Search Strategies for Partial-Order Causal-Link Planning with Preferences

Pascal Bercher and Fabian Ginter and Susanne Biundo

Institute of Artificial Intelligence, Ulm University, Germany

Abstract. This paper studies how to solve classical planning problems with preferences by means of a partial-order causal-link (POCL) planning algorithm. Preferences are given by soft goals – optional goals which increase a plan's benefit if satisfied at the end of a plan. Thus, we aim at finding a plan with the best *net-benefit*, which is the difference of the achieved preferences' benefit minus the cost of all actions in the plan that achieves them.
While many approaches compile soft goals away, we study how they can be addressed *natively* by a POCL planning system. We propose novel search and flaw selection strategies for that problem class and evaluate them empirically.

1 Introduction

Partial-order causal-link (POCL) planning in the tradition of SNLP [16] and UCPOP [17] provides plans as partially ordered sets of actions where causal dependencies between the actions are explicitly shown. This allows for flexibility with respect to the order in which actions can finally be executed. Moreover, it enables a human user to grasp the causal structure of the plan and to understand why certain actions are part of it [22]. Therefore, POCL planning is particularly suitable when plans have to be generated that are not to be carried out by machines or systems, but by humans. This appears, for example, in applications that realize support of the elderly [1,7,19] or provide assistance in some daily routines [5,6]. Plan quality is of particular importance in this kind of applications, since plans have to be "accepted" by the user which is more likely if they respect the individual's personal preferences.

In this paper, we address POCL planning with preferences by optimizing the net-benefit of solutions. In net-benefit planning [11], a classical planning problem is augmented by a set of soft goals: state features one would like to see satisfied in the final state produced by a solution plan, but if they are not, the plan is still regarded a valid solution. Each soft goal has a certain benefit which has a positive impact on the solution quality; every action contained in the solution decrements the respective quality, however. Thus, the goal is to find a solution to a planning problem that satisfies the preferences to the largest extent while taking into account the negative impact, like costs or time consumption, of the actions necessary to achieve them. This way, individual user preferences

can be defined through soft goals with a corresponding (individual) benefit. We describe these preferences by arbitrary formulae over state features, so-called *simple preferences*.

In the literature, several kinds of preferences like simple preferences, state trajectory constraints, and action occurrences have been addressed. Many planning systems exist capable of handling such preferences and implementing various paradigms and search algorithms including hierarchical task network (HTN) planning [15,24], search in the space of states [2,3,8], and search using binary decision diagrams [10]. While several of these approaches keep the explicit representation of the preferences, other approaches suggest to compile them away [9,14]. This allows to use any planning system without making any changes due to the presence of preferences. If one does not compile the preferences away, one can either select a subset of the optional planning goals before the planning process and regard all of them as non-optional [23], or one keeps the preferences and decides during the planning process on which preference to work on, as it is done in this work. Not compiling away preferences but addressing them natively allows to incorporate a human user into the planning process, s.t. he can actively influence the planning process [21].

In our setting, a problem specification is given in terms of a POCL problem. The preferences are specified by so-called *at-end* preferences, as described in PDDL3 [11], the language for the fifth International Planning Competition (IPC-5). An at-end preference (or simple preference) is an arbitrary formula over state features which should hold at the end of a plan if possible. We developed a POCL planning algorithm that is capable of solving this kind of problems. Each preference is transformed into disjunctive normal form (DNF). Our algorithm employs two different strategies to handle disjunctive preferences. Furthermore, several flaw selection strategies for planning with preferences were implemented. We evaluated these strategies on a large number of problems and show how they influence planning performance.

2 Problem Setting

POCL planning performs search in the space of plans. A plan P in POCL planning is a tuple $(PS, V, C, \prec)$. PS is a set of labeled plan steps, i.e., each plan step $l{:}o$ consists of a (partially) instantiated operator o and a label $l \in L$ to differentiate between multiple occurrences of the same operator, L being an infinite set of label symbols. For the sake of simplicity, we refer to a plan step $l{:}o$ by l. Operators are defined as usual, consisting of a precondition and an effect, each of which is a conjunction of literals. A ground operator is called an action and every action a has an associated cost, denoted by `cost(a)`. V is a set of variable constraints $v \circ x$ with v being a variable and $\circ \in \{=, \neq\}$ denoting a co- or a non-co-designation with x, x being an object or another variable. Please note that the variable constraints V can be interpreted as a constraint satisfaction problem (CSP) in which domains of the variables are the available objects. C is a set of causal links. A causal link (l, ϕ, l') denotes that the precondition literal

ϕ of plan step l' is an effect of plan step l and that ϕ is *supported* or *protected* that way. Causal links are used to ensure that all preconditions of all plan steps in a plan are satisfied in the state in which they are to be executed. The set $\prec$ of ordering constraints $(l, l'), l, l' \in L$ is used to resolve conflicts occurring during search.

