Technical Report #280*: Addendum to 'Interval Based Relaxation Heuristics for Numeric Planning with Action Costs'

Johannes Aldinger and **Bernhard Nebel**
Albert-Ludwigs-Universität, Institut für Informatik
79110 Freiburg, Germany
{aldinger, nebel}@informatik.uni-freiburg.de

Abstract

This report contains two theorems and their proofs supplementing our work published at the 40th German Conference on Artificial Intelligence under the title 'Interval Based Relaxation Heuristics for Numeric Planning with Action Costs'.

Intro

In our paper on 'Interval Based Relaxation Heuristics for Numeric Planning with Action Costs' published at the 40th German Conference on Artificial Intelligence (Aldinger and Nebel 2017), we make two assertions which are supported by the theorems in this addendum: the first on insertions into a priority queue for acyclic numeric planning tasks and the second assertion on NP-completeness of the *global target value optimization problem* and of the generalized marking method of h_{FF} for numeric planning. In each of the two upcoming sections, we elaborate on one of these assertions. Both section starts with providing the background for the respective assertion, continue with a formalization of the theorem and conclude with a proof thereof.

Acyclicity and Priority Queue Insertions

In Section 3.3 of our paper (Aldinger and Nebel 2017) we discuss relaxation heuristics for numeric planning which use a *priority queue* based fact selection strategy in a *repetition relaxation*, an interval based relaxation where the semantics of actions captures arbitrarily many repetitions of each effect to the variables in an effort to make relaxed actions idempotent.

As numeric variables can attain infinitely many values, polynomial time computable heuristics have to restrict the number of considered variable-interval pairs (denoted as *facts*). One method is to only consider variable-interval pairs that a generalized Dijkstra algorithm would use. Facts are processed according to a *priority queue* storing the cost to achieve them. A change in the value of a variable v triggers all actions that have an effect *depending* on v, that is v appears in the assigned expression of one of the actions effects. If such an effect extends the current interval of v, then the corresponding fact is enqueued (mapping to the convex union of the old value of v and the value reached by the action's effect). The facts which are considered by the heuristic are then convex unions of the intervals in the queue with the current value of the variable at dequeue time. This additional use of the convex union is necessary to ensure monotonic growth of the intervals, as other actions can alter a variable while the new interval is still in the priority queue.

If actions are non-idempotent, such a *priority queue* based fact selection method is not sufficient to bound the number of facts. The motivation to use the repetition relaxation is to make actions idempotent. The repetition relaxation captures one source of non-idempotence: repeatedly applying the same effect to a variable. However, there are other sources of non-idempotence which come from the interaction of several variables. A variable v *depends* directly on another variable w if w appears in the assigned expression of a numeric effect of some action on v (Aldinger, Mattmüller, and Göbelbecker 2015). This dependency relation induces a *dependency graph*. If the *dependency graph* is acyclic, the repetition relaxation can be computed in polynomial time by evaluating variables according to the topology of that graph. This way, the intervals of topologically higher variables have already converged and changes in topologically lower layers can not influence the values of variables in topologically higher layers.

Unfortunately, for planning tasks with action costs, *priority queue* based approaches alter variables in an order of the cost to achieve new values, and this order does not necessarily respect the topology. As such, the number of required enqueue operations can become exponential in the number of variables. The problem occurs if variables achieved by cheap actions depend on many other variables achieved by more expensive actions which reside in topologically higher layers of the *dependency graph*.

Theorem 1. *The number of insertions of numeric facts into the priority can become super-polynomial even if the dependency graph is acyclic.*

Proof. Let $\mathcal{V}_N = \{v_0, \ldots, v_n\}$ be a set of variables and $\mathcal{A} = \{a_n\} \cup \{a_{ij} | 0 \leq i < j \leq n\}$ be a set of $\frac{n^2+n}{2} + 1$ actions with costs $\gamma(a_{ij}) = 2^i$, and $\gamma(a_n) = 2^n$. So there are n actions with first index $i = 0$ all costing $\gamma = 2^0 = 1$,

*University of Freiburg, Department of Computer Science
http://tr.informatik.uni-freiburg.de/2017/

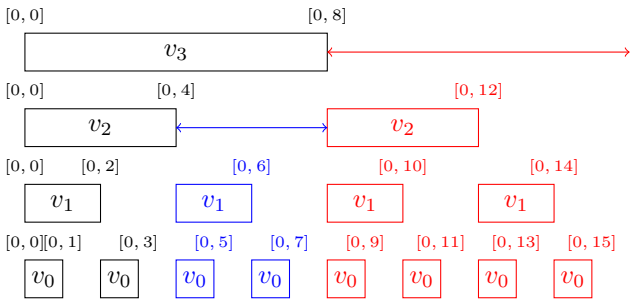

Figure 1: Enqueue and Dequeue times of variables for $n = 3$

$n - 1$ actions with first index $i = 1$ and cost $\gamma = 2$ and so forth.

