by the random and greedy approaches. Cases where random approach is better than greedy approach are marked with *.

	log-easy	rocket-a	log-a	log-b	log-c	log-d
δ_a	0.041/0.35	0.067/0.65	0.067/0.25	0.131/0.1*	0.126/0.15	0.128/0.2
δ_s	0.035/0.4	0.05/0.8	0.096/0.5	0.147/0.4	0.140/0.5	0.101/0.5
δ_{cl}	0.158/0.8	0.136/0.95	0.256/0.55	0.459/0.15*	0.346/0.3*	0.349/0.45

is significantly smaller in terms of the number of variables (n vs. $k \times n$), smaller in terms of the number of global constraints (1 vs. $k(k-1)/2$), and each global constraint also contains lesser number of variables (n vs. $2 \times n$).[13] Thus, each encoding in the greedy approach is easier to solve. However, because each solution depends on all previously found solutions, the encoding can be unsolvable if the previously found solutions comprise a bad initial solution set.

5.1.3. Empirical evaluation

We implemented the parallel and greedy approaches discussed earlier for the three distance measures and tested them with the benchmark set of Logistics problems provided with the Blackbox planner [34]. All experiments were run on a Linux Pentium 4, 3 GHz machine with 512 MB RAM. For each problem, we test with different d values ranging from 0.01 (1%) to 0.95 (95%)[14] and k increases from 2 to n where n is the maximum value for which GP-CSP can still find solutions within the plan horizon. The horizon (parallel plan steps) limit is 30.

We found that the greedy approach outperformed the parallel approach and solved significantly higher number of problems. Therefore, we focus on the greedy approach hereafter. For each combination of d, k, and a given distance measure, we record the solving time and output the average/min/max pairwise distances of the solution sets.

Baseline comparison: As a baseline comparison, we have also implemented a *randomized* approach. In this approach, we do not use global constraints but use random value ordering in the CSP solver to generate k different solutions without enforcing them to be pairwise d-distance apart. For each distance d, we continue running the random algorithm until we find k_{max} solutions where k_{max} is the maximum value of k that we can solve for the greedy approach for that particular d value. In general, we want to compare with our approach of using global constraint to see if the random approach can effectively generate diverse set of solutions by looking at: (1) the average time to find a solution in the solution set; and (2) the maximum/average pairwise distances between $k \geqslant 2$ randomly generated solutions.

Table 2 shows the comparison of average solving time to find one solution in the greedy and random approaches. The results show that on average, the random approach takes significantly more time to find a single solution, regardless of the distance measure used by the greedy approach. To assess the diversity in the solution sets, Table 3 shows the comparison of: (1) the average pairwise minimum distance between the solutions in sets returned by the random approach; and (2) the maximum d for which the greedy approach still can find a set of diverse plans. The comparisons are done for all three distance measures. For example, the first cell (0.041/0.35) in Table 3, implies that the minimum pairwise distance averaged for all solvable $k \geqslant 2$ using the random approach is $d = 0.041$ while it is 0.35 (i.e., $8\times$ more diverse) for the greedy approach using the δ_a distance measure. Except for 3 cases, using global constraints to enforce minimum pairwise distance between solutions helps GP-CSP return significantly more diverse set of solutions. On average, the greedy approach returns $4.25\times$, $7.31\times$, and $2.79\times$ more diverse solutions than the random approach for δ_a, δ_s and δ_{cl}, respectively.

Analysis of the different distance-bases: Overall, we were able to solve 1264 (d, k) combinations for three distance measures $\delta_a, \delta_s, \delta_{cl}$ using the greedy approach. We were particularly interested in investigating the following issues:

- **Q1: Computational efficiency** — Is it easy or difficult to find a set of diverse solutions using different distance measures? Thus, (1) for the same d and k values, which distance measure is more difficult (time consuming) to solve; and (2) given an encoding horizon limit, how high is the value of d and k for which we can still find a set of solutions for a given problem using different distance measures.

[13] However, each constraint is more complicated because it encodes $(i-1)$ previously found solutions.
[14] Increments of 0.01 from 0.01 to 0.1 and of 0.05 thereafter.

Table 4
For each given d value, each cell shows the largest solvable k for each of the three distance measures δ_a, δ_s, and δ_{cl} (in this order). The maximum values in cells are in bold.

d	log-easy	rocket-a	log-a	log-b	log-c	log-d
0.01	11, 5, **28**	8, **18**, 12	9, 8, **18**	3, 4, **5**	4, 6, **8**	**8**, 7, 7
0.03	6, 3, **24**	8, **13**, 9	7, 7, **12**	2, **4**, 3	4, **6**, **6**	4, **7**, 6
0.05	5, 3, **18**	6, **11**, 9	5, 7, **10**	2, **4**, 3	4, **6**, 5	3, **7**, 5
0.07	2, 3, **14**	6, **10**, 8	4, **7**, 6	2, **4**, 2	4, **6**, 5	3, **7**, 5
0.09	2, 3, **14**	6, **9**, 6	3, **6**, **6**	2, **4**, 2	3, **6**, 4	3, **7**, 4
0.1	2, 3, **10**	6, **9**, 6	3, **6**, **6**	2, **4**, 2	2, **6**, 4	3, **7**, 4
0.2	2, 3, **5**	5, **9**, 6	2, **6**, **6**	1, **3**, 1	1, **5**, 2	2, **5**, 3
0.3	2, 2, **3**	4, **7**, 5	1, **4**, **4**	1, **2**, 1	1, **3**, 2	1, **3**, **3**
0.4	1, 2, **3**	3, **6**, 5	1, **3**, **3**	1, **2**, 1	1, **2**, 1	1, 2, **3**
0.5	1, 1, **3**	2, 4, **5**	1, **2**, **2**	–	1, **2**, 1	1, **2**, 1
0.6	1, 1, **2**	2, 3, **4**	–	–	–	–
0.7	1, 1, **2**	1, **2**, **2**	–	–	–	–
0.8	1, 1, **2**	1, **2**, **2**	–	–	–	–
0.9	–	1, 1, **2**	–	–	–	–

Table 5
Cross-validation of distance measures δ_a, δ_s, and δ_{cl}.

	δ_a	δ_s	δ_{cl}
δ_a	–	1.262	1.985
δ_s	0.485	–	0.883
δ_{cl}	0.461	0.938	–

- **Q2: Solution diversity** — What, if any, is the correlation/sensitivity between different distance measures? Thus, how the comparative diversity of solutions is when using different distance measures.

Regarding Q1, Table 4 shows the highest solvable k value for each distance d and base δ_a, δ_s, and δ_{cl}. For a given (d, k) pair, enforcing δ_a appears to be the most difficult, then δ_s, and δ_{cl} is the easiest. GP-CSP is able to solve 237, 462, and 565 combinations of (d, k) respectively for δ_a, δ_s and δ_{cl}. GP-CSP solves *d*DISTANT*k*SET problems more easily with δ_s and δ_{cl} than with δ_a due to the fact that solutions with different action sets (diverse with regard to δ_a) will likely cause different trajectories and causal structures (diverse with regard to δ_s and δ_{cl}). Between δ_s and δ_{cl}, δ_{cl} solves more problems for easier instances (log-easy, rocket-a and log-a) but less for the harder instances, as shown in Table 4. We conjecture that for solutions with more actions (i.e., in bigger problems) there are more causal dependencies between actions and thus it is harder to reorder actions to create a different causal structure.

For running time comparisons, among 216 combinations of (d, k) that were solved by all three distance measures, GP-CSP takes the least amount of time for δ_a in 84 combinations, for δ_s in 70 combinations and in 62 for δ_{cl}. The first three lines of Table 2 show the average time to find one solution in d-diverse k-set for each problem using δ_a, δ_s and δ_{cl} (which we call t_a, t_s and t_c respectively). In general, t_a is the smallest and $t_s > t_c$ in most problems. Thus, while it is harder to enforce δ_a than δ_s and δ_{cl} (as indicated in Table 4), when the encodings for all three distances can be solved for a given (d, k), then δ_a takes less time to search for one plan in the diverse plan set; this can be due to tighter constraints (more pruning power for the global constraints) and simpler global constraint setting.

To test Q2, in Table 5, we show the cross-comparison between different distance measures δ_a, δ_s, and δ_{cl}. In this table, cell $\langle row, column\rangle = \langle \delta', \delta''\rangle$ indicates that over all combinations of (d, k) solved for distance δ', the average value d''/d' where d'' and d' are distance measured according to δ'' and δ', respectively ($d' \geqslant d$). For example, $\langle \delta_s, \delta_a\rangle = 0.485$ means that over 462 combinations of (d, k) solvable for δ_s, for each d, the average distance between k solutions measured by δ_a is $0.485 \times d_s$. The results indicate that when we enforce d for δ_a, we will likely find even more diverse solution sets according to δ_s ($1.26 \times d_a$) and δ_{cl} ($1.98 \times d_a$). However, when we enforce d for either δ_s or δ_{cl}, we are not likely to find a more diverse set of solutions measured by the other two distance measures. Nevertheless, enforcing d using δ_{cl} will likely give comparable diverse degree d for δ_s ($0.94 \times d_c$) and vice versa. We also observe that d_s is highly dependent on the difference between the parallel lengths of plans in the set. The distance d_s seems to be the smallest (i.e., $d_s < d_a < d_c$) when all k plans have the same/similar number of time steps. This is consistent with the fact that δ_a and δ_{cl} do not depend on the steps in the plan execution trajectory while δ_s does.

5.2. Finding diverse plan set with LPG

In this section, we consider the problem of generating diverse sets of plans using another planning approach, in particular the LPG planner which is able to scale up to bigger problems, compared to GP-CSP. We focus on the action-based distance measure between plans, which has been shown in the previous section to be the most difficult to enforce diversity. LPG is a local-search-based planner, that incrementally modifies a partial plan in a search for a plan that contains no flaws [27].

The behavior of LPG is controlled by an evaluation function that is used to select between different plan candidates in a neighborhood generated for local search. At each search step, the elements in the search neighborhood of the current partial plan π are the alternative possible plans repairing a selected flaw in π. The elements of the neighborhood are evaluated according to an *action evaluation function* E [27]. This function is used to estimate the cost of either adding or of removing an action node a in the partial plan p being generated.