A POCL planning problem π is a tuple $(\mathcal{O}, P_{init})$, where $\mathcal{O}$ is the set of available operators and P_{init} is the initial plan. The initial plan contains two artificial actions l_0:`init` and l_∞:`goal` which encode the initial state and the goal description, respectively: l_0 has no precondition and the initial state as effect, whereas l_∞ has no effect and the goal description as precondition. Initially, $(l_0, l_\infty) \in \prec$ and during planning all inserted plan steps are inserted between these two actions.

A plan $P_{sol} = (PS, V, C, \prec)$ is called a solution of π if and only if the following criteria hold:

1. P_{sol} is a refinement of $P_{init} = (PS', V', C', \prec')$, i.e., $PS \supseteq PS', V \supseteq V', C \supseteq C'$, and $\prec \supseteq \prec'$,
2. every precondition is protected by a causal link, i.e., for every precondition literal ϕ of any plan step $l{:}o \in PS$ there is a plan step $l'{:}o' \in PS$, s.t. $(l, \phi, l') \in C$,
3. no causal links are threatened, i.e., for each causal link $(l, \phi, l') \in C$ the ordering constraints $\prec$ and variable constraints V ensure that no plan step $l''{:}o'' \in PS$ with an effect $\neg\psi$ can be ordered between l and l', s.t. $\neg\phi$ is unifiable with $\neg\psi$,
4. the ordering constraints and causal links are free of contradiction, i.e., $\prec$ does not contradict the ordering induced by C and $\prec$ does not induce cycles, and
5. P_{sol} is ground, i.e., each variable in V has a domain of size 1.

Please note that these syntactical solution criteria ensure that every linearization of the plan steps in P_{sol} that is compatible with its ordering constraints is an action sequence that is executable in the initial state and leads to a state satisfying the goal description.

In addition to a planning problem π, we are given a set of preferences $\mathcal{P}$, each of which is an arbitrary formula over state features. The function $b : \mathcal{P} \to \mathbb{R}$ maps each preference p to its benefit $b(p)$. The preferences follow the semantics of the at-end preferences described in PDDL3. Thus, a solution P_{sol} satisfies a preference p if and only if every linearization of the actions in P_{sol} that is compatible with its ordering constraints generates a state s such that s satisfies p.

Let $P_{sol} = (PS, V, C, \prec)$ be a solution to the planning problem π. We are interested in finding good solutions w.r.t. action costs and satisfied preferences. To that end, the solution quality is defined by its net-benefit $\texttt{netBen}(\texttt{P}_{sol}) := \sum_{p \in \mathcal{P}(P_{sol})} b(p) - \sum_{a \in PS} \texttt{cost}(\texttt{a})$, where $\mathcal{P}(P_{sol})$ denotes the set of preferences satisfied by P_{sol}.

3 A POCL Algorithm for Planning with Preferences

In this section, we introduce our POCL planning algorithm (cf. Alg. 1) that is capable of solving problems containing simple preferences.

Algorithm 1: Preference-based POCL algorithm

Input : The fringe $\texttt{fringe} = \{(P_{init}, \texttt{flaws}_h(P_{init}), \texttt{flaws}_s(P_{init}))\}$.
Output : A plan or `fail`.

```
1  best-netBen := −∞
2  best-P := fail
3  while fringe ≠ ∅ and "no timeout" do
4  |  N := (P, flaws_h(P), flaws_s(P)) := planSel(fringe)
5  |  if flaws_h(P) = ∅ and netBen(P) > best-netBen then
6  |  |  best-netBen := netBen(P)
7  |  |  best-P := P
8  |  f := flawSel(flaws_h(P) ∪ flaws_s(P))
9  |  fringe := (fringe \ {N}) ∪ resolveFlaw(N, f)
10 return best-P
```

Alg. 1 is a standard *flaw-based* POCL algorithm, meaning that every violation of a solution criterion is represented by a so-called *flaw*. More precisely, *for every plan component* that is involved in the violation of solution criteria 2. to 5. (cf. previous section), like a precondition of a plan step that is not yet protected by a causal link, one flaw is introduced to the set of all flaws of the current plan. Hence, a plan is a solution if there are no flaws. The main procedure consequently follows the idea of (1) selecting a promising search node/plan from a search fringe, (2) select a flaw for the current plan, and (3) resolve the selected flaw using all possible plan modifications and insert the resulting successor plans into the fringe; this is also the step, where new flaws are calculated. This loop is continued until the search space is exhausted or some timeout criterion is satisfied (cf. line 3).