Let each action $a_{ij} = \langle \text{pre}_{ij}, \text{eff}_{ij} \rangle$ have an empty precondition $\text{pre}_{ij} = \emptyset$ and a single assign effect $\text{eff}_{ij} = (v_i := v_j + 2^i)$ on variable v_i. The additional action a_n has an effect $\text{eff}_n = (v_n := 2^n)$. This way, each variable v_i has a *dependency* on all variables v_j with higher index: $v_i \prec v_j$ iff $i < j$. All variables are initialized to $v_i \mapsto [0, 0]$. The idea of this construction is to have exactly one fact be dequeued at each time step (cost) from 1 to $2^{n+1} - 1$. The lower bound of all intervals remains at 0 for all variables, and the upper bound of the altered variable is extended to a value coinciding to the current cost. Initially, all actions are applicable, and as such all of them enqueue their effect facts with their respective costs. Note that all actions a_{ij} with the same first index i have the same effect $v_i \mapsto [0, 2^i]$ as $v_j \mapsto [0, 0]$ for all j. Therefore, only one fact will be dequeued for each variable.

We determine the number of required enqueue operations enq inductively: action a_n has no dependencies and its effect $(v_n := 2^n)$ enqueues the fact $v_n \mapsto [0, 2^n]$ with cost $\gamma = 2^n$ once: $\text{enq}(v_n) = 1$. We will now show inductively that $\text{enq}(v_{i-1}) = 2 \cdot \text{enq}(v_i)$. Whenever a fact referring to a variable v_i is enqueued, so is a fact referring to v_{i-1} by construction. Whenever an action achieves a new value for higher-layer variable k with $i < k$, it will trigger action a_{ik} for v_i and $a_{(i-1)k}$ for v_{i-1}. E.g. consider the example depicted in Figure 1 for $n = 3$. Action a_3 sets v_3 to $[0, 8]$ and triggers a_{23} on v_2, a_{13} on v_1 and a_{03} on v_0 enabling the respective intervals to reach 12, 10 or 9 respectively. The second enqueue operation on v_{i-1} is triggered by v_i changing itself. Because of action $a_{(i-1)i}$, v_{i-1} is not only enqueued at the same time as v_i, but it is enqueued again whenever v_i is dequeued. The initial situation is slightly different in that there are n actions enqueued for each variable n instead of one. Save for the overhead of $\frac{n^2-n}{2}$ enqueue operations in the initially state, this already sets an upper bound of $\text{enq}(v_{i-1}) = 2 \cdot \text{enq}(v_i)$ enqueue operations for each variable. It remains to be shown that this upper bound is actually reached. Independent of k, a_{ik} costs twice as much as $a_{(i-1)k}$, and therefore, the effect triggered by v_k on v_{i-1} will be dequeued before the effect triggered on v_i. Therefore, all changes triggered at *enqueue* time are processed before v_i is dequeued. We still have to ensure that the facts triggered from v_i at *dequeue* time are processed before v_i is enqueued again. E.g. in Figure 1, for v_2 we have to ensure that the changes marked in blue fit into the space marked by the blue double-arrow or that the actions triggered by v_3 marked in red fit into the region marked by the red double-arrow. Changes in v_{i-1} trigger changes in v_{i-2} and so on. The execution of the sequence of actions $a_{(i-1)(i)}, \ldots, a_{01}$ costs $\sum_{k=0}^{i-1} 2^k = 2^i - 1$ which is cheaper than 2^i. Therefore, also the changes triggered at dequeue time are processed before the variable is reenqueued, resulting in twice as many enqueue operations for v_{i-1}.

