Finding User-friendly Linearizations of Partially Ordered Plans

Daniel Höller, Pascal Bercher, Felix Richter, Marvin Schiller, Thomas Geier, and Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany
{forename.surname}@uni-ulm.de

Abstract. Planning models usually do not discriminate between different possible execution orders of the actions within a plan, as long as the sequence remains executable. As the formal planning problem is an abstraction of the real world, it can very well occur that one linearization is more favorable than the other for reasons not captured by the planning model — in particular if actions are performed by a human. Post-hoc linearization of plans is thus a way to improve the quality of a plan enactment. The cost of this transformation decouples from the planning process, and it allows to incorporate knowledge that cannot be expressed within the limitations of a certain planning formalism. In this paper we discuss the idea of finding useful plan linearizations within the formalism of hybrid planning (although the basic ideas are applicable to a broader class of planning models). We propose three concrete models for plan linearization, discuss their ramifications using the application domain of automated user-assistance, and sketch out ways how to empirically validate the assumptions underlying these user-centric models.

Keywords: Hybrid Planning, POCL Planning, HTN Planning, Plan Linearization, User-centered Planning, User Assistance, Plan Execution

1 Introduction

In the last years we showed in different application domains that AI Planning can help to give a human user support on complex tasks like the operation of a smart phone, where the system helps the user to send messages, create contacts or appointments [2]. Recently we described a system that helps the user assemble a complex home theater system [1]. A hybrid planning system is utilized to find plans that help the user to reach her or his goal. Hybrid planning combines elements from Hierarchical Task Network (HTN) planning and Partial Order Causal Link (POCL) planning. Abstract tasks are decomposed repeatedly until all tasks are specific enough to be applied directly. Causal links make the causal structure of the plan explicit and ensure executability. The plans are partially ordered and thus enable a flexible execution order. Any of these orderings will transfer the initial state to a state that fulfills the desired goal properties.

However, when the plans have to be communicated to a human user, or are even executed by a human, some linearizations of the partially ordered plan might be better than others. We identified the following possible objectives for the task of plan linearization. The choice of the objective especially influences the experimental setup when the different utility functions are evaluated.

1. *imitate human behavior*
2. maximize the *user's subjective appraisal*
3. optimize some *objective metric* – this could be, e.g., the time a user needs to execute a plan

We do not want to imitate humans' behavior. In our context, the second and third objective are more relevant. Which of them to choose might depend on a specific application. In some situations, the user's appraisal is the top goal. In others a speedy execution of the plan might be the ultimate objective.

In this paper we introduce and discuss three utility functions, relying on different properties of the planning problem and its solution, that enable the comparison of different linearizations. These utilities are discussed in context of our home theater assistance system [1].

The following sections introduce the hybrid planning approach (Sec. 2) and our example domain (Sec. 3). Sec. 4 introduces the different utility functions that are discussed in Sec. 5. That section also outlines several alternatives for empirically evaluating the proposed utility functions. Sec. 6 concludes the paper.

2 The Hybrid Planning Framework

Hybrid planning [1,3] combines elements from Hierarchical Task Network (HTN) planning and Partial Order Causal Link (POCL) planning. Like in HTN planning, abstract tasks are repeatedly decomposed using *decomposition methods* until they can be executed directly. Like in POCL planning, the system utilizes *Causal Links* to prevent already established preconditions from being changed. The domain modeler is free to allow the system to insert primitive tasks independent from hierarchical decomposition.

A planning domain is a tuple $\mathcal{D} = \langle T, M \rangle$, where T is a set of tasks that can be partitioned into the sets T_C and T_P of *compound* and *primitive* tasks, respectively. It holds $T = T_C \cup T_P$ and $T_C \cap T_P = \emptyset$. Each $t(\overline{\tau}) \in T$ is a tuple $\langle \mathit{pre}, \mathit{eff} \rangle$, where *pre* and *eff* are conjunctions of literals over the task parameters $\overline{\tau} = \tau_1, \ldots, \tau_n$ and specify the preconditions and effects of a task. A *partial plan* has the structure $P = (PS, \prec, VC, CL)$, where PS is a set of plan steps, i.e. of uniquely identified tasks. $\prec$ is a set of constraints of the form (t_i, t_j) that define a strict partial order on the tasks in PS. VC is a set of variable constraints that codesignate or non-codesignate task parameters to other task parameters or to constants. CL is a set of causal links. A causal link has the form $t \rightarrow_{\varphi} t'$ and states that the precondition φ of task t' is established by task t. The precondition φ is said to be *supported* by that causal link.