Our algorithm still follows that basic procedure; however, there are important additional considerations one has to make, since we want to search for a solution with the best net-benefit, which imposes certain problems to standard flaw-based POCL algorithms.

To be able to search for plans which satisfy preferences, we need to include them in the plan structure. To that end, we first normalize the preferences by transforming them into DNFs. Then, we alter the structure of the artificial goal action l_∞, s.t. it additionally contains all preferences. As our POCL planning algorithm is based on the concept of flaws, we include additional flaws for the unsatisfied preferences. However, since these flaws do not violate a solution criterion, they must be distinguishable from the "ordinary" ones. Hence, we differentiate between two classes of flaws: *hard flaws*, as they were already described in

the beginning of this section, and *soft flaws*, which indicate the non-satisfaction of a preference. The test for a solution is then performed purely based on the set of hard flaws (cf. line 5).

Alg. 1 has two "decision" points: (1) which search node/plan to work on and (2) given a search node, which flaw should be resolved next and *how* will it be resolved. The implementation of the first decision point defines the main search strategy. A reasonable strategy is A* with some heuristic judging the quality or goal distance of a plan. Hence, this heuristic should incorporate both action costs and preferences. We developed such a heuristic in previous work [4] and thus focus on the second decision point in this paper: the flaw selection in the presence of preferences (cf. line 8).

The question which flaw to resolve next is of major importance for the performance of any POCL planner, as the order in which flaws are selected directly influences the produced search space. Hence, several papers address the problem of finding appropriate flaw selection functions in POCL planning [12,13,18,20], but none is tailored to the selection of soft goals. The specification and empirical evaluation of flaw selection strategies for POCL systems in the presence of soft flaws is thus one of our main contributions.

The question *how* to resolve soft goals rises from two sources: First, a soft flaw does not *need* to be resolved and must consequently be treated differently than a hard flaw. Second, since soft flaws are represented as a DNF of literals in this work, many standard flaw selection strategies are not applicable anymore, since they are based on flaws being single literals.

4 Soft Flaw Resolution Strategies

The function $\texttt{resolveFlaw}(N, f)$ in line 9 of Algorithm 1 calculates all successor plans that resulted from addressing and/or resolving the previously selected flaw f in the plan P of the current search node $N = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P))$.

An important observation when planning in the presence of soft flaws is that it is not sufficient to alter the POCL solution criterion from "there are no flaws" to "there are no hard flaws" (cf. line 5). Note that the flaw selection is not a backtrack point, as in standard POCL planning (without optional goals), *all* flaws need to be resolved and hence the actual order is irrelevant for completeness. Since working on a preference is optional, selecting and resolving a soft might invalidate completeness; hence, one *must* be able to ignore a soft flaw in order to maintain completeness. Thus, given a search node $N = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P))$ and a soft flaw f, the function $\texttt{resolveFlaw}(N, f)$ must also include $(P, \texttt{flaws}_h(P), \texttt{flaws}_s(P) \setminus \{f\})$. Otherwise, the choice to work on the soft flaw f could have the effect that certain solutions might not be found. Please note that this observation generalizes to POCL planning with soft flaws and is thus not specific to the case in which a preference is an at-end preference in DNF.

Given an unsatisfied preference p in DNF, there are several possibilities how to work on p, i.e., how to define the function $\texttt{resolveFlaw}(N, f)$ for a search node N containing the soft flaw f, which represents p.

We evaluated two approaches called the *split strategy* and the *no-split strategy*. The general idea of the split strategy is to prevent the system from working on different disjuncts within the same preference. The no-split strategy, on the other hand, does allow to work on different disjuncts within the same plan.

4.1 The Split Strategy

The idea behind this strategy is to prevent the search process to work on different disjuncts in order to safe unnecessary search effort, as a preference in DNF is satisfied is a single disjuncts satisfied – protecting literals in other disjuncts is thus unnecessary search effort.