5.2.1. Revised evaluation function for diverse plans

In order to manage $d$DISTANCE$k$SET problems, the function E has been extended to include an additional evaluation term that has the purpose of penalizing the insertion and removal of actions that *decrease* the distance of the current partial plan p under adaptation from a reference plan p_0. In general, E consists of four weighted terms, evaluating four aspects of the quality of the current plan that are affected by the addition $(E(a)^i)$ or removal $(E(a)^r)$ of a

$$E(a)^i = \alpha_E \cdot \mathit{Execution\_cost}(a)^i + \alpha_T \cdot \mathit{Temporal\_cost}(a)^i + \alpha_S \cdot \mathit{Search\_cost}(a)^i + \alpha_D \cdot \left|(p_0 - p) \cap p_R^i\right|,$$
$$E(a)^r = \alpha_E \cdot \mathit{Execution\_cost}(a)^r + \alpha_T \cdot \mathit{Temporal\_cost}(a)^r + \alpha_S \cdot \mathit{Search\_cost}(a)^r + \alpha_D \cdot \left|(p_0 - p - a) \cap p_R^r\right|.$$

The first three terms of the two forms of E are unchanged from the standard behavior of LPG. The fourth term, used only for computing diverse plans, is the new term estimating how the proposed plan modification will affect the distance from the reference plan p_0. Each cost term in E is computed using a relaxed temporal plan p_R [27].

The p_R plans are computed by an algorithm, called RelaxedPlan, formally described and illustrated in [27]. We have slightly modified this algorithm to penalize the selection of actions decreasing the plan distance from the reference plan. The specific change to RelaxedPlan for computing diverse plans is very similar to the change described in [22], and it concerns the heuristic function for selecting the actions for achieving the subgoals in the relaxed plans. In the modified function for RelaxedPlan, we have an extra 0/1 term that penalizes an action b for p_R if its addition decreases the distance of $p + p_R$ from p_0 (in the plan repair context investigated in [22], b is penalized if its addition *increases* such a distance).

The last term of the modified evaluation function E is a measure of the decrease in plan distance caused by adding or removing a: $|(p_0 - p) \cap p_R^i|$ or $|(p_0 - p - a) \cap p_R^r|$, where p_R^i contains the new action a. The α-coefficients of the E-terms are used to weigh their relative importance.[15] The values of the first 3 terms are automatically derived from the expression defining the plan metric for the problem [27]. The coefficient for the fourth new term of E (α_D) is automatically set during search to a value proportional to $d/\delta_a(p, p_0)$, where p is the current partial plan under construction. The general idea is to dynamically increase the value of α_D according to the number of plans n that have been generated so far: if n is much higher than k, the search process consists of finding many solutions with not enough diversification, and hence the importance of the last E-term should increase.

5.2.2. Making LPG return a set of plans

In order to compute a set of k d-distant plans solving a $d$DISTANCE$k$SET problem, we run LPG multiple times, until the problem is solved, with the following two additional changes to the standard version of LPG: (i) the preprocessing phase computing mutex relations and other reachability information exploited during the relaxed plan construction is done only once for all runs; (ii) we maintain an incremental set of valid plans, and we dynamically select one of them as the reference plan p_0 for the next search. Concerning (ii), let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be the set of n valid plans that have been computed so far, and $\mathit{CPlans}(p_i)$ the subset of $\mathcal{P}$ containing all plans that have a distance greater than or equal to d from a reference plan $p_i \in P$.

The reference plan p_0 used in the modified heuristic function E is a plan $p_{max} \in \mathcal{P}$ which has a maximal set of diverse plans in $\mathcal{P}$, i.e.,

$$p_{max} = \underset{p_i \in \mathcal{P}}{\operatorname{argmax}} \left\{ \left| \mathit{CPlans}(p_i) \right| \right\}. \tag{13}$$

The plan p_{max} is incrementally computed each time the local search finds a new solution. In addition to being used to identify the reference plan in E, p_{max} is also used for defining the initial state (partial plan) of the search process. Specifically, we initialize the search using a (partial) plan obtained by randomly removing some actions from a (randomly selected) plan in the set $\mathit{CPlans}(p_{max}) \cup \{p_{max}\}$.

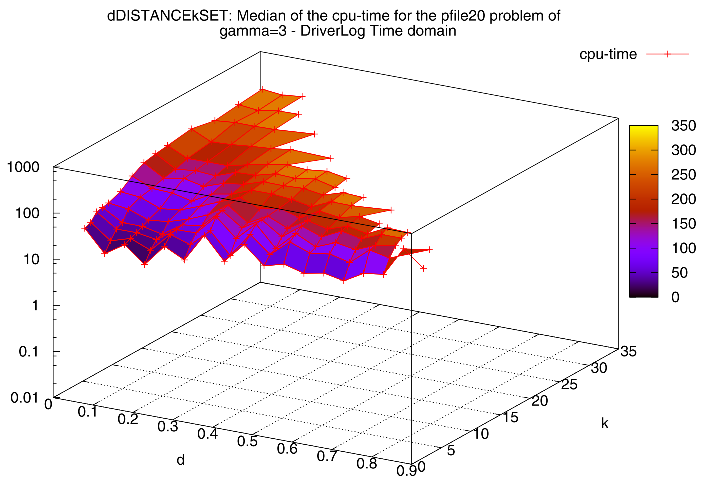
The process of generating diverse plans starting from a dynamically chosen reference plan continues until at least k plans that are all d-distant from each other have been produced. The modified version of LPG to compute diverse plans is called LPG-d.

5.2.3. Experimental analysis with LPG-d

Recall that the distance function δ_a, using set-difference, can be written as the sum of two terms:

$$\delta_a(p_i, p_j) = \frac{|A(p_i) - A(p_j)|}{|A(p_i) \cup A(p_j)|} + \frac{|A(p_j) - A(p_i)|}{|A(p_i) \cup A(p_j)|}. \tag{14}$$

[15] These coefficients are also normalized to a value in $[0, 1]$ using the method described in [27].



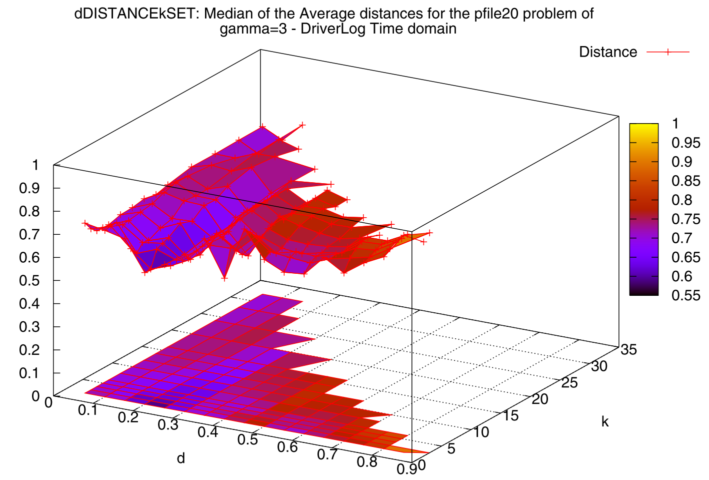


Fig. 8. Performance of LPG-d (CPU time and plan distance) for the problem pfile20 in the DriverLog-Time domain.

The first term represents the contribution of the actions in p_i to the plan difference, while the second term indicates the contribution of p_j to δ_a. We experimentally observed that in some cases the differences between two diverse plans computed using δ_a are mostly concentrated in only one of the δ_a components. This asymmetry means that one of the two plans can have many more actions than the other one, which could imply that the quality of one of the two plans is much worse than the quality of the other plan. In order to avoid this problem, we can parametrize δ_a by imposing the two extra constraints

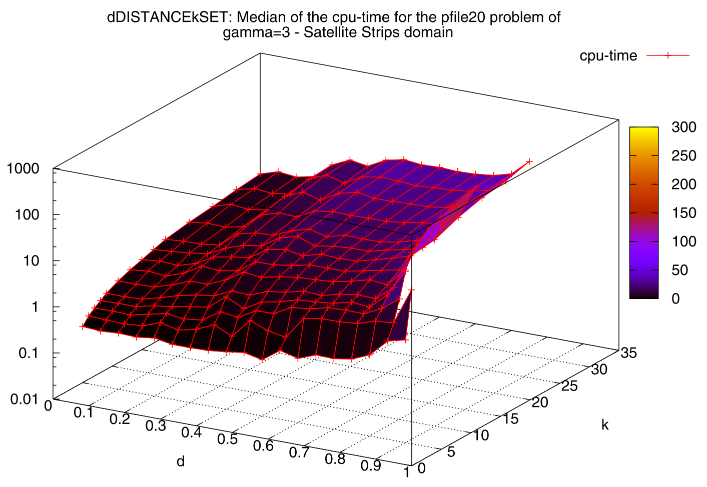
$$\delta_a^A \geqslant d/\gamma \quad \text{and} \quad \delta_a^B \geqslant d/\gamma,$$

where δ_a^A and δ_a^B are the first and second terms of the RHS of Eq. (14), respectively, and γ is an integer parameter "balancing" the diversity of p_i and p_j.

In this section, we analyze the performance of LPG-d in four different benchmark domains: DriverLog, Satellite, Storage and FloorTile from the 3rd, 5th and 7th IPCs.[16] The main goals of the experimental evaluation were (i) showing that LPG-d can efficiently solve a large set of (d, k)-combinations, (ii) investigating the impact of the δ_a γ-constraints on performance, (iii) comparing LPG-d and the standard LPG.

We tested LPG-d using both the default and parametrized versions of δ_a, with $\gamma = 2$ and $\gamma = 3$. We give detailed results for $\gamma = 3$ and a more general evaluation for $\gamma = 2$ and the original δ_a. We consider d that varies from 0.05 to 0.95, using 0.05 increment step, and with $k = 2, \ldots, 5$, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32 (overall, a total of 266 (d, k)-combinations). Since LPG-d is a stochastic planner, we use the median of the CPU times (in seconds) and the median of the average plan distances (over five runs). The average plan distance for a set of k plans solving a specific (d, k)-combination (δ^{av}) is the average of the plans distances between all pairs of plans in the set. The tests were performed on an Intel(R) Xeon(TM) CPU 3.00 GHz, 3 GB RAM. The CPU-time limit was 300 seconds.

[16] We have similar results for other domains: Rovers (IPC3–5), Pathways (IPC5), Logistics (IPC2), ZenoTravel (IPC3).



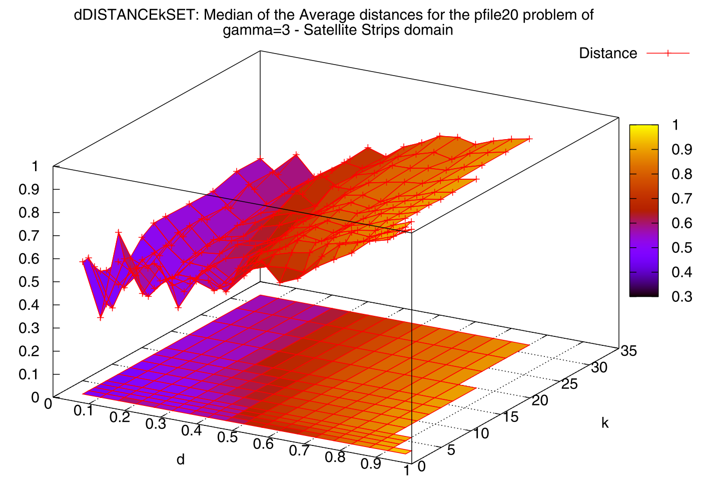


Fig. 9. Performance of LPG-d (CPU time and plan distance) for the problem pfile20 in the Satellite-Strips domain.