As given in the beginning, the compound tasks of a plan have to be decomposed until only primitive tasks are left. This is done using (decomposition) methods given by the second domain element M. A method $m \in M$ has the form

$m = \langle t(\overline{\tau}), P\rangle$ and defines a mapping of the compound task $t(\overline{\tau})$ to a partial plan P.

A planning problem $\mathcal{P}$ is given by a planning domain and a definition of an initial state, a goal state and an initial partial plan P_{init}. Initial and goal state are usually given as two special primitive tasks *init* and *goal* that have the initial state as effect and the goal state as precondition, respectively. States and the applicability of primitive tasks are defined as usual. Fully grounded primitive tasks are also called *actions*.

A partial plan $P = (PS, \prec, VC, CL)$ is a solution to a planning problem if and only if the following conditions hold:

1. It is a refinement of the initial plan P_{init}, i.e. it is achieved by decomposition, insertion of causal links and ordering constraints, and (if intended by the domain designer) task insertion.
2. It does not contain compound tasks.
3. All preconditions are supported by causal links.
4. There is no causal threat. A threat is a situation where a task t' that has an effect $\neg\varphi$ can be placed between two tasks t and t'' when there is a causal link $t \rightarrow_{\varphi} t''$ in CL. A task t' can be placed between the tasks t and t'' when $\prec \cup \{(t, t'), (t', t'')\}$ is also a valid strict partial order.

A *partial plan* that is a solution is also referred to as *plan*. We denote it by $P^* = (PS^*, \prec^*, VC^*, CL^*)$.

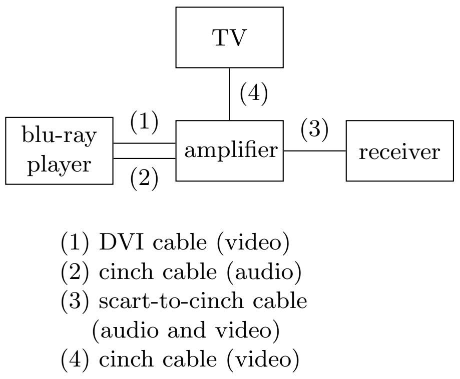
3 Example Domain

We exemplify the proposed utility functions in the home theater domain [1, Section "Domain Model"]. In that domain several devices, such as a blu-ray player, a satellite receiver, an amplifier (audio/video receiver), and a television, have to be connected with each other, s.t. the television receives the audio signals of the blu-ray player and the satellite receiver and the television receives the video signals of these devices. Many different cables are modeled, such as cinch or DVI and HDMI cables. The specific devices and cables are modeled by constants of the respective sort.

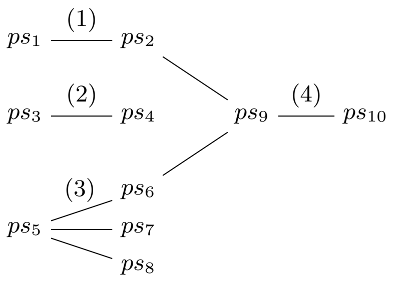
For connecting these devices using cables, the domain features *plugIn* actions that take as argument four constants: the two hardware components that have to be connected with each other (where one is a cable and the other a device such as a blu-ray player) and the two ports that are connected (for example, an HDMI port in case of a blu-ray player and one of the ends of an HDMI cable). Thus, for connecting two devices using one single cable, (at least) two actions are needed: one for each end of the cable. The domain is modeled in such a way that actions may only be executed in the order in which the signal is transported from its source to its destination. So, if a cinch cable were utilized to transport the audio signal of the blu-ray player to the amplifier, then the cable needs first to be plugged into the blu-ray player and afterwards into the amplifier although the pure "physics" would also allow the other execution order.