Let $p = \varphi_1 \vee \cdots \vee \varphi_n$ with $\varphi_i = \psi_{i_1} \wedge \cdots \wedge \psi_{i_m}$ be a preference that has never been addressed before, i.e., the corresponding soft flaw f has never been selected by the flaw selection function $\texttt{flawSel}$. Let $N = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P))$ and $f \in \texttt{flaws}_s(P)$.

When f is selected the first time, $\texttt{resolveFlaw}(N, f) = \{N', N_1, \ldots N_n\}$, where $N' = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P) \setminus \{f\})$ for the reason of completeness, and $N_i = (P_i, \texttt{flaws}_h(P), \texttt{flaws}_s(P))$, and P_i being P with p being set to φ_i.

When f is selected, but not for the first time, $\texttt{resolveFlaw}(N, f)$ produces all possibilities to protect an unprotected literal of the conjunction $\varphi_i = \psi_{i_1} \wedge \cdots \wedge \psi_{i_m}$. The literal to be protected is chosen by the flaw selection function defined in the next section. The flaw f is removed from the current plan under consideration as soon as every literal in φ_i is protected by a causal link.

4.2 The No-Split Strategy

As opposed to the last strategy, this one does allow to work different disjuncts within the same preference. While this strategy does allow for plans containing redundant plan steps and causal links, it shows advantages in terms of flexibility.

Let p be in DNF and f be the corresponding soft flaw. When we do not perform splitting into different conjunctions, $N' = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P) \setminus \{f\}) \in \texttt{resolveFlaw}(N, f)$ for $N = (P, \texttt{flaws}_h(P), \texttt{flaws}_s(P))$ and f being selected for the first time is not a sufficient criterion for completeness. Although it is still a necessary criterion, we also have to take into account p's disjunctive structure as follows: Let f and, in particular, the literal ψ_{i_j} be selected by the flaw selection function. Now, we distinguish two cases: Either some literal in the disjunct φ_i in which ψ_{i_j} occurs[1] was already selected before or ψ_{i_j} is the first literal selected in φ_i. In the former case, $\texttt{resolveFlaw}(N, f)$ contains all plans resulting from protecting ψ_{i_j}. In the latter case, this set additionally contains P but with p modified by φ_i set to false.

[1] For the sake of simplicity we assume that there is exactly one disjunct in p containing ψ_{i_j}. The described procedure is easily adapted to the general case.

Example Let $p = \psi_{1_1} \vee (\psi_{2_1} \wedge \psi_{2_2})$ be a preference in the solution plan P with none of its literals being protected by a causal link. Further, let f be the soft flaw representation of p. Let us assume that P can not be developed into a solution if ψ_{2_1} is protected by a causal link, but there are solutions if protecting ψ_{1_1}. Let us further assume the flaw selection function `flawSel` estimates ψ_{2_1} to be the most promising literal hence selecting f and, in particular, ψ_{2_1}. By assumption, none of the successor plans of P can be developed to a solution. However, we lose completeness, since there are solutions if protecting only ψ_{1_1}, but not protecting ψ_{2_1}. To solve that problem it is sufficient to allow ignoring disjuncts in the same way as we allow ignoring a complete preference: When addressing the disjunct $(\psi_{2_1} \wedge \psi_{2_2})$ the first time, we additionally add P' to the search fringe in which p is replaced by $p' = \psi_{1_1}$.

5 Flaw Selection Functions

In line 8 of Algorithm 1, the function `flawSel` selects a flaw from the current set of hard and soft flaws. In our experiments we observed the hard flaw selection strategy *Least-Cost Flaw Repair (LCFR)* [13] being one of the best performing strategies. It always selects a flaw f with a minimal number of refinement options $|\texttt{resolveFlaw}(N, f)|$. When selecting a soft flaw, we followed that idea and implemented a strategy selecting a soft flaw with a minimal estimated branching factor taking into account that each preference is a disjunction (possibly with exactly one disjunct if the split strategy is used) of conjuncts. We call the resulting strategies $LCFR_{DNF}$-$\sum$, $LCFR_{DNF}$-$\prod$, and $LCFR_{DNF}$-min, which estimate the branching factor based on the "cheapest" disjunct of a preference based on summarizing, multiplying or taking the minimal number of refinement options for each literal in that disjunct.