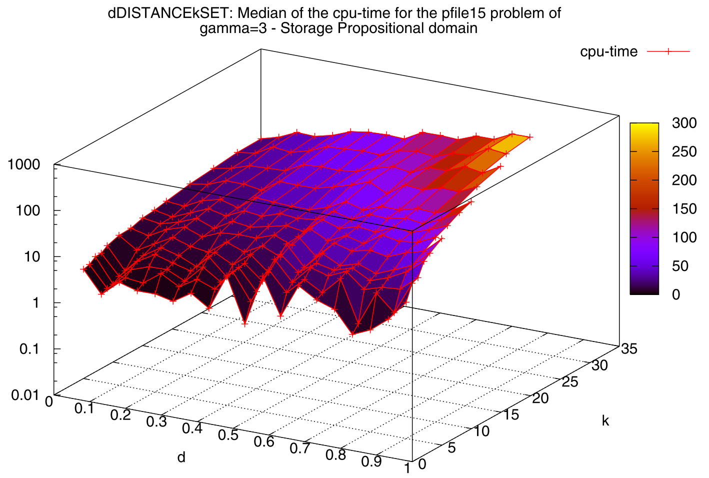
Fig. 8 gives the results for the largest problem in the IPC-3 DriverLog-Time domain (fully-automated track). LPG-d solves 161 (d,k)-combinations, including combinations $d \leqslant 0.4$ and $k = 20$, and $d = 0.95$ and $k = 2$. The average CPU time (top plots) is 151.85 seconds. The average δ^{av} (bottom plots) is 0.73, with δ^{av} always greater than 0.57. With the original δ_a function LPG-d solves 168 (d,k)-combinations, the average CPU time is 149.5 seconds, and the average δ^{av} is 0.73; while with $\gamma = 2$ LPG-d solves 139 combinations, the average CPU time is 144.2 seconds, and the average δ^{av} is 0.72.

Fig. 9 shows the results for the largest problem in the IPC-3 Satellite-Strips domain. LPG-d solves 242 (k,d)-combinations; 153 of them require less than 10 seconds. The average CPU time is 5.46 seconds, and the average δ^{av} is 0.69. We observed similar results when using the original δ_a function or the parametrized δ_a with $\gamma = 2$ (in the second case, LPG-d solves 230 problems, while the average CPU time and the average δ^{av} are nearly the same as with $\gamma = 3$).

Fig. 10 shows the results for a middle-size problem in the IPC-5 Storage-Propositional domain. With $\gamma = 3$ LPG-d solves 252 (k,d)-combinations, 58 of which require less than 10 seconds, while 178 of them require less than 50 seconds. The average CPU time is 25.4 seconds and the average δ^{av} is 0.91. With the original δ_a, LPG-d solves 257 (k,d)-combinations, the average CPU time is 14.5 seconds, and the average δ^{av} is 0.9; with $\gamma = 2$, LPG-d solves 201 combinations, the average CPU time is 31 seconds and the average δ^{av} is 0.93.

Fig. 11 gives the results for the largest problem in the IPC-7 FloorTile-MetricTime domain. LPG-d solves 210 (d,k)-combinations; 171 of them require less than 10 seconds. The average CPU time is 3.6 seconds, and the average δ^{av} is 0.7. We observed similar results when using the original δ_a function or the parametrized δ_a with $\gamma = 2$ (in the second case, LPG-d solves 191 problems, while the average CPU time and the average δ^{av} are nearly the same as with $\gamma = 3$).

The local search in LPG is randomized by a “noise” parameter that is automatically set and updated during search [27]. This randomization is one of the techniques used for escaping local minima, but it also can be useful for computing diverse plans: if we run the search multiple times, each search is likely to consider different portions of the search space, which can lead to different solutions. It is then interesting to compare LPG-d and a method in which we simply run the standard LPG until k d-diverse plans are generated. An experimental comparison of the two approaches show that in many cases



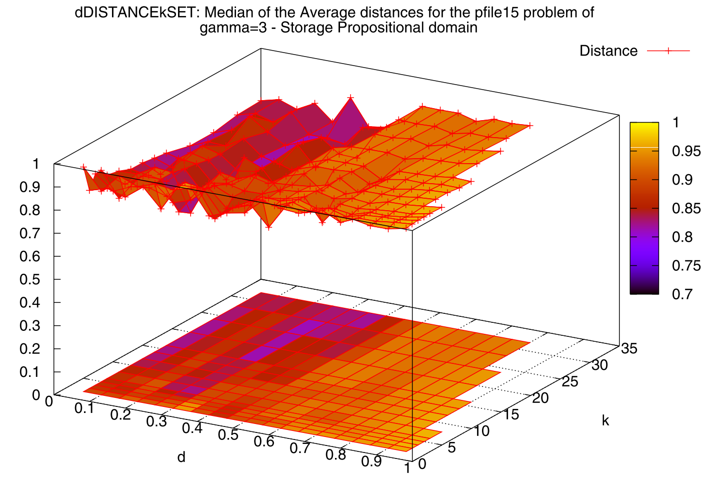


Fig. 10. Performance of LPG-d (CPU time and plan distance) for the problem pfile15 in the Storage-Propositional domain.

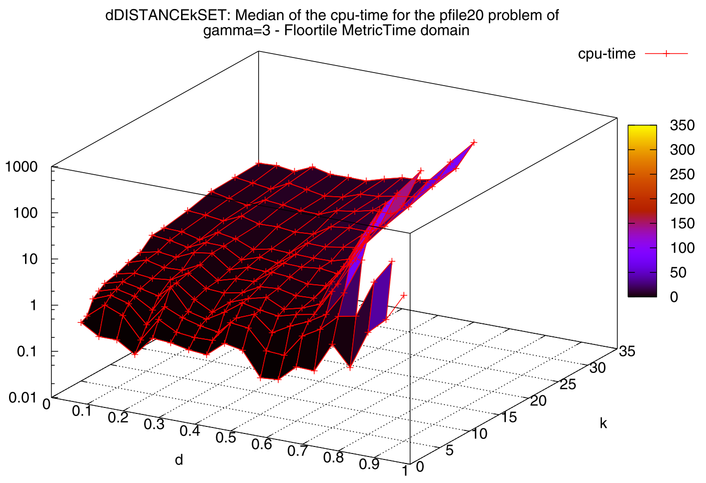
LPG-d performs better. In particular, the new evaluation function E is especially useful for planning problems that are easy to solve for the standard LPG, and that admit many solutions. In these cases, the original E function produces many valid plans with not enough diversification. This problem is significantly alleviated by the new term in E. An example of domain where we observed this behavior is Logistics.[17]

6. Generating plan sets with partial preference knowledge

In this section, we consider the problem of generating plan sets when the user preferences are only partially expressed. In particular, we focus on metric temporal planning where the preference model is assumed to be represented by an incomplete value function specified by a convex combination of two features: *plan makespan* and *execution cost*, with the exact trade-off value w drawn from a given distribution. The quality value of plan sets is measured by the ICP value, as formalized in Eq. (12). Our objective is to find a set of plans $\mathcal{P} \subseteq \mathcal{S}$ where $|\mathcal{P}| \leqslant k$ and $ICP(\mathcal{P})$ is the lowest.

Notice that we restrict the size of the solution set returned, not only for the comprehension issue discussed earlier, but also for an important property of the ICP measure: it is a monotonically non-increasing function of the solution set (specifically, given two solution sets $\mathcal{P}_1$ and $\mathcal{P}_2$ such that the latter is a superset of the former, it is easy to see that $ICP(\mathcal{P}_2) \leqslant ICP(\mathcal{P}_1)$).

[17] For example, LPG-d solved 176 instances for the log_a problem, 47 of them in less than 1 CPU second and 118 of them in less than 10 CPU seconds; the average CPU time was 3.75 seconds and the average δ^{av} was 0.47. While using the standard LPG, only 107 instances were solved, 27 of them in less than 1 CPU seconds and 73 of them in less than 10 CPU seconds; the average CPU time was 5.14 seconds and the average δ^{av} was 0.33.



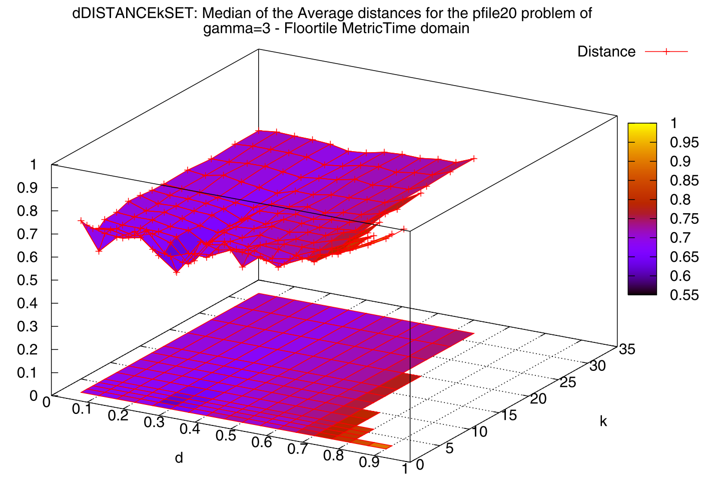


Fig. 11. Performance of LPG-d (CPU time and plan distance) for the problem pfile20 in the FloorTile-MetricTime domain.

6.1. Sampling weight values

Given that the distribution of trade-off value w is known, the straightforward way to find a set of representative solutions is to first sample a set of k values for w: $\{w_1, w_2, \dots, w_k\}$ based on the distribution $h(w)$. For each value w_i, we can find an (optimal) plan p_i minimizing the value of the overall value function $V(p, w_i) = w_i \times t_p + (1 - w_i) \times c_p$. The final set of solutions $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$ is then filtered to remove duplicates and dominated solutions, thus selecting the plans making up the lower convex hull. The final set can then be returned to the user. While intuitive and easy to implement, this sampling-based approach has several potential flaws that can limit the quality of its resulting plan set.