The total number of enqueue operations is then $\sum_{i=0}^{n} \text{enq}(v_i) = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ plus the overhead of $\frac{n^2-n}{2}$ enqueue operations in the initially state which is exponential in the input size. □

Generalization of the Marking Method of h_{FF}: The Target Value Explication Problem

In Section 3.4–3.5 of our paper (Aldinger and Nebel 2017) we propose a generalization of the *marking* method of the h_{FF} heuristic (Hoffmann and Nebel 2001). In relaxed numeric planning, the progression step of the heuristic creates a sequence of relaxed states with the property that the first state contains degenerate intervals (the state for which the heuristic is computed), the intervals in the sequence are monotonically expanding for each variable (because of the convex union) and the last state satisfies the goal condition. For the planning graph approach, the state sequence corresponds to the layers of the planning graph, while for a priority queue based approach it corresponds to the sequence obtained from dequeuing new facts. Actions connect facts in this state sequence. Depending on the fact selection strategy these connections are not restricted to consecutive facts. The intervals of each fact are determined by the *convex union* of the fact interval from the previous state and all effect intervals from incoming actions modifying the respective variable. The cost of a fact is increased from one state to the next iff the corresponding interval is extended. An example of such a state sequence is depicted in Figure 2. Action a_1 has an effect $v_1 \mathrel{+}= 2$ and a_0 assigns $v_1 := v_0$. The fact for v_1 in s_2 is then the convex union of the interval $[0, 2]$ of v_1 in s_1 and the effect of action a_0: $v_1 \mapsto [-1, 0]$ resulting in $s_2(v_1) = [0, 2] \sqcup [-1, 0] = [-1, 2]$.

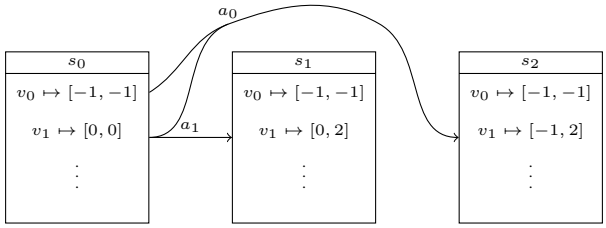

Figure 2: A state sequence produced by the progression phase

A *marking* of actions in such a sequence is a labeling procedure that assigns a label *marked* or *not marked* to each

action in each step. For the repetition relaxation, actions are marked with a number instead: the repetitions count. A simple *checking* procedure can verify whether such a marking corresponds to a *relaxed plan*: we progress the state sequence and assign new intervals to each variable where the interval for each fact is now restricted to the convex union of the interval from the previous layer and the effect intervals from actions that are both *marked* and *applicable* (where intervals are computed by repeating the actions effect according to the repetitions count if appropriate). If a state satisfying the goal condition is reached by this state sequence the marking is valid and corresponds to a relaxed plan. Note that the intervals in this new state sequence obtained from applying only marked actions are sub-intervals of the corresponding states obtained from the progression phase.

A marking is *optimal* if it is valid and the cost of the corresponding plan (that is the sum of the costs of all marked actions) is minimal among all valid markings.

Differences to the marking procedure from classical planning come from interval valued facts and actions which connect relaxed states over several layers. As numeric actions are non-idempotent, it is also not sufficient to apply actions only once and as such, actions can have to be marked more than once at different steps in the state sequence. For the repetition relaxation, also the marking label is generalized as actions are marked by assigning a repetition counter to them. Nevertheless, finding an optimal marking is still in NP.

The actions of an optimal relaxed plan might not be present in the structure generated by the progression method. Instead, we are interested in a marking so that the corresponding plan has minimal cost regarding the structure obtained from the progression search. In order to show that the optimal marking cannot be found in polynomial time, we consider the corresponding decision problem where a cost bound on the plan is given in advance.

Theorem 2. *Marking actions in an interval or repetition relaxed state sequence so that the corresponding interval relaxed plan is bounded by β is NP-complete.*

Proof. NP-hardness of the marking problem for numeric planning is evident it is a generalization of the h_{FF} marking problem for classical planning. Reducing minimum set cover to h^+: the cost of an optimal relaxed plan, can therefore also be used to show NP-hardness of the generalized numeric marking problem.

Membership in NP is shown by guessing a marking of actions. The plan cost bound β sets an upper bound on the number of actions that can be applied in total: β divided by the cost of the cheapest action $\lfloor \frac{\beta}{\min_{a \in \mathcal{A}} \gamma(a)} \rfloor$. The minimum exists as action costs are required to be positive. This number also ensures that the rational numbers on the interval bounds can be represented in polynomial space.

For the repetition relaxation we also have to guess a number of repetitions for each action. The plan cost bound β also bounds the number of repetitions that can be chosen at most for each action, namely $\lfloor \frac{\beta}{\gamma(a)} \rfloor$ for each action a.