Let $p = \varphi_1 \vee \cdots \vee \varphi_n$ with $\varphi_i = \psi_{i_1} \wedge \cdots \wedge \psi_{i_m}$ and f the soft flaw representing p in plan P. For $\circ \in \{\sum, \prod, \min\}$, we define the preference-based Least-Cost Flaw Repair strategy as follows:

$$LCFR_{DNF}\text{-}\circ\,(N, f) := \min_{\varphi_i} \circ_{\psi_{i_j}} |\texttt{resolveFlaw}(N, f(\psi_{i_j}))|$$

where $f(\psi_{i_j})$ is the flaw representation of the literal ψ_{i_j} assuming only unprotected literals are taken into account.

Example $LCFR_{DNF}$-min always selects a soft flaw for which there is a literal with a minimal number of supporters. However, it completely ignores the size of the conjuncts. For example, let $p = \psi_{1_1} \vee (\psi_{2_1} \wedge \psi_{2_2})$ and $|\texttt{resolveFlaw}(N, f(\psi))|$ be $2, 1$, and 3 for $\psi := \psi_{1_1}, \psi_{2_1}$, and ψ_{2_2}, respectively. Then, $LCFR_{DNF}$-min $= 1$ selecting its *argmin* ψ_{2_1} ignoring that ψ_{2_2} also has to be supported afterwards in order to fulfill p. The other two strategies try to address that overly optimistic estimate of $LCFR_{DNF}$-min by taking into account *all* conjuncts. In the previous example, $LCFR_{DNF}$-$\circ = 2$ for $\circ \in \{\sum, \prod\}$ and the *argmin* ψ_{1_1}.

6 Evaluation

In this section, we evaluate the POCL algorithm using the presented strategies. Our evaluation focuses on three different questions:

- Does one of the two proposed soft-flaw-resolution techniques perform better than the other?
- In which order should hard and soft flaws be addressed, i.e., is it beneficial to work on preferences before a plan is a solution?
- Is there a soft flaw selection strategy that outperforms the others?

6.1 System Configuration

Since we want to maximize the net-benefit, we define our plan selection strategy `planSel` to prefer a plan P with maximal value of $\texttt{netBen}(\texttt{P}) - h(P)$, where $h(P)$ is a heuristic function estimating the action costs necessary to achieve all hard goals of the current partial plan P. To be more precise, h is a variant of the *additive heuristic for POCL planning*, as described by Younes and Simmons [25]. Our heuristic guidance is thus limited to action costs ignoring the benefit of satisfied preferences. Although we developed a heuristic for POCL which takes into account optional goals [4], we did not yet implement that heuristic.

We observed that our plan selection strategy `planSel` selects a *unique* plan in only 45% to 70% of all cases. To avoid a random plan selection among the plans with an identical value of $\texttt{netBen}(\texttt{P}) - h(P)$, we can define an arbitrarily long sequence of tie-breakers. In all experiments, the following sequence was used: *Maximize Benefit (MB)* $\rightarrow$ *Minimize Costs (MC)* $\rightarrow$ *Minimize # Ignored Preferences (MIP)*. MB maximizes the sum of the fulfilled preferences' benefit thus ignoring action costs, whereas MC minimizes these costs. *MIP* minimizes the number of ignored preferences to favor plans which still have the potential to fulfill more preferences. If a unique plan was still not found, a plan is picked at random.

The function `flawSel` consists of two parts: a sequence responsible for selecting a hard flaw and a sequence for selecting soft flaws, respectively. The hard flaw selection sequence $\texttt{flawSel}_h$ is always given by *Prefer Hard Flaw (PHF)* $\rightarrow$ *LCFR* $\rightarrow$ *Earliest Flaws (EF)* $\rightarrow$ *Select Hard Flaw (SHF)*. *PHF* always favors a hard flaw before a soft flaw. As a consequence, a soft flaw can only be selected by a subsequent flaw selection if the current plan has no more hard flaws. *LCFR* minimizes the branching factor as explained in the previous section. *EF* favors flaws which were introduced earlier to a plan and *SHF* finally ensures that some (random) hard flaw is selected if the choice is still invariant. For the selection of soft flaws, we use the flaw selection sequence $\texttt{flawSel}_s$ given by *Prefer Soft Flaw (PSF)* $\rightarrow$ $LCFR_{DNF}$-$\circ$ $\rightarrow$ *Select Soft Flaw (SHF)*. *PSF* and *SSF* behave exactly as *PHF* and *SHF*, but for soft instead of hard flaws. $LCFR_{DNF}$-$\circ$ is one of the three strategies as described in the last section.