First, given that k solution plans are searched sequentially and independently of each other, even if the plan p_i found for each w_i is optimal, the final solution set $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$ may not even be the optimal set of k solutions with regard to the ICP measure. More specifically, for a given set of solutions $\mathcal{P}$, some trade-off value w, and two non-dominated plans p, q such that $V(p, w) < V(q, w)$, it is possible that $ICP(\mathcal{P} \cup \{p\}) > ICP(\mathcal{P} \cup \{q\})$. In our running example (Fig. 5), let $\mathcal{P} = \{p_2, p_5\}$ and $w = 0.8$ then $V(p_1, w) = 0.8 \times 4 + 0.2 \times 25 = 8.2 < V(p_7, w) = 0.8 \times 12 + 0.2 \times 5 = 10.6$. Thus, the planner will select p_1 to add to $\mathcal{P}$ because it looks locally better given the weight $w = 0.8$. However, $ICP(\{p_1, p_2, p_5\}) \approx 10.05 > ICP(\{p_2, p_5, p_7\}) \approx 7.71$ so indeed by taking previous set into consideration then p_7 is a much better choice than p_1.

Second, the values of the trade-off parameter w are sampled based on a given distribution, and independently of the particular planning problem being solved. As there is no relation between the sampled w values and the solution space of a given planning problem, sampling approach may return very few distinct solutions even if we sample a large number of weight values w. In our example, if all w samples have values $w \leqslant 0.67$ then the optimal solution returned for any of them will always be p_7. However, we know that $\mathcal{P}^* = \{p_1, p_3, p_7\}$ is the optimal set according to the ICP measure. Indeed, if $w \leqslant 0.769$ then the sampling approach can only find the set $\{p_7\}$ or $\{p_3, p_7\}$ and still not be able to find the optimal set $\mathcal{P}^*$.

Algorithm 1: Incrementally find solution set $\mathcal{P}$.

```
1  Input: A planning problem with a solution space $\mathcal{S}$; maximum number of plans required $k$; number of sampled trade-off values $k_0$ $(0 < k_0 < k)$;
   time bound $t$;
2  Output: A plan set $\mathcal{P}$ $(|\mathcal{P}| \leqslant k)$;
3  begin
4      $W \leftarrow$ sample $k_0$ values for $w$;
5      $\mathcal{P} \leftarrow$ find good quality plans in $\mathcal{S}$ for each $w \in W$;
6      while $|\mathcal{P}| < k$ and $search\_time < t$ do
7          Search for $p$ s.t. $ICP(\mathcal{P} \cup \{p\}) < ICP(\mathcal{P})$
8          $\mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$
9      end
10     Return $\mathcal{P}$
11 end
```

6.2. ICP sequential approach

Given the potential drawbacks of the sampling approach outlined above, we also pursued an alternative approach that takes into account the ICP measure more actively. Specifically, we incrementally build the solution set $\mathcal{P}$ by finding a solution p such that $\mathcal{P} \cup \{p\}$ has the lowest ICP value. We can start with an empty solution set $\mathcal{P} = \emptyset$, then at each step try to find a new plan p such that $\mathcal{P} \cup \{p\}$ has the lowest ICP value.

While this approach directly takes the ICP measure into consideration at each step of finding a new plan and avoids the drawbacks of the sampling-based approach, it also has its own share of potential flaws. Given that the set is built incrementally, the earlier steps where the first "seed" solutions are found are very important. The closer the seed solutions are to the global lower convex hull, the better the improvement in the ICP value. In our example (Fig. 5), if the first plan found is p_2 then the subsequent plans found to best extend $\{p_2\}$ can be p_5 and thus the final set does not come close to the optimal set $\mathcal{P}^* = \{p_1, p_3, p_7\}$.

6.3. Hybrid approach

In this approach, we aim to combine the strengths of both the sampling and ICP-sequential approaches. Specifically, we use sampling to find several plans optimizing for different weights. The plans are then used to seed the subsequent ICP-sequential runs. By seeding the hybrid approach with good quality plan set scattered across the Pareto optimal set, we hope to gradually expand the initial set to a final set with a much better overall ICP value. Algorithm 1 shows the pseudo-code for the hybrid approach. We first independently sample the set of k_0 values (with k_0 pre-determined) of w given the distribution on w (step 4). We then run a heuristic planner multiple times to find an optimal (or good quality) solution for each trade-off value w (step 5). We then collect the plans found and seed the subsequent runs when we incrementally update the initial plan set with plans that lower the overall ICP value (steps 6–8). The algorithm terminates and returns the latest plan set (step 9) if k plans are found or the time bound exceeds.

6.4. Making LPG search sensitive to ICP

We use a modified version of the Metric-LPG planner [28] in implementing our algorithms, introducing the *totalcost* numerical fluent into the domain to represent the plan cost that we are interested in.[18] Not only is Metric-LPG equipped with a very flexible local-search framework that has been extended to handle various objective functions, but it can also be made to search for single or multiple solutions. Specifically, for the sampling-based approach, we first sample the w values based on a given distribution. For each w value, we set the metric function in the domain file to: $w \times makespan + (1 - w) \times totalcost$, and run the original LPG in the quality mode to heuristically find the best solution within the time limit for that metric function. The final solution set is filtered to remove any duplicate solutions, and returned to the user.

For the ICP-sequential and hybrid approach, we cannot use the original LPG implementation as is and need to modify the neighborhood evaluation function in LPG to take into account the ICP measure and the current plan set $\mathcal{P}$. For the rest of this section, we will explain this procedure in detail.

Background: Metric-LPG uses local search to find plans within the space of *numerical action graphs* (NA-graph). This leveled graph consists of a sequence of interleaved proposition and action layers. The proposition layers consist of a set of propositional and numerical nodes, while each action layer consists of at most one action node, and a number of no-op links. An NA-graph G represents a valid plan if all actions' preconditions are supported by some actions appearing in the earlier level in G. The search neighborhood for each local-search step is defined by a set of graph modifications to fix some remaining

[18] Although we are interested in the plan cost as summation of action costs, our implementation can also be extended for planning problems where plan cost is an expression involving numerical fluents.

inconsistencies (unsupported preconditions) p at a particular level l. This can be done by either inserting a new action a supporting p or removing from the graph the action a that p is a precondition of (which can introduce new inconsistencies).

Each local move creates a new NA-graph G', which is evaluated as a weighted combination of two factors: $\mathit{SearchCost}(G')$ and $\mathit{ExecCost}(G')$. Here, $\mathit{SearchCost}(G')$ is the amount of search effort to resolve inconsistencies newly introduced by inserting or removing action a; it is measured by the number of actions in a relaxed plan R resolving all such inconsistencies. The total cost $\mathit{ExecCost}(G')$, which is a default function to measure plan quality, is measured by the total *action execution costs* of all actions in R. The two weight adjustment values α and β are used to steer the search toward either finding a solution quickly (higher α value) or better solution quality (higher β value). Metric-LPG then selects the local move leading to the smallest $E(G')$ value.

Adjusting the evaluation function $E(G')$ for finding set of plans with low ICP measure: To guide Metric-LPG towards optimizing our ICP-sensitive objective function instead of the original minimizing cost objective function, we need to replace the default plan quality measure $\mathit{ExecCost}(G')$ with a new measure $\mathit{ICPEst}(G')$. Specifically, we adjust the function for evaluating each new NA-graph generated by local moves at each step to be a combination of $\mathit{SearchCost}(G')$ and $\mathit{ICPEst}(G')$. Given the set of found plans $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, $\mathit{ICPEst}(G')$ guides Metric-LPG's search toward a plan p generated from G' such that the resulting set $\mathcal{P} \cup \{p\}$ has a minimum ICP value: $p = \operatorname{argmin}_p \mathit{ICP}(\mathcal{P} \cup \{p\})$. Thus, $\mathit{ICPEst}(G')$ estimates the expected total ICP value if the best plan p found by expanding G' is added to the current found plan set $\mathcal{P}$. Like the original Metric-LPG, p is estimated by $p_R = G' \cup R$ where R is the relaxed plan resolving inconsistencies in G' caused by inserting or removing a. The $\mathit{ICPEst}(G')$ for a given NA-graph G' is calculated as: $\mathit{ICPEst}(G') = \mathit{ICP}(\mathcal{P} \cup p_R)$ with the ICP measure as described in Eq. (12). Notice here that while $\mathcal{P}$ is the set of valid plans, p_R is not. It is an invalid plan represented by a NA-graph containing some unsupported preconditions. However, Eq. (12) is still applicable as long as we can measure the time and cost dimensions of p_R. To measure the makespan of p_R, we estimate the time points at which unsupported facts in G' would be supported in $p_R = G' \cup R$ and propagate them over actions in G' to its last level. We then take the earliest time point at which all facts at the last level appear to measure the makespan of p_R. For the cost measure, we just sum the individual costs of all actions in p_R.

At each step of Metric-LPG's local search framework, combining two measures $\mathit{ICPEst}(G')$ and $\mathit{SearchCost}(G')$ gives us an evaluation function that fits right into the original Metric-LPG framework and prefers a NA-graph G' in the neighborhood of G that gives the best trade-off between the estimated effort to repair and the estimated decrease in quality of the next resulting plan set.

6.5. Experimental results

We have implemented several approaches based on our algorithms discussed in the previous sections: Sampling (Section 6.1), ICP-sequential (Section 6.2) and Hybrid that combines both (Section 6.3) with both the uniform and triangular distributions. We consider two types of distributions in which the most probable weight for plan makespan are 0.2 and 0.8, which we will call “w02” and “w08” distributions respectively (Fig. 12 shows these distributions). We test all implementations against a set of 20 problems in each of several benchmark temporal planning domains used in the previous International Planning Competitions (IPC): ZenoTravel, DriverLog, and Depots. The only modification to the original benchmark set is the added action costs. The descriptions of these domains can be found at the IPC website (ipc.icaps-conference.org). The experiments were conducted on an Intel Core2 Duo machine with 3.16 GHz CPU and 4 GB RAM. For all approaches, we search for a maximum of $k = 10$ plans within the 10-minute time limit for each problem (i.e., $t = 10$ minutes), and the resulting plan set is used to compute the ICP value. In the Sampling approach, we generate ten trade-off values w between *makespan* and *plan cost* based on the distribution, and for each one we search for a plan p subject to the value function $V(p, w) = w \times t_p + (1 - w) \times c_p$. In the Hybrid approach, on the other hand, the first Sampling approach is used with $k_0 = 3$ generated trade-off values to find an initial plan set, which is then improved by the ICP-sequential runs. As Metric-LPG is a stochastic local search planner, we run it three times for each problem and average the results. In 77% and 70% of 60 problems in the three tested domains for the Hybrid and Sampling approaches respectively, the standard deviation of ICP values of plan sets are at most 5% of the average values. This indicates that ICP values of plan set in different runs are quite stable. As the Hybrid approach is an improved version of ICP-sequential and gives better results in almost all tested problems, we omit the ICP-sequential in discussions below. We now analyze the results in more detail.