The checking procedure that verifies that a given marking is indeed a relaxed plan is a simple progression scan that is restricted to the marked actions. During each phase, all marked actions are executed, unless they are not applicable in which case the marking is either suboptimal or not a valid relaxed plan. If the last state of the verification procedure satisfies the goal condition, the marked sequence is indeed a relaxed plan. The procedure runs in time polynomial in $\lfloor \frac{\beta}{\min_{a \in \mathcal{A}} \gamma(a)} \rfloor \times l$: the number of marked actions and the length of the sequence l, thus demonstrating membership in NP. □

Numeric constraints can often be satisfied by intervals from previous states in the sequence that can be achieved with lower cost. Also, for a state sequence obtained from a *planning graph* based fact selection strategy, several actions can affect the same fact, but not all of them extend the interval in the intended direction. Therefore, we suggest that marking a numeric fact includes choosing a *target value* in the respective interval, so that reaching the *target value* is sufficient to satisfy the desired constraint. If several actions rely on the same fact, several *target values* can be required for one fact. However, this number can never exceed two: one for increasing and one for decreasing the current value of a variable. If several *target values* are greater (less) than the current value, the greatest (smallest) among them should be kept, as the other *target values* will then be automatically reached by the convex union.

This opens the challenge to select suitable *target values*. *Local target value constraints* restrict the fact intervals to feasible sub-intervals for a *global target value optimization problem*. The *local target value constraints* determine feasible sub-intervals for all facts that appear in preconditions of numeric actions, so that the desired effect values can be reached. The *global target value optimization problem* is then the optimization problem that has to select target values within the feasible sub-intervals so that the corresponding relaxed plan has minimal cost.

Target values are of great practical help for a regressive marking procedure which has to select numeric facts that do not have to be achieved completely. Additionally, *target values* allow us to select an appropriate achieving action in case several actions affect the same fact such in a state sequence obtained from a *planning graph* based fact selection strategy.

Originally, we assumed that *target values* would not only be helpful to determine a marking regressively, but that it would also be beneficial to prove NP-completeness of the generalized marking method of h_{FF}. During the refinement of our proof sketch we realized it is more elegant to first prove NP-completeness of the generalization of the h_{FF} marking method (Theorem 2) and then show that the *global target value optimization problem* is equivalent (Theorem 3).

Given a *marking* of actions, determining facts and feasible *target values* comes for free: the interval bounds of the facts generated by the checking procedure can be used if the respective interval is extended from one step to the next. In case the application of an action extends both bounds of a certain variable, we select two *target values* instead.

Theorem 3. *The* global target value optimization problem *is NP-complete.*

Proof. We show that given a marking of actions we can determine *target values* for all numeric facts and vice versa. Thus, the *global target value optimization problem* is another formulation of Theorem 2 which has already be shown to be NP-complete.

$\Rightarrow$: Given a marking of actions in a relaxed state sequence, we determine a marking of facts and feasible *target values* as follows: For each marked action, we select the bounding element of the bound which was extended as optimal *target value* for the fact in the successor state. If the marked action can extend both bounds of the interval, we select two *target values* instead, one for each bounds. If the same fact is assigned target values from several actions, only the greater of the *target values* for extending the upper bound and the lower of the *target values* for extending the lower bound is used. In case a bound diverges to infinity (which can only happen by division by an interval containing zero), a sufficiently large number is used. Because of the convex union, all values in the fact intervals are also achieved and yield the same fact progression as the one obtained from applying all marked actions.

$\Leftarrow$: We can not only determine a *marking* of facts and *target values* from a marking of facts and actions, but also vice versa. Starting from the current state s_0, we proceed the state sequence and if a fact is assigned a target value which is not supported by the current interval, we mark the cheapest achieving action. □

Acknowledgments This work was supported by the DFG through grants EXC1086 (BrainLinks-BrainTools) and NE 623/13-2 (HYBRIS-2).

We thank Robert Mattmüller for reviewing our proofs.

References

[Aldinger and Nebel 2017] Aldinger, J., and Nebel, B. 2017. Interval Based Relaxation Heuristics for Numeric Planning with Action Costs. In *Proceedings of the 40th German Conference on Artificial Intelligence (KI 2017)*.

[Aldinger, Mattmüller, and Göbelbecker 2015] Aldinger, J.; Mattmüller, R.; and Göbelbecker, M. 2015. Complexity of Interval Relaxed Numeric Planning. In *Proceedings of the 38th German Conference on Artificial Intelligence (KI 2015)*.

[Hoffmann and Nebel 2001] Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research 14 (JAIR 2001)* 253–302.