In the empirical evaluation, we tested the system configuration with the `planSel` function as described above and all of the following combinations:

- For f being a soft flaw, $\texttt{resolveFlaw}(P, f)$ implements the *Split Strategy* or the *No-Split Strategy*.
- The flaw selection strategy $\texttt{flawSel}$ is one of the sequences $\texttt{flawSel}_{hs} := \texttt{flawSel}_h \rightarrow \texttt{flawSel}_s$ and $\texttt{flawSel}_{sh} := \texttt{flawSel}_s \rightarrow \texttt{flawSel}_h$

Note that there are actually six flaw selection sequences, as each soft flaw selection sequence contains the $LCFR_{DNF}$-$\circ$ strategy which is parametrized by $\circ \in \{\sum, \prod, \min\}$. In total, we thus evaluated twelve different configurations for every problem instance.

6.2 Benchmarks and Empirical Results

To compare our proposed strategies we evaluated two very different domains.

The first domain is a randomly generated domain to obtain a very large set of problem instances containing many preferences. The problems specify a goal description (which is not mandatory in preference-based planning) and 100 preferences consisting of 2 to 5 disjuncts, each of which being a conjunction of size 2 to 6.

The second domain is from an ongoing research project in which we want to assist a human user in every-day life situations like making appointments and going shopping. The domain is still rather small and only a few problem instances are modeled, yet. Each of them specifies certain mandatory goals as well as between 5 and 12 preferences in DNF.

Random Domain In the random domain, we evaluated 120 problem instances and report the net-benefit of the best plan found by each configuration within a search space limit of 10,000 plans.

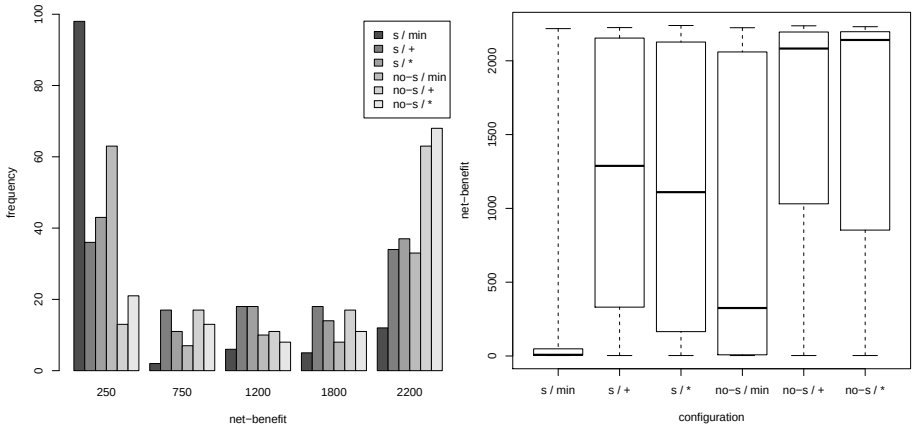
We visualize our results by means of a histogram (Fig. 1a) and a boxplot (Fig. 1b). Please note that we only include 6 of the 12 tested configurations in the diagrams because we observed one configuration parameter to cause the search to fail consistently: In only two problem instances our system was able to find a solution if the flaw selection sequence $\texttt{flawSel}_{sh}$ is chosen, in which soft flaws are resolved first. This perfectly meets our expectations, as it is more important to have any valid solution than to have a plan that satisfies many preferences but does not respect the solution criteria.

Comparing the remaining configurations, we clearly notice that using the split strategy in combination with $LCFR_{DNF}$-min is dominated by all other configurations (Fig. 1b). In almost 100 of all 120 instances it produced only plans with a net-benefit of up to 250, the mean being close to zero and 2200 being the maximal net-benefit achieved by any configuration/problem instance combination (Fig. 1a). This result is not surprising, since further investigation shows that both the split strategy as well as the $LCFR_{DNF}$-min strategy perform worse than their respective counterparts.

When we compare the different versions of the soft flaw selection strategy, the results clearly show that $LCFR_{DNF}$-min is dominated by $LCFR_{DNF}$-$\sum$ and $LCFR_{DNF}$-$\prod$ (Fig. 1b). While the latter two strategies mostly find solutions of

similar quality (Fig. 1a), the respective difference between these two strategies and $LCFR_{DNF}$-min is quite large. This observation is quite plausible as the $LCFR_{DNF}$-min strategy completely ignores the size of the conjunctions upon which it bases its decision.