The utility of using the partial knowledge of user's preferences: To evaluate the utility of taking partial knowledge of user preferences into account, we first compare our results against the naive approaches that generate a plan set without explicitly taking into account partial knowledge. Specifically, we run the default LPG planner with different random seeds to find multiple non-dominated plans. The LPG planner was run with both *speed* setting, which finds plans quickly, and *diverse* setting, which takes longer time to find better set of diverse plans. Fig. 13 shows the comparison between quality of plan sets returned by Sampling and those naive approaches when the distribution of the trade-off value w between *makespan* and *plan cost* is assumed to be uniform. Overall, among 20 tested problems for each of the ZenoTravel, DriverLog, and Depots domains, the Sampling approach is better than LPG-speed in 19/20, 20/20 and 20/20 and is better than LPG-d in 18/20, 18/20, and 20/20 problems respectively. We observed similar results comparing Hybrid and those two approaches: in particular, the Hybrid approach is better than LPG-speed in all 60 problems and better than LPG-d in 19/20, 18/20, and

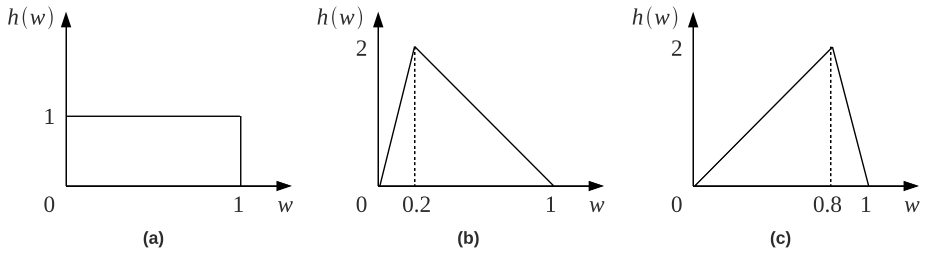


Fig. 12. The distributions: (a) uniform, (b) w02, (c) w08 (see text).

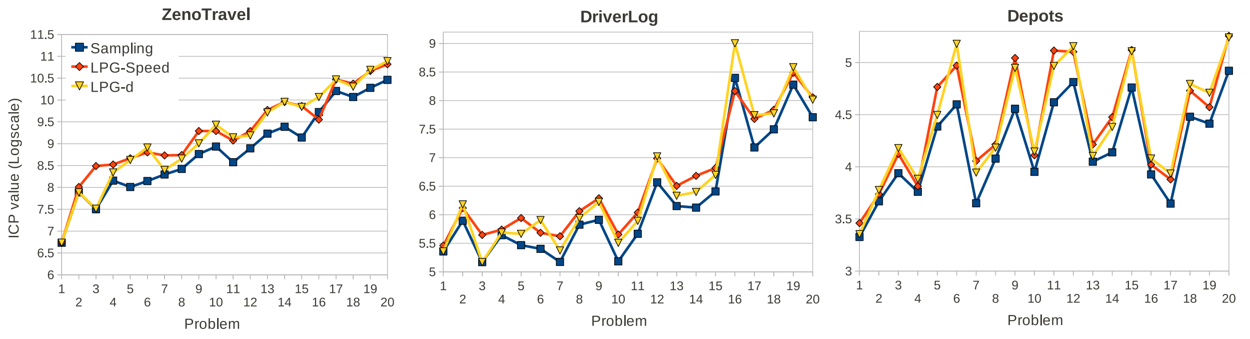


Fig. 13. Results for the ZenoTravel, DriverLog and Depots domains comparing the Sampling and baseline LPG approaches on the overall ICP value (log scale) with the uniform distribution.

20/20 problems respectively. These results support our intuition that taking into account the partial knowledge about user preferences (if it is available) increases the quality of the plan set.

Comparing the Sampling and Hybrid approaches: We now compare the effectiveness of the Sampling and Hybrid approaches in terms of the quality of returned plan sets with the uniform, w02 and w08 distributions.

ICP value: We first compare the two approaches in terms of the ICP values of plan sets returned indicating their quality evaluated by the user. Tables 6, 7, and 8 show the results in the three domains. In general, Hybrid tends to be better than Sampling in this criterion for most of the domains and distributions. In particular, in the ZenoTravel domain it returns higher quality plan sets in 15/20 problems when the distribution is uniform, 10/20 and 13/20 problems when it is w02 and w08 respectively (both approaches return plan sets with equal ICP values for two problems with the w02 distribution and one problem with the w08 distribution). In the DriverLog domain, Hybrid returns better plan sets for 11/20 problems with the uniform distribution (and for other three problems the plan sets have equal ICP values), but worse with the triangular distributions: 8/20 (another 2 equals) and 9/20 (another one equals) with w02 and w08. The improvement on the quality of plan sets that Hybrid contributes is more significant in the Depots domain: it is better than Sampling in 11/20 problems with the uniform distribution (and equal in 3 problems), in 12/20 problems with the w02 and w08 distributions (with w02 both approaches return plan sets with equal ICP values for 4 problems, and for 2 problems when it is w08).

In many large problems of the ZenoTravel and DriverLog domains where Sampling performs better than Hybrid, we notice that the first phase of the Hybrid approach that searches for the first 3 initial plans normally takes most of the allocated time, and therefore there is not much time left for the second phase to improve the quality of the plan set. We also observe that among the three settings of the trade-off distributions, the positive effect of the second phase in Hybrid approach (which is to improve the quality of the initial plan sets) tends to be more stable across different domains with uniform distribution, but less with the triangular, in particular Sampling beats Hybrid in the DriverLog domain when the distribution is w02. Perhaps this is because with the triangular distributions, the chance that LPG planner (that is used in our Sampling approach) returns the same plans even with different trade-off values would increase, especially when the most probable value of makespan happens to be in a (wide) range of weights in which one single plan is optimal. This result agrees with the intuition that when the knowledge about user preferences is *almost* complete (i.e., the distribution of trade-off value is "peak"), then the Sampling approach with smaller number of generated weight values may be good enough (assuming that a good planner optimizing a complete value function is available).

Since the quality of a plan set depends on how the two features makespan and plan cost are optimized, and how the plans "span" the space of time and cost, we also compare the Sampling and Hybrid approaches in terms of those two criteria. In particular, we compare plan sets returned by the two approaches in terms of (i) their *median* values of makespan and cost, which represent how "close" the plan sets are to the origin of the space of makespan and cost, and (ii) their *standard deviation* of makespan and cost values, which indicate how the sets span each feature axis.

Table 6
The ICP value of plan sets in the ZenoTravel domain returned by the Sampling and Hybrid approaches with the distributions (a) uniform, (b) w02 and (c) w08. The problems where Hybrid returns plan sets with better quality than Sampling are marked with *.

Prob	Sampling	Hybrid	Prob	Sampling	Hybrid	Prob	Sampling	Hybrid
1*	840.00	839.98	1	972.00	972.00	1	708.00	708.00
2*	2661.43	2661.25	2	3067.20	3067.20	2*	2255.792	2255.788
3*	1807.84	1805.95	3*	2083.91	2083.83	3*	1535.54	1535.32
4*	3481.31	3477.49	4*	4052.75	4026.92	4*	2960.84	2947.66
5*	3007.97	2743.85	5*	3171.86	3171.73	5*	2782.16	2326.94
6*	3447.37	2755.25	6*	4288.00	3188.61	6*	2802.00	2524.18
7*	4006.38	3793.44	7*	4644.40	4377.40	7*	3546.95	3235.63
8*	4549.90	4344.70	8*	5060.81	5044.43	8*	3802.60	3733.90
9*	6397.32	5875.13	9*	7037.87	6614.30	9*	5469.24	5040.88
10*	7592.72	6826.60	10*	9064.40	7472.37	10*	6142.68	5997.45
11*	5307.04	5050.07	11*	5946.68	5891.76	11*	4578.09	4408.36
12*	7288.54	6807.28	12*	7954.74	7586.28	12	5483.19	5756.89
13*	10,208.11	9956.94	13*	11,847.13	11,414.88	13*	8515.74	8479.09
14	11,939.22	13,730.87	14	14,474.00	15,739.19	14*	11,610.38	11,369.46
15	9334.68	13,541.28	15	16,125.70	16,147.28	15*	11,748.45	11,418.59
16*	16,724.21	13,949.26	16	19,386.00	19,841.67	16	14,503.79	15,121.77
17*	27,085.57	26,822.37	17	29,559.03	32,175.66	17	21,354.78	22,297.65
18	23,610.71	25,089.40	18	28,520.17	29,020.15	18	20,107.03	21,727.75
19	29,114.30	29,276.09	19	34,224.02	36,496.40	19	23,721.90	25,222.24
20	34,939.27	37,166.29	20	39,443.66	42,790.97	20	28,178.45	28,961.51
	(a)			(b)			(c)	

Table 7
The ICP value of plan sets in the DriverLog domain returned by the Sampling and Hybrid approaches with the distributions (a) uniform, (b) w02 and (c) w08. The problems where Hybrid returns plan sets with better quality than Sampling are marked with *.

Prob	Sampling	Hybrid	Prob	Sampling	Hybrid	Prob	Sampling	Hybrid
1	212.00	212.00	1	235.99	236.00	1	188.00	188.00
2*	363.30	348.38	2*	450.07	398.46	2*	333.20	299.70
3	176.00	176.00	3	203.20	203.20	3	148.80	148.80
4*	282.00	278.45	4*	336.01	323.79	4*	238.20	233.20
5*	236.83	236.33	5	273.80	288.51	5*	200.80	199.52
6*	222.00	221.00	6	254.80	254.80	6*	187.47	187.20
7	176.50	176.50	7*	226.20	203.80	7	149.20	149.20
8*	338.96	319.43	8	387.53	397.75	8	300.54	323.87
9*	369.18	301.72	9*	420.64	339.05	9*	316.80	263.92
10*	178.38	170.55	10*	196.44	195.11	10*	158.18	146.12
11*	289.04	232.65	11*	334.13	253.09	11*	245.38	211.60
12	711.48	727.65	12*	824.17	809.93	12*	605.86	588.82
13*	469.50	460.99	13	519.92	521.05	13	388.80	397.67
14	457.04	512.11	14	524.56	565.94	14	409.02	410.53
15*	606.81	591.41	15*	699.49	643.72	15	552.79	574.95
16	4432.21	4490.17	16	4902.34	6328.07	16	3580.32	4297.47
17	1310.83	1427.70	17	1632.86	1659.46	17	1062.03	1146.68
18*	1800.49	1768.17	18	1992.32	2183.13	18	1448.36	1549.09
19	3941.08	4278.67	19	4614.13	7978.00	19*	3865.54	2712.08
20	2225.66	2397.61	20	2664.00	2792.90	20	1892.28	1934.11
	(a)			(b)			(c)	

Table 9 summarizes for each domain, distribution and feature the number of problems in which each approach (either Sampling or Hybrid) generates plan sets with better median of each feature value (makespan and plan cost) than the other. There are 60 problems across 3 different distributions, so in total, 180 cases for each feature. Sampling and Hybrid return plan sets with better makespan in 40 and 62 cases, and with better plan cost in 52 and 51 cases (respectively), which indicates that Hybrid is slightly better than Sampling on optimizing makespan but is possibly worse on optimizing plan cost. In ZenoTravel domain, for all distributions Hybrid likely returns better plan sets on the makespan than Sampling, and Sampling is better on the plan cost feature. In the DriverLog domain, Sampling is better on the makespan feature with both non-uniform distributions, but worse than Hybrid with the uniform. On the plan cost feature, Hybrid returns plan sets with better median than Sampling on the uniform and w02 distributions, and both approaches perform equally well with the w08 distribution. In the Depots domain, Sampling is better than Hybrid on both features with the uniform distribution, and only better than Hybrid on the makespan with the distribution w08.