As an example, consider the following solution to the given planning problem:

The blu-ray player is connected to the amplifier using two cables: a DVI cable is used for video and a cinch cable for the audio signal. The satellite receiver is connected to the amplifier using a scart-to-cinch cable. That cable transports both audio and video. It uses a scart connector at one end (that is plugged into the satellite receiver) and three single cinch ports at the other: two transport the audio signal and one the video. Finally, the amplifier is connected to the television using a further cinch cable. Since putting an end of a cable into a device is modeled using a single action, that solution consists of 10 plan steps (two actions for each cable except scart-2-cinch, which requires four). This solution is depicted graphically in Fig. 1.



(a) Obtained setup



(b) Solution plan

Fig. 1: Fig. (a) gives a graphical representation of the solution to the described problem of the home theater domain. The graph in Fig. (b) schematically depicts a solution plan to the corresponding planning problem. Nodes ps_i represent plan steps. There is an edge between two nodes if and only if there is at least one causal link between the corresponding nodes in the graph. The plan steps ps_1 and ps_2 represent the *plugIn* actions to connect the blu-ray player to the amplifier using a DVI cable, whereas ps_3 and ps_4 connect those devices using a cinch cable. The plan steps ps_5 to ps_8 represent the *plugIn* actions for connecting the satellite receiver with the amplifier. Audio connection is established using ps_7 to ps_8 (left and right channel) and video using ps_6. The amplifier is connected to the television with the cinch cable using ps_9 and ps_{10}. Since cables may only be connected in the order of signal transportation, the execution of ps_9 requires that ps_2 and ps_6 have already been executed (since afterwards the amplifier has both required video signals that are transported to the television).

From the solution criteria of hybrid planning we can be sure that any plan step linearization respecting the ordering constraints and causal structure is an executable solution. However, some linearizations might be confusing to some users. Consider the plan in Fig. 4: The sequence $\langle ps_1, ps_5, ps_3, ps_6, ps_2\rangle$ is a prefix of a valid linearization of its plan steps. However, the user repeatedly switches back and forth between different cables and devices in that linearization:

1. ps_1: plug **DVI cable** into **blu-ray player**
2. ps_5: plug scart end of **scart-to-cinch cable** into **satellite receiver**
3. ps_3: plug **cinch cable** into **blu-ray player**
4. ps_6: plug cinch-video end of **scart-to-cinch cable** into **amplifier**
5. ps_2: plug the other end of the **DVI cable** into **amplifier**

A more reasonable solution would be to execute ps_2 directly after ps_1 and ps_4 directly after ps_3, and so on. In the following, we introduce three different possibilities to choose reasonable execution orders. These are based on the actions' parameters, on the causal link structure of the plan, and on the hierarchical structure of the planning domain.

4 Plan Linearization

This section gives three different utility functions for plan linearization. The intuition behind the following definition of the overall utility is that a good next action to execute has to fit into the context, i.e. the sequence of actions that have already been executed. Therefore the overall utility is a sum of local utilities between each action and its predecessors in the linearization.

Definition 1 (Utility of Plan Linearizations)
Given a plan $P^* = (PS^*, \prec^*, VC^*, CL^*)$ and a linearization of its plan steps $L = \langle ps^1, ps^2, \ldots, ps^n\rangle$ that is valid with respect to $\prec^*$, we define the (overall) utility $U(L)$ of the linearization as

$$U(L) = \sum_{i=1}^{n} \sum_{j=1}^{i-1} w(j) * u(ps^i, ps^{i-j})$$

The additional weight function $w(j)$ is introduced to allow for different factors, e.g. to give the recently executed actions more weight than those at the beginning of the execution. The following subsections give three possible definitions of that local utility function $u(ps, ps')$. The first one is based on the actions' parameters, the second on the causal links, and the last one on the domain's task hierarchy.