The last remaining comparison is the one between the two flaw resolution strategies split and no-split. We observe that the no-split strategy clearly dominates the split strategy. We do not know the cause of that behavior and regard this result as the most interesting one. A possible cause of the advantage of the split strategy might be its flexibility to switch to some disjunct after the planner already worked on another. Consider the following example for clarification: Let P be a solution and $p = \psi_{1_1} \vee (\psi_{2_1} \wedge \psi_{2_2})$ one of its preferences. Further, let ψ_{2_2} be the literal selected by the flaw selection function. Let us assume protecting ψ_{2_2} introduced new hard flaws which need be to be resolved first before the planning system can continue working on p. Let P' be the resulting solution in which ψ_{1_1} and ψ_{2_1} are still unprotected. Assuming that no solution exists if ψ_{2_1} is protected by a causal link, the only possibility for the split strategy to protect ψ_{1_1} is to refine P which might take longer than refining P' or a plan along the path from P to P'.



(a) Histogram for net-benefit.

(b) Boxplot for mean value and distribution of net-benefit.

Fig. 1: Fig. 1a shows the impact of the chosen configurations on plan quality while Fig. 1b shows the distribution and mean value of the net-benefit of the best solution found by the different configurations. *s* stands for the split strategy, *no-s* for the no-split strategy and *min*, +, and $*$ stand for corresponding $LCFR_{DNF}$-$\circ$ soft flaw selection strategies, $\circ$ being min, $\sum$, and $\prod$, respectively.

User Assistance Domain Because our domain models and corresponding problem instances are quite small, optimal solutions can be found very fast. Thus, we set a timeout of only 1.5 minutes and report the best net-benefit of any solution and the current search space size when that solution was found. Tab. 1 shows the results obtained by two runs taking the mean values.

Table 1: This table shows the impact of the different configurations (rows) on different problem instances (columns) in the user assistance domain. `netBen` denotes the net benefit of the best found solution and SS the size of the search space when it was found. Configurations are named as in Fig. 1. hs stands for the flaw selection function $\texttt{flawSel}_{hs}$ and sh for $\texttt{flawSel}_{sh}$.

		#1		#2		#3		#4		#5	
		`netBen`	SS	`netBen`	SS	`netBen`	SS	`netBen`	SS	`netBen`	SS
s/min	hs	25	64194	10	786	7	184	2	5079	31	121
	sh	-42	1	0	1	7	294	0	1	0	1
s/+	hs	26	15908	20	658	12	272687	2	2223	31	125
	sh	-42	1	0	1	7	153	0	1	0	1
s/∗	hs	13	177	13	4705	12	418678	2	2223	31	123
	sh	-42	1	0	1	7	133	0	1	0	1
no-s/min	hs	28	1466	10	144	9	82561	2	1768	31	619
	sh	-42	1	0	1	7	76	0	1	15	285942
no-s/+	hs	17	134	20	442	7	94	2	1215	31	701
	sh	-42	1	0	1	7	100	0	1	0	1
no-s/*	hs	32	1776	15	243	7	82901	2	1215	31	116
	sh	-42	1	0	1	7	60	0	1	0	1

In the evaluation of the previous domain we were able to identify five very clear observations. The results of the user assistance domain do not reproduce these observations that clearly. In this domain, almost all configurations can find optimal solutions; there are only a few deviations which can not justify any conclusion.

However, our results do not *contradict* the previous observations as well, since all configuration comparisons result more or less in a tie. Furthermore, there is one observation that we can confirm very clearly. Preferring soft flaws before hard flaws (strategy $\texttt{flawSel}_{sh}$) is strongly dominated by the reverse ordering. In almost all cases the best quality is significantly worse compared to the opposite ordering. There is only one instance (cf. instance 3, last configuration), in which $\texttt{flawSel}_{sh}$ explores less nodes than $\texttt{flawSel}_{hs}$.

7 Conclusion & Future Work

In this paper, we introduced a POCL algorithm capable of solving problems with (simple) preferences while optimizing their net-benefit. For selecting and

satisfying a preference during planning, we developed three novel flaw selection functions, which are based on a successful strategy known from (standard) POCL planning without preferences. Furthermore, we addressed the question of *how* to address preferences when they are represented in terms of a disjunctive normal form. We evaluated these strategies empirically on two different domains.

Our empirical evaluation is still preliminary: So far, the plan selection strategy does not take into account the preferences; future work will thus include an empirical evaluation with a plan selection function guided by the preferences. Furthermore, we did not evaluate problems from the IPC which also remains future work.