In terms of spanning plan sets, Hybrid performs much better than Sampling on both features across three domains, as shown in Table 10. In particular, over 360 cases for both makespan and plan cost features, there are only 10 cases where Sampling produces plan sets with better standard deviation than Hybrid on each feature. Hybrid, on the other hand, generates plan sets with better standard deviation on makespan in 91 cases, and in 85 cases on the plan cost.

Table 8
The ICP value of plan sets in the Depots domain returned by the Sampling and Hybrid approaches with the distributions (a) uniform, (b) w02 and (c) w08. The problems where Hybrid returns plan sets with better quality than Sampling are marked with *.

Prob	Sampling	Hybrid	Prob	Sampling	Hybrid	Prob	Sampling	Hybrid
1	27.87	27.87	1	28.56	28.56	1*	28.50	27.85
2	39.22	39.22	2	41.12	41.12	2	38.26	38.26
3*	51.36	50.43	3*	54.44	52.82	3*	49.49	48.58
4	43.00	43.00	4	46.00	46.00	4*	40.87	40.00
5	80.36	81.01	5	82.93	84.45	5	75.96	78.99
6	99.40	111.11	6	102.58	110.98	6	94.79	98.40
7*	38.50	38.49	7*	40.53	40.40	7*	37.04	36.60
8*	59.08	58.41	8*	62.15	62.08	8*	55.89	54.67
9	95.29	103.85	9	100.59	106.00	9	87.93	95.05
10*	52.04	50.00	10	52.40	52.40	10*	47.86	47.60
11	101.43	107.66	11*	110.18	108.07	11	97.56	99.06
12	123.09	129.34	12*	144.67	135.80	12	124.58	128.01
13*	57.37	57.22	13*	60.83	60.72	13	54.66	54.66
14*	62.75	62.33	14*	70.32	69.87	14*	65.20	62.02
15	116.82	117.86	15	113.15	124.28	15	101.09	124.43
16*	50.77	49.36	16*	54.98	54.12	16*	47.04	46.35
17*	38.38	37.77	17*	42.86	41.50	17*	37.56	36.92
18*	88.28	85.55	18*	94.53	90.02	18*	76.73	75.29
19*	82.60	82.08	19*	94.21	89.28	19*	74.73	72.45
20*	137.13	133.47	20*	150.80	135.93	20*	122.43	120.31
	(a)			(b)			(c)	

Table 9
Number of problems for each domain, distribution and feature where Sampling (Hybrid) returns plan sets with better (i.e., smaller) *median* of feature value than that of Hybrid (Sampling), denoted in the table by $S > H$ ($H > S$, respectively). We mark bold the numbers of problems that indicate the outperformance of the corresponding approach.

Domain	Distribution	Median of makespan		Median of cost	
		$S > H$	$H > S$	$S > H$	$H > S$
ZenoTravel	uniform	3	**17**	**16**	4
	w02	6	**12**	**14**	4
	w08	6	**13**	**13**	6
DriverLog	uniform	6	**11**	7	**11**
	w02	**10**	8	8	**10**
	w08	**10**	7	9	9
Depots	uniform	**9**	8	**9**	7
	w02	7	**9**	5	**9**
	w08	**11**	7	7	**11**

Table 10
Number of problems for each domain, distribution and feature where Sampling (Hybrid) returns plan sets with better (i.e., larger) *standard deviation* of feature value than that of Hybrid (Sampling), denoted in the table by $S > H$ ($H > S$, respectively). We mark bold the numbers of problems that indicate the outperformance of the corresponding approach.

Domain	Distribution	SD of makespan		SD of cost	
		$S > H$	$H > S$	$S > H$	$H > S$
ZenoTravel	uniform	8	**12**	6	**14**
	w02	4	**14**	7	**11**
	w08	6	**13**	8	**11**
DriverLog	uniform	5	**11**	6	**10**
	w02	7	**10**	7	**9**
	w08	8	**9**	**10**	7
Depots	uniform	**10**	7	7	**9**
	w02	7	**9**	5	**10**
	w08	5	**13**	7	**11**

These experimental results support our arguments in Section 6.1 about the limits of sampling idea. Since one single plan could be optimal for a wide range of weight values, the search in the Sampling approach with different trade-off values may focus on looking for plans only at the same region of the feature space (specified by the particular value of the weight), which can reduce the chance of having plans with better value on some particular feature. On the opposite side, the Hybrid

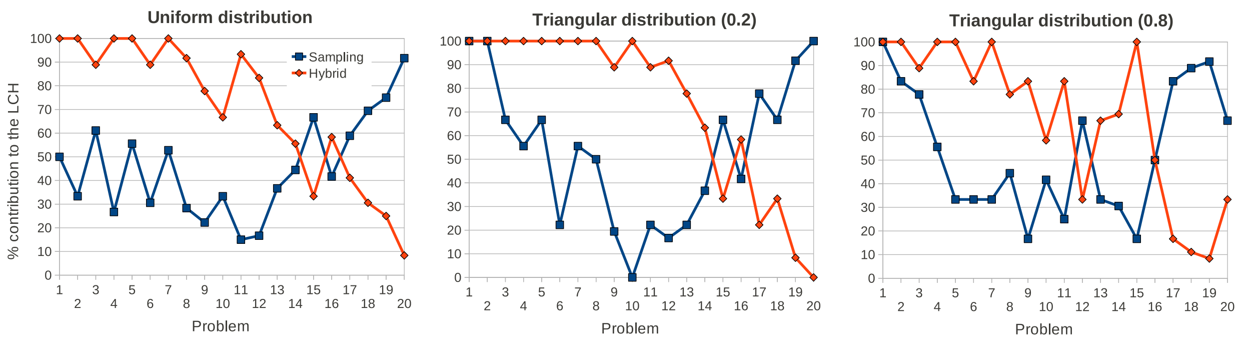


Fig. 14. The contribution into the common lower convex hull of plan sets in the ZenoTravel domain with different distributions.

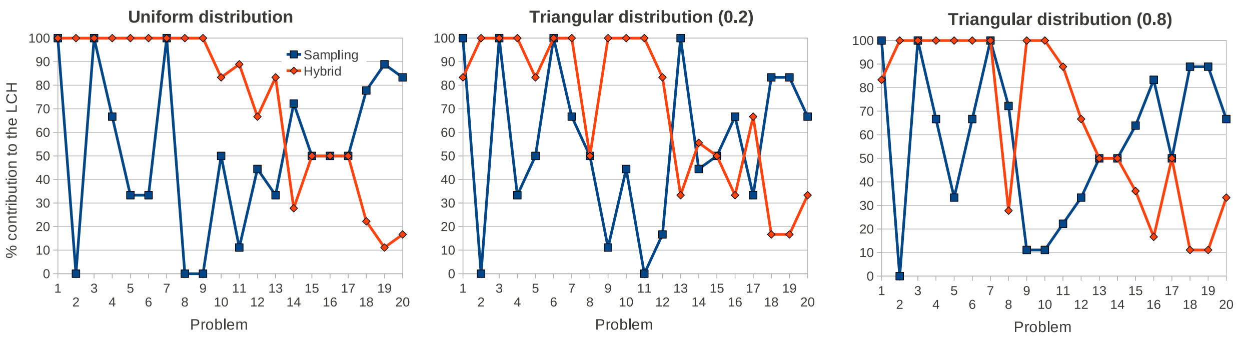


Fig. 15. The contribution into the common lower convex hull of plan sets in the DriverLog domain with different distributions.

approach tends to be better in spanning plan sets to a larger region of the space, as the set of plans that have been found is taken into account during the search.

Contribution to the lower convex hull: The comparison above between Sampling and Hybrid considers the two features separately. We now examine the relation between plan sets returned by those approaches on the joint space of both features, in particular taking into account the dominance relation between plans in the two sets. In other words, we compare the relative total number of plans in the lower convex hull (LCH) found by each approach. Given that this is the set that should be returned to the user (to select one from), the higher number tends to give her a better expected utility value. To measure the relative performance of both approaches with respect to this criterion, we first create a set S combining the plans returned by them. We then compute the set $S_{lch} \subseteq S$ of plans in the lower convex hull among all plans in S. Finally, we measure the percentages of plans in S_{lch} that are actually returned by each of our tested approaches. Figs. 14, 15 and 16 show the contribution to the LCH of plan sets returned by Sampling and Hybrid in the ZenoTravel, DriverLog and Depots domains.

In general, we observe that the plan set returned by Hybrid contributes more into the LCH than that of Sampling for most of the problems (except for some large problems) with most of the distributions and domains. Specifically, in the ZenoTravel domain, Hybrid contributes more plans to the LCH than Sampling in 15/20, 13/20 (and another 2 equals), 13/20 (another 2 equals) problems for the uniform, w02 and w08 distributions respectively. In the DriverLog domain, it is better than Sampling in 10/20 (another 6 equals), 10/20 (another 4 equals), 8/20 (another 5 equals) problems; and Hybrid is better in 11/20 (another 6 equals), 11/20 (another 4 equals) and 11/20 (another 4 equals) for the uniform, w02 and w08 distributions in the Depots domain. Again, similar to the ICP value, the Hybrid approach is less effective on problems with large size (except with the w08 distribution in the Depots domain) in which the search time is mostly used for finding initial plan sets. We also note that a plan set with higher contribution to the LCH is *not* guaranteed to have better quality, except for the extreme case where one plan set contributes 100% and completely dominates the other which contributes 0% to the LCH. For example, consider problem 14 in the ZenoTravel domain: even though the plan sets returned by Hybrid contribute more than those of Sampling in all three distributions, it is only the w08 where it has a better ICP value. The reason for this is that the ICP value depends also on the range of the trade-off value (and its density) for which a plan in the LCH is optimal, whereas the LCH is constructed by simply comparing plans in terms of their makespan and cost separately (i.e., using the dominance relation), ignoring their relative importance.

The sensitivity of plan sets to the distributions: All analysis having been done so far is to compare the effectiveness of approaches with respect to a particular distribution of the trade-off value. In this part, we examine how sensitive the plan sets are with respect to different distributions.