4.1 Parameter Similarity

The first (local) utility function is entirely based on the steps in the plan. More precisely, it employs a pairwise comparison of two plan steps' parameters. This is based on the assumption that the tasks represent things to do, while the parameters represent entities that are necessary to do so. In our example the user may want to complete connecting a specific cable or device before switching his attention to other devices or cables.

Definition 2 (Parameter-based Utility)
We define the parameter-based utility u^p of two ground plan steps $ps^1(c^1_1, \dots, c^1_m)$ and $ps^2(c^2_1, \dots, c^2_n)$ as

$$u^p(ps^1(c^1_1, \dots, c^1_m), ps^2(c^2_1, \dots, c^2_n)) = \|\sum_{i=1}^{m} \sum_{j=1}^{n} u^c(c^1_i, c^2_j)\|$$

where $\|\cdot\|$ is an arbitrary normalization operator.

The given function compares each pair of the parameters of ps^1 and ps^2. A simple function to compare two parameters is given in Def. 3. More sophisticated functions could e.g. include the parameters' sorts when they are not identical.

Definition 3 (Parameter Similarity)

$$u^c(c_1, c_2) = \begin{cases} 1, & \textit{if } c_1 = c_2 \\ 0, & \textit{else} \end{cases}$$

After summing up the different parameter similarities, the overall utility value of the plan steps has to be normalized to prevent the approach from preferring plan steps with many parameters. A straightforward realization of this function would consist in, for example, dividing the utility by the product of the parameter count of both plan steps (if greater than zero).

There are some self-evident extensions of the given utility definition, e.g. by including also the task schemata of the plan steps. In our domain, only a single task schema is used (there are only *plugIn* actions), but in general several different task schemata are possible; depending on the domain it might be preferable to execute similar task schemata consecutively. For instance, in another domain, it might be plausible to execute all "shopping" actions together before (or after) executing "cleaning" actions.

4.2 Causal Link Structure

The next utility function is based on the plan's causal link structure. Like POCL planning, our planning approach is problem-driven, i.e. every causal link in the plan is necessary to support some precondition. Thus the structure of the causal links represents the causal structure of the plan. This causality information can be utilized for plan linearization.

Definition 4 (Causal Link-based Utility)
We define the causal link-based utility of two plan steps ps^1 and ps^2 as

$$u^{cl}(ps^1, ps^2) = u^l(\{\varphi \mid (ps^1 \rightarrow_\varphi ps^2) \in CL^*\})$$

where $u^l(\cdot)$ maps to an utility number.

The function $u^l(\cdot)$ maps the set of causal links between two plan steps to a utility number. One possibility is the set cardinality. However, there are several alternatives, e.g. to return 1 if there is at least one link and 0, otherwise.

The given definition is based on the causal links between two plan steps. In combination with an appropriate weight function w, this utility results in linearizations where the steps are ordered in a way that the preconditions of some step are established (directly) predecessing plan steps.

4.3 Decomposition Information

Since a planning domain is (usually) modeled by a human, the way the task hierarchy is modeled might also carry information. Partial plans that are used in decomposition methods are implementations of their abstract tasks [3]. Tasks that are within the same partial plan can hence be considered to be semantically related. We generalize this relationship between tasks in the plan of the same method to those of different methods and give another utility that is based on the *Task Decomposition Graph* (TDG). Our definition of the TDG is a simplified variant of the TDG that was used in earlier work [4].