Acknowledgements

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. Bahadori, S., Cesta, A., Iocchi, L., Leone, G., Nardi, D., Pecora, F., Rasconi, R., Scozzafava, L.: Towards ambient intelligence for the domestic care of the elderly. In: Remagnino, P., Foresti, G., Ellis, T. (eds.) Ambient Intelligence, pp. 15–38. Springer New York (2005)
2. Baier, J.A., Bacchus, F., McIlraith, S.A.: A heuristic search approach to planning with temporally extended preferences. Artificial Intelligence 173, 593–618 (2009)
3. Benton, J., Do, M., Kambhampati, S.: Anytime heuristic search for partial satisfaction planning. Artificial Intelligence 173, 562–592 (2009)
4. Bercher, P., Biundo, S.: A heuristic for hybrid planning with preferences. In: Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2012). pp. 120–123. AAAI Press (2012)
5. Bidot, J., Biundo, S.: Artificial intelligence planning for ambient environments. In: Heinroth, T., Minker, W. (eds.) Next Generation Intelligence Environments - Ambient Adaptive Systems, chap. 6, pp. 195–225. Springer, 1 edn. (2011)
6. 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
7. Cesta, A., Cortellessa, G., Pecora, F., Rasconi, R.: Supporting interaction in the robocare intelligent assistive environment. In: Proceedings of AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants. pp. 18–25. AAAI (2007)
8. Coles, A., Coles, A.: LPRPG-P: Relaxed plan heuristics for planning with preferences. In: Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011). pp. 26–33 (2011)
9. Edelkamp, S.: On the compilation of plan constraints and preferences. In: Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006). pp. 374–377. AAAI Press (2006)

10. Edelkamp, S., Kissmann, P.: Optimal symbolic planning with action costs and preferences. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 1690–1695. AAAI Press (July 2009)
11. Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. Artificial Intelligence 173(5-6), 619–668 (2009)
12. Gerevini, A., Schubert, L.K.: Accelerating partial-order planners: Some techniques for effective search control and pruning. Journal of Artificial Intelligence Research (JAIR) 5, 95–137 (1996)
13. Joslin, D., Pollack, M.E.: Least-cost flaw repair: A plan refinement strategy for partial-order planning. In: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994). pp. 1004–1009 (1994)
14. Keyder, E., Geffner, H.: Soft goals can be compiled away. Journal of Artificial Intelligence Research 36, 547–556 (2009)
15. Lin, N., Kuter, U., Sirin, E.: Web service composition with user preferences. In: ESWC'08: Proceedings of the 5th European Semantic Web Conference. pp. 629–643. Springer, Berlin, Heidelberg (2008)
16. McAllester, D., Rosenblitt, D.: Systematic nonlinear planning. In: Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI 1991). pp. 634–639 (1991)
17. Penberthy, J.S., Weld, D.S.: UCPOP: A sound, complete, partial order planner for ADL. In: Proceedings of the third International Conference on Knowledge Representation and Reasoning. pp. 103–114 (1992)
18. Pollack, M.E., Joslin, D., Paolucci, M.: Flaw selection strategies for partial-order planning. Journal of Artificial Intelligence Research (JAIR) 6, 223–262 (1997)
19. Pollack, M.: Planning technology for intelligent cognitive orthotics. In: Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS 2002). pp. 322–331 (2002)
20. Schattenberg, B., Bidot, J., Biundo, S.: On the construction and evaluation of flexible plan-refinement strategies. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) Advances in Artificial Intelligence, Proceedings of the 30th German Conference on Artificial Intelligence (KI 2007). Lecture Notes in Artificial Intelligence, vol. 4667, pp. 367–381. Springer, Osnabrck, Germany (September 2007)
21. Schattenberg, B., Bidot, J., Geßler, S., Biundo, S.: A framework for interactive hybrid planning. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) Advances in Artificial Intelligence, Proceedings of the 32nd German Conference on Artificial Intelligence (KI 2009). Lecture Notes in Artificial Intelligence, vol. 5803, pp. 17–24. Springer (September 2009)
22. Seegebarth, B., Müller, F., Schattenberg, B., Biundo, S.: Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012). pp. 225–233. AAAI Press (6 2012)
23. Smith, D.E.: Choosing objectives in over-subscription planning. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS 2004). pp. 393–401. AAAI Press (2004)
24. Sohrabi, S., Baier, J.A., McIlraith, S.A.: HTN planning with preferences. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). pp. 1790–1797. AAAI Press (July 2009)
25. Younes, H.L.S., Simmons, R.G.: VHPOP: Versatile heuristic partial order planner. Journal of Artificial Intelligence Research (JAIR) 20, 405–430 (2003)