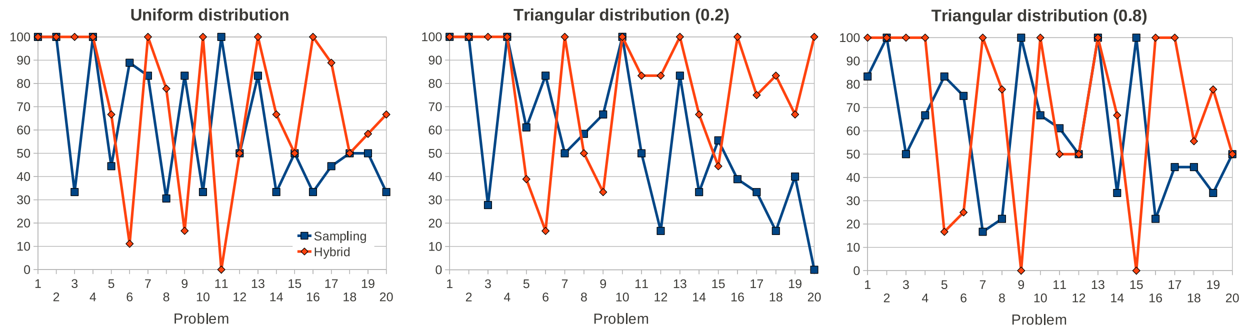


Fig. 16. The contribution into the common lower convex hull of plan sets in the Depots domain with different distributions.

Table 11
Number of problems for each approach, domain and feature where the plan sets returned with the w02 (w08) distribution with better (i.e., smaller) *median* of feature value than that with w08 (w02), denoted in the table by w02 > w08 (w08 > w02, respectively). For each approach, we mark bold the numbers for domains in which there are more problems whose plan sets returned with w08 (w02) have better makespan (plan cost) median than those with w02 (w08, respectively).

Approach	Domain	Median of makespan		Median of cost	
		w02 > w08	w08 > w02	w02 > w08	w08 > w02
Sampling	ZenoTravel	**5**	**13**	**11**	**8**
	DriverLog	**6**	**10**	**13**	**5**
	Depots	**6**	**12**	**10**	**7**
Hybrid	ZenoTravel	**5**	**10**	**10**	**4**
	DriverLog	**4**	**10**	6	9
	Depots	**8**	**10**	4	11

Optimizing high-priority feature: We first consider how plan sets are optimized on each feature (makespan and plan cost) by each approach with respect to two non-uniform distributions w02 and w08. Those are the distributions representing scenarios where the users have different priority on the features, and plan sets should be biased to optimizing the feature that has higher priority (i.e., larger value of weight). In particular, plans generated using the w08 distribution should have better (i.e., *smaller*) makespan values than those found with the w02 distribution (since in the makespan has higher priority in w08 than it is in w02); on the other hand, plan set returned with w02 should have better values of plan cost than those with w08.

Table 11 summarizes for each domain, approach and feature, the number of problems in which plan sets returned with one distribution (either w02 or w08) have better *median* value than with the other. We observe that for both features, the Sampling approach is very likely to “push” plan sets to regions of the space of makespan and cost with better value of more interested feature. On the other hand, the Hybrid approach tends to be more sensitive to the distributions on both the features in the ZenoTravel domain, and is more sensitive only on the makespan feature in the DriverLog and Depots domains. Those results generally show that our approaches can bias the search towards optimizing features that are more desired by the user.

Spanning plan sets on individual features: Next, we examine how plan sets span each feature, depending on the degree of incompleteness of the distributions. Specifically, we compare the *standard deviation* of plan sets returned using the uniform distribution with those generated using the w02 and w08 distributions. Intuitively, we expect that the plan sets returned with the uniform distribution will have higher standard deviation than those with the distributions w02 and w08.

Table 12 shows for each approach, domain and feature, the number of problems generated with the uniform distribution that have better standard deviation on the feature than those found with the distribution w02. We observe that with the makespan feature, both approaches return plan sets that are more “spanned” on makespan in the Depots domain, but not with ZenoTravel and DriverLog. With the plan cost feature, Hybrid shows its positive impact on all three domains, whereas Sampling shows it with the ZenoTravel and Depots domains. Similarly, Table 13 shows the results comparing the uniform and w08 distributions. This time, Sampling returns plan sets with better standard deviation on both features in the ZenoTravel and Depots domains, but not in DriverLog. Hybrid also shows this in the ZenoTravel domain, but for the remaining two domains, it tends to return plan sets with expected standard deviation on the plan cost feature only. From all of these results, we observe that with the uniform distribution, both approaches likely generate plan sets that span better than with non-uniform distributions, especially on the plan cost feature.

In summary, the experimental results in this section support the following hypotheses:

- Instead of ignoring the user preferences which are partially specified, one should take them into account while synthesizing plans, as plan sets returned would have better quality.

Table 12
Number of problems for each approach, domain and feature where the plan sets returned with the uniform (w02) distribution have better (i.e., higher) *standard deviation* of the feature value than that with w02 (uniform), denoted in the table by $U >$ w02 (w02 $> U$, respectively). For each approach and feature, we mark bold the numbers for domains in which there are more problems whose plan sets returned with the uniform distribution have better standard deviation value of the feature than those with the w02 distribution.

Approach	Domain	SD of makespan		SD of cost	
		$U >$ w02	w02 $> U$	$U >$ w02	w02 $> U$
Sampling	ZenoTravel	9	10	**10**	**7**
	DriverLog	6	8	7	8
	Depots	**9**	**6**	**8**	**7**
Hybrid	ZenoTravel	9	10	**12**	**7**
	DriverLog	6	9	**8**	**7**
	Depots	**8**	**6**	**9**	**4**

Table 13
Number of problems for each approach, domain and feature where the plan sets returned with the uniform (w08) distribution with better (i.e., higher) *standard deviation* of feature value than that with w08 (uniform), denoted in the table by $U >$ w08 (w08 $> U$, respectively). For each approach and feature, we mark bold the numbers for domains in which there are more problems whose plan sets returned with the uniform distribution have better standard deviation value of the feature than those with the w08 distribution.

Approach	Domain	SD of makespan		SD of cost	
		$U >$ w08	w08 $> U$	$U >$ w08	w08 $> U$
Sampling	ZenoTravel	**11**	**8**	**15**	**4**
	DriverLog	5	10	5	9
	Depots	**12**	**7**	**12**	**6**
Hybrid	ZenoTravel	**10**	**9**	**15**	**4**
	DriverLog	7	7	**8**	**6**
	Depots	5	8	**11**	**4**

- In generating plan sets sequentially to cope with the partial user preferences, the Sampling approach that searches for plans separately and independently of the solution space tends to return worse quality plan sets than the Hybrid approach.
- The resulting plan sets returned by the Hybrid approach tend to be more sensitive to the user preferences than those found by the Sampling approach.

7. Discussion

To the best of our knowledge, this work is a first step in domain-independent planning with preferences when the user preferences are not completely specified, in the same spirit of *model-lite* planning [33]. Our "language" to represent the partial preference model assumes a *complete* set of attributes of interest and a parameterized value function with unknown parameter values. Although in our work the unknown values are restricted in a *continuous* range, they can also be represented by a set of possible *discrete* values. These two representations of parameters' incompleteness are also the ways imprecise parameters are modeled in bounded-parameter MDPs [29] and MDPs with imprecise reward functions [45,46,55]. Boutilier et al. [9] consider the preference elicitation problem with a more general framework where both the set of attributes and the utility function are incomplete.

Our current representation and plan synthesis approach do have some limitations:

- The representation of the underlying complete preference model in our setting, i.e., the convex combination of metric quantities, is a subset of the preference language defined by PDDL3 [26], which has been commonly used to represent preferences in planning domains. In PDDL3, preferences are constraints on the state trajectory of plans with "penalty" values (or weights) of being violated, and a plan is more preferable if it has lower total penalty value. While one can model partially specified "penalty" for preferences in PDDL3 with a distribution over continuous range or set of discrete values, it is unclear how to represent incompleteness for other constructs of the language. Similarly, it is an interesting question on how incompleteness can be extended for conditional preferences [7].
- Using a convex combination of attributes as a utility function in our setting assumes that the criteria of interest are *mutual preferential independence*: although each attribute is important, it does not affect the way in which the user trades off the other attributes against each other [47]. This property may be violated, for instance when we want to extend this setting to include preference statements in PDDL3 as attributes of interest. In a travel domain, for example, a passenger might be more willing to accept a more expensive ticket for a non-stop flight if she has to fly at night (i.e., the weight on the importance of "cost" is smaller).
- Our current implementation ignores the fact that changing the scale on objectives (e.g. from "hours" to "minutes" in the makespan of plans) may change the bias of the distribution of the Pareto set of plans on the objective axis. In other

words, the set may look more uniform on the objective space using one scale than it is with a different scale [12]. Although the ICP value agrees with the set Pareto dominance relation regardless of the scaling mechanism used [21], this effect can introduce a wrong evaluation about the distribution of the entire Pareto set of plans in the objective space to a user observing the representative set of plans (which may be biased towards some region of an axis due to the scaling mechanism used).
- Given that IPF is a nonlinear function, it is a challenge to modify the Metric-LPG planner to efficiently search for a set of plans optimizing such a quality measure. We believe that the current modification of Metric-LPG used for our experiments can be improved by designing new specific heuristics that are more effective for optimizing the measure. In addition, as observed by Kim et al. [35], the computation time for IPF measure increases roughly exponentially with the number of objectives, and thus it is also challenging as to how to effectively incorporate the measure into the search for planning problems with a high number of criteria.

8. Conclusion and future work

In this paper, we consider the planning problem with partial user preferences in two scenarios where the knowledge about preferences is completely unknown or only part of it is given. We propose a general approach to this problem where a set of plans is presented to the user from which she can select. For each type of the preference incompleteness, we define a different quality measure for plan sets and investigate approaches to generating plan sets with respect to the quality measure. In the first scenario when the user is known to have preferences over plans, but the details are completely unknown, we define the quality of plan sets as their diversity value, specified with syntactic features of plans (its action set, sequence of states, and set of causal links). We then consider generating diverse sets of plans using two state-of-the-art planners, GP-CSP and LPG. The approaches we developed for supporting the generation of diverse plans in GP-CSP are broadly applicable to other planners based on bounded horizon compilation approaches for planning. Similarly, the techniques we developed for LPG, such as biasing the relaxed plan heuristics in terms of distance measures, could be applied to other heuristic planners. The experimental results with GP-CSP explicate the relative difficulty of enforcing the various distance measures, as well as the correlation among the individual distance measures (as assessed in terms of the sets of plans they find). The experiments with LPG demonstrate the potential of planning using heuristic local search in producing large sets of highly diverse plans.