Definition 5 (Task Decomposition Graph)
Let $\langle V, E\rangle$ be an AND/OR *graph with the set of vertices V and the set of edges $E \subseteq V \times 2^V$. We define the Task Decomposition Graph (TDG) as the minimal graph that fulfills the following specification:*

1. *base case:*
   (a) $t_{init} \in V$*, where t_{init} is a new artificial root node*
   (b) $V \supseteq \bigcup_{T \in Ground(PS_{init}, VC_{init})} T$
   (c) $E \supseteq \bigcup_{T \in Ground(PS_{init}, VC_{init})} \{(t_{init}, T)\}$
2. $\forall t(\overline{c}) \in V : \langle t(\overline{\tau}'), \langle PS', \prec', VC', CL'\rangle\rangle \in M \wedge \theta$ *is mgu of* $\overline{c}$ *and* $\overline{\tau}'$ $\Rightarrow$
   (a) $V \supseteq \bigcup_{T \in Ground(PS', VC' \cup \theta)} T$ *and*
   (b) $E \supseteq \bigcup_{T \in Ground(PS', VC' \cup \theta)} \{(t(\overline{c}), T)\}$

Where $Ground(PS, VC)$ denotes a set of sets. Each element is a set of ground tasks obtained by grounding the plan steps PS with respect to the variable constraints in VC. The returned outer set contains all valid groundings.

The TDG includes nodes for every ground instance of every task in the initial task network (1b). To connect these nodes in the initial plan, a new node is introduced (1a) that is the parent node of them (1c). For every task in V and every compatible method, the groundings of the tasks in the method's plan are also nodes in the graph (2a). And each compatible grounding of the method's plan is connected via a multi-edge (a *connector*) to the parent task (2b).

Definition 6 (Decomposition-based Utility)
Let the decomposition-based utility between two plan steps $l_1{:}t_1$ and $l_n{:}t_n$ be defined as

$$\begin{aligned} h^d(l_n{:}t_n, l_m{:}t_m) = \min\{n + m \mid & \langle t_1, \ldots, t_n\rangle, \langle t'_1, \ldots, t'_m\rangle,\ t'_1 = t_1,\ t'_m = t_m, \text{ and} \\ & \text{for } 1 < j \leq n \text{ holds } (t_{j-1}, V') \in E \text{ and } t_j \in V', \\ & \text{for } 1 < k \leq m \text{ holds } (t'_{k-1}, V') \in E \text{ and } t'_k \in V'\} \end{aligned}$$

The utility gives the minimal distance of two tasks in the possible decompositions from the initial plan. To reach one node from the other, one has to go up the tree until a node is reached that is a parent node of both of them. Afterwards one has to go down the tree until the second node is reached. Since here we return some kind of similarity, please be aware that this function has to be minimized (in contrast to the others). As this utility relies solely on decomposition it can, in the current form, not deal with task insertion (cf. solution criterion (1)).

When a planning system generated a solution, it has found *one* valid decomposition of the initial plan that resulted in the plan at hand. That decomposition, called Decomposition Tree (DT) [5], is quite related to our TDG. From a practical point of view the node distance can be taken from this tree. However, while the TDG gives a measure for the semantic similarity of tasks, the DT is affected by the preconditions and effects as well as by the planning process itself. It is just one valid decomposition, but might not be minimal.

5 Discussion

In this section we first discuss how the proposed utility functions would perform in the home theater domain. Afterwards, we discuss how the different linearizations could be evaluated in a field study with human users.

5.1 Linearization Behavior in the Home Theater Scenario

We will now discuss the proposed utility functions based on the example solution depicted in Fig. 1.

Parameter-based Utility. Recall that each *action* takes four arguments: the source hardware like a device (e.g., a blu-ray player) or a cable, its port (where also cables have "ports", these are their ends), and the destination hardware and port. Let us assume the plan step ps_1 is picked first for being presented to the user. We regard it as plausible to continue with ps_2, because both plan steps are concerned with connecting the DVI cable. We know that both ps_1 and ps_2 have a common constant that represents this cable. We can assume that the parameter similarity-based utility function may select ps_2 after ps_1 rather than selecting ps_5, where completely different devices and cables (hence disjoint constants) are used.

Note that the question how ties are broken (in case two actions are similar with respect to their parameter similarity-based utility value) can also influence the resulting linearization. In our example, recall that ps_1 and ps_2 share exactly one constant: the one representing the DVI cable. However, ps_1 and ps_3 also share a constant: the one representing the blu-ray player. Thus, rather than selecting ps_2 after the execution of ps_1, also ps_3 might be selected. Both selection strategies might be regarded useful: the one completes connecting the *cable* that was used last, whereas the other completes connecting the *device* that was used last.

Causal Link-based Utility. If one looks at the solution plan in Fig. 1 it seems quite natural that we execute ps_2 directly after ps_1 and, as another example, ps_6 to ps_8 directly after ps_5. Why is this the case? Because, as depicted in the figure, these actions are causally related to each other. In hybrid planning, these causal dependencies are modeled explicitly using causal links and can thus easily be used for plan linearization.

Again, assume ps_1 was picked first. Since the only plan step to which ps_1 is causally related is ps_2, that plan step is picked next. The next causally relevant plan step (for ps_2) is ps_9. However, because ps_9 requires ps_6 to be executed first, another plan step must be chosen.

The question of tie-breaking is also quite important for this plan linearization strategy as can be seen with the next example. Assume ps_5 (putting the scart end of the scart-to-cinch-cable in the satellite receiver) is selected for execution after ps_2. The next candidates are ps_6 to ps_8 (which correspond to plugging the cinch ends of the scart-to-cinch-cable into the amplifier). Assuming ps_6 were selected next, we are free to select either ps_9 or ps_7 or ps_8 next. The first tie-breaking strategy corresponds to depth-first selection, whereas the latter correspond to breadth-first selection. Selecting in a depth-first manner corresponds to establishing the video signal for the television first. Selecting in a breadth-first manner would first complete connecting the scart-to-cinch cable before proceeding.

Decomposition-based Utility. The last linearization technique is based on the decomposition hierarchy of the planning domain. Suppose the planning problem is given by three initial abstract *connect* tasks each taking two devices, which need to be connected. For each abstract task, the decomposition methods contain pre-defined standard solutions for these tasks. For instance, the two plan step sequences $\langle ps_1, ps_2\rangle$ and $\langle ps_3, ps_4\rangle$ can be put together in a partial plan P that is used by a decomposition method $m = \langle t(\bar{c}), P)\rangle$ for the task $t(\bar{c}) = \textit{connect}(\textsc{bluray}, \textsc{amplifier})$. Analogously, the sub plan in Fig. 1 involving the plan steps ps_5 to ps_8 can be defined as a standard solution to the task $\textit{connect}(\textsc{receiver}, \textsc{amplifier})$.

Now, let us say ps_1 is picked first for execution. Since the decomposition distance to ps_2 and ps_3 is the same and smaller than the distance to ps_5, one of these two plan steps is picked first. So, all plan steps ps_1 to ps_4 will be presented for execution before any other plan step of the solution. For tie-breaking between these plan steps, further strategies need to be considered since all of them have the same decomposition distance according to Def. 6.

5.2 Empirical Evaluation

The ultimate goal of the presented techniques is to assist users in real-world scenarios. The suitability of the three different utility functions for plan linearization and their relative strengths and weaknesses therefore call for empirical investigation. Can one of them be considered preferable, or does this depend, for example, on the application domain? Such questions are shared with other subfields in HCI, where several alternative forms of empirical evaluation have been employed (see e.g. [6,7]):

1. the output/behavior of a system is compared to human experts (requiring a metric to quantify the difference),
2. users provide a subjective rating for a system's performance, or
3. users have to solve a task assisted by a system, and task performance is used as the test metric.

These three forms of empirical evaluation measure different objectives, akin to the ones presented in Sec. 1:

1. in how far a system succeeds in *imitating* humans,
2. a system's performance as subjectively perceived by prospective users, or
3. an objective metric for a system's capability in assisting users, for example, users' performance in carrying out a plan presented using the techniques outlined in this paper, as measured, for example, in terms of time taken to carry out a plan.

For the evaluation of the three utility functions for plan linearization, a first step is to identify test cases where the different approaches lead to different solutions (i.e. plan linearizations). The most simple experimental design would follow the second approach to evaluation, namely to present these different plan linearizations to the participants of an experiment, whose task is to rate them for appropriateness.