When part of the user preferences is given, in particular the set of features that the user is interested in and the distribution of weights representing their relative importance, we propose the use of *Integrated Preference Function*, and its special case *Integrated Convex Preference* function, to measure the quality of plan sets, and propose various heuristic approaches based on the Metric-LPG planner [28] to find a good plan set with respect to this measure. We show empirically that taking partial knowledge of user preferences into account does improve the quality of the plan set returned to the user, and that our proposed approaches are sensitive to the degree of preference incompleteness, represented by the distribution.

While a planning agent may well start with some partial knowledge of the user preference model, in the long run, we would like the agent to be able to improve it through repeated interactions with the user. In our context, at the beginning when the degree of incompleteness is high, the learning will involve improving the estimate of $h(\alpha)$ based on feedback about the specific plan that the user selects from the set returned by the system. This learning phase is in principle well connected to the Bayesian parameter estimation approach in the sense that the whole distribution of parameter vector, $h(\alpha)$, is updated after receiving feedback from the user, taking into account the current distribution of all models (starting from a prior, for instance the uniform distribution). Although such interactive learning framework has been discussed previously, as in [16], the set of user's decisions in this work is assumed to be given, whereas in planning scenarios the cost of plan synthesis should be incorporated into the our interactive framework, and the problem of presenting plan sets to the user needs also to be considered. Recent work by Li et al. [36] considered learning user preferences in planning, but restricting to preference models that can be represented with hierarchical task networks.

Acknowledgements

We thank Menkes van den Briel for drawing our attention to ICP measure initially. Kambhampati's research is supported in part by an IBM Faculty Award, the NSF grant IIS2013308139, ONR grants N00014-09-1-0017, N00014-07-1-1049, N000140610058, and by a Lockheed Martin subcontract TT0687680 to ASU as part of the DARPA Integrated Learning program. Tuan Nguyen was also supported by a Science Foundation of Arizona fellowship.

References

[1] J. Baier, S. McIlraith, Planning with preferences, AI Magazine 29 (4) (2009) 25.
[2] M. Bienvenu, C. Fritz, S. McIlraith, Planning with qualitative temporal preferences, in: Proceedings of the 10th International Conference on Knowledge Representation and Reasoning (KR), 2006, pp. 134–144.
[3] G. Birkhoff, Lattice Theory, 2nd (revised) edition, American Mathematical Society Colloquium Publications, vol. XXV, American Mathematical Society, New York, 1948.
[4] A. Blum, M. Furst, Fast planning through planning graph analysis, Artificial Intelligence 90 (1–2) (1997) 281–300.
[5] M. Boddy, J. Gohde, T. Haigh, S. Harp, Course of action generation for cyber security using classical planning, in: International Conference on Automated Planning and Scheduling (ICAPS), 2005, pp. 12–21.

[6] C. Boutilier, A POMDP formulation of preference elicitation problems, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2002, pp. 239–246.
[7] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, D. Poole, CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artificial Intelligence Research 21 (1) (2004) 135–191.
[8] C. Boutilier, T. Dean, S. Hanks, Decision-theoretic planning: Structural assumptions and computational leverage, Journal of Artificial Intelligence Research 11 (1) (1999) 94.
[9] C. Boutilier, K. Regan, P. Viappiani, Simultaneous elicitation of preference features and utility, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2010, pp. 1160–1197.
[10] B. Bozkurt, J. Fowler, E. Gel, B. Kim, M. Köksalan, J. Wallenius, Quantitative comparison of approximate solution sets for multicriteria optimization problems with weighted Tchebycheff preference function, Operations Research 58 (3) (2010) 650–659.
[11] R. Brafman, C. Domshlak, Preference handling—An introductory tutorial, AI Magazine 30 (1) (2009) 58–86.
[12] J. Branke, Consideration of partial user preferences in evolutionary multiobjective optimization, Multiobjective Optimization 5252 (2008) 157–178.
[13] D. Bryce, W. Cushing, S. Kambhampati, Model-lite planning: Diverse multi-option plans & dynamic objective functions, in: ICAPS 2007 Workshop on Planning and Plan Execution for Real World Systems, 2007.
[14] W.M. Carlyle, J.W. Fowler, E.S. Gel, B. Kim, Quantitative comparison of approximate solution sets for bi-criteria optimization problems, Decision Sciences 34 (1) (2003) 63–82.
[15] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, B. Srivastava, Adaptation in Web Service Composition and Execution, in: International Conference on Web Services, ICWS'06, 2006, pp. 549–557.
[16] U. Chajewska, D. Koller, D. Ormoneit, Learning an agent's utility function by observing behavior, in: Proceedings of the Eighteenth International Conference on Machine Learning (ICML), 2001, pp. 35–42.
[17] U. Chajewska, D. Koller, R. Parr, Making rational decisions using adaptive utility elicitation, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2000, pp. 363–369.
[18] M. desJardins, K. Wagstaff, Dd-pref: A language for expressing preferences over sets, in: Proceedings of the National Conference on Artificial Intelligence, 2005.
[19] M. Do, S. Kambhampati, Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP, Artificial Intelligence 132 (2) (2001) 151–182.
[20] M. Do, S. Kambhampati, Sapa: A multi-objective metric temporal planner, Journal of Artificial Intelligence Research 20 (1) (2003) 155–194.
[21] J. Fowler, B. Kim, W. Carlyle, E. Gel, S. Horng, Evaluating solution sets of a posteriori solution techniques for bi-criteria combinatorial optimization problems, Journal of Scheduling 8 (1) (2005) 75–96.
[22] M. Fox, A. Gerevini, D. Long, I. Serina, Plan stability: Replanning versus plan repair, in: Proc. ICAPS, AAAI Press, 2006, pp. 212–221.
[23] M. Fox, D. Long, PDDL2.1: An extension to PDDL for expressing temporal planning domains, Journal of Artificial Intelligence Research 20 (1) (2003) 61–124.
[24] C. Fritz, S. McIlraith, Decision-theoretic golog with qualitative preferences, in: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, Lake District, UK, June 2006.
[25] M. Gelain, M. Pini, F. Rossi, K. Venable, T. Walsh, Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies, Artificial Intelligence 174 (3–4) (2010) 270–294.
[26] A. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners, Artificial Intelligence 173 (5–6) (2009) 619–668.
[27] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in LPG, Journal of Artificial Intelligence Research 20 (1) (2003) 239–290.
[28] A. Gerevini, A. Saetti, I. Serina, An approach to efficient planning with numerical fluents and multi-criteria plan quality, Artificial Intelligence 172 (8–9) (2008) 899–944.
[29] R. Givan, S. Leach, T. Dean, Bounded-parameter Markov decision processes, Artificial Intelligence 122 (1–2) (2000) 71–109.
[30] V. Ha, P. Haddawy, A hybrid approach to reasoning with partially elicited preference models, in: Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence, 1999, pp. 263–270.
[31] G. Hazen, Partial information, dominance, and potential optimality in multiattribute utility theory, Operations Research (1986) 296–310.
[32] E. Hebrard, B. Hnich, B. O'Sullivan, T. Walsh, Finding diverse and similar solutions in constraint programming, in: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI), 2005, p. 372.
[33] S. Kambhampati, Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain theories, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2007, p. 1601.
[34] H. Kautz, B. Selman, BLACKBOX: A new approach to the application of theorem proving to problem solving, in: AIPS98 Workshop on Planning as Combinatorial Search, 1998.
[35] B. Kim, E. Gel, J. Fowler, W. Carlyle, J. Wallenius, Evaluation of nondominated solution sets for k-objective optimization problems: An exact method and approximations, European Journal of Operational Research 173 (2) (2006) 565–582.
[36] N. Li, S. Kambhampati, S. Yoon, Learning probabilistic hierarchical task networks to capture user preferences, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 2009.
[37] G. Linden, S. Hanks, N. Lesh, Interactive assessment of user preference models: The automated travel assistant, in: Courses and Lectures—International Centre for Mechanical Sciences, 1997, pp. 67–78.
[38] H. McMahan, G. Gordon, A. Blum, Planning in the presence of cost functions controlled by an adversary, in: Proceedings of the Twentieth International Conference on Machine Learning (ICML), 2003, p. 536.
[39] A. Memon, M. Pollack, M. Soffa, Hierarchical GUI test case generation using automated planning, IEEE Transactions on Software Engineering 27 (2) (2001) 144–155.
[40] K. Myers, Metatheoretic plan summarization and comparison, in: International Conference on Automated Planning and Scheduling (ICAPS-06), 2006.
[41] K. Myers, T. Lee, Generating qualitatively different plans through metatheoretic biases, in: Proceedings of the National Conference on Artificial Intelligence, 1999, pp. 570–576.
[42] T. Nguyen, M. Do, S. Kambhampati, B. Srivastava, Planning with partial preference models, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kaufmann Publishers, 2009, pp. 1772–1777.
[43] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, IEEE, 1977, pp. 46–57.
[44] I. Refanidis, I. Vlahavas, Multiobjective heuristic state-space planning, Artificial Intelligence 145 (1–2) (2003) 1–32.
[45] K. Regan, C. Boutilier, Regret-based reward elicitation for Markov decision processes, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2009, pp. 444–451.
[46] K. Regan, C. Boutilier, Robust policy computation in reward-uncertain MDPs using nondominated policies, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, 2010, pp. 1127–1133.
[47] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.

[48] I. Serina, Kernel functions for case-based planning, Artificial Intelligence 174 (2010) 16–17.
[49] D. Smith, Choosing objectives in over-subscription planning, in: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling, 2004, pp. 393–401.
[50] T. Son, E. Pontelli, Planning with preferences using logic programming, Theory and Practice of Logic Programming 6 (5) (2006) 559–607.
[51] B. Srivastava, S. Kambhampati, T. Nguyen, M. Do, A. Gerevini, I. Serina, Domain-independent approaches for finding diverse plans, in: IJCAI, 2007, pp. 2016–2022.
[52] A. Tate, J. Dalton, J. Levine, Generation of multiple qualitatively different plan options, Technical Report, University of Edinburgh, 1998.
[53] M. Van Den Briel, R. Sanchez, M. Do, S. Kambhampati, Effective approaches for partial satisfaction (over-subscription) planning, in: Proceedings of the National Conference on Artificial Intelligence, 2004, pp. 562–569.
[54] P. Viappiani, B. Faltings, P. Pu, Preference-based search using example-critiquing with suggestions, Journal of Artificial Intelligence Research 27 (1) (2006) 465–503.
[55] H. Xu, S. Mannor, Parametric regret in uncertain Markov decision processes, in: Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 2009 28th Chinese Control Conference, CDC/CCC 2009, IEEE, 2009, pp. 3606–3613.
[56] Y. Zhang, J. Callan, T. Minka, Novelty and redundancy detection in adaptive filtering, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2002, pp. 81–88.