More concretely, such a simple experiment might consist in randomly assigning participants to three groups, each of which is presented with different plan linearizations, all for the same sequence of plans. The use of the three utility functions for generating the linearizations is counterbalanced across the three groups, such that for each plan each of the three approaches is used for one third of the participants, allowing for the average ratings from the participants to be compared (i.e., a latin square type of experiment design). For the presentation of plan steps within such an experiment, several options are available, ranging from simple lists to speech output [2]. In order to avoid confounds, in an experimental setting the most simple one is clearly preferable (i.e., a bullet point list).

As to the task that the participants have to fulfill in the experiment, the most simple design would ask them to provide a subjective rating for the plan linearizations, similar to the approach in [1]. This would reveal only the perceived usefulness or plausibility (as judged by the participants), not an objective measure for the effect of a particular ordering of plan steps has on a user's success in carrying out a suggested plan.

By contrast, an experimental design measuring participants' performance in carrying out plans with steps ordered by different linearizations would offer greater practical validity, but might fail to deliver results for various reasons. For example, if the problem turns out to be too easy or too difficult in general, or if the way the generated plan steps are presented turns out to be unsuitable, differences in the employed linearization might not show strongly enough to be measured reliably.

The remaining option for designing an experiment – the first item in the above list – would be to offer participants the possibility to arrange an initially unordered plan in the order they prefer to see, and to evaluate the employed linearization approaches according to their agreement with the most frequent solutions. However, one needs to be careful that the order produced by the participants is not unintentionally influenced by potential flaws in the experimental setup, for example if the method through which participants are introduced to the plan, its steps, and by which they are to manipulate and indicate the final order of steps is biased towards a particular solution.

Another problem to consider is that the success of the given approaches (first of all, the quality of the linearizations) depends on the way the domain is modeled and, in case of the causal link utility function, the planning process that is used to generate the plans. This is due to the fact that (some of) the decisions of the planning system are reflected in the causal links. Thus the relative performance of linearization metrics might change when they are applied in different domains. However, these domain-independent methods might be a good starting point for more elaborated domain-specific heuristics.

6 Conclusion

Though each linearization of a partially ordered solution plan solves the given planning problem, a system that interacts with a human user should come up with a linearization that is intuitive to her or him. We addressed this issue in the context of a system that helps the user to assemble a complex home theater. We introduced three utility functions that are based on different properties of the planning problem and its solution. The first one relies solely on the similarity of the steps in the given plan. The second and third function benefit from our hybrid planning approach: they exploit the causal link structure generated during the planning process, and the way in which a solution was decomposed from the initial abstract plan. Further we discussed how one would go on to empirically evaluate the proposed models.

Acknowledgment

This work is done within the Transregional Collaborative Research Centre SFB/ TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

1. Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: Plan, repair, execute, explain - how planning helps to assemble your

home theater. In: Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014). pp. 386–394. AAAI Press (2014)
2. Biundo, S., Bercher, P., Geier, T., Müller, F., Schattenberg, B.: Advanced user assistance based on AI planning. Cognitive Systems Research 12(3-4), 219–236 (2011), special Issue on Complex Cognition
3. Biundo, S., Schattenberg, B.: From abstract crisis to concrete relief (a preliminary report on combining state abstraction and HTN planning). In: Proceedings of the 6th European Conference on Planning (ECP 2001). pp. 157–168 (2001)
4. Elkawkagy, M., Bercher, P., Schattenberg, B., Biundo, S.: Improving hierarchical planning performance by the use of landmarks. In: Proceedings of the 26th National Conference on Artificial Intelligence (AAAI 2012). pp. 1763–1769. AAAI Press (7 2012)
5. Geier, T., Bercher, P.: On the decidability of HTN planning with task insertion. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011). pp. 1955–1961 (2011)
6. Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., Sykes, C.: Automatic generation of textual summaries from neonatal intensive care data. Artificial Intelligence 173(7), 789–816 (2009)
7. Reiter, E., Belz, A.: An investigation into the validity of some metrics for automatically evaluating natural language generation systems. Computational Linguistics 35(4), 529–558 (2009)