Two Algorithms Suitable for Constructive Utility Planning

Wolfgang Nejdl and Jörg Bachmayer
RWTH Aachen
Informatik V
Ahornstr. 55, D-52056 Aachen, Germany
{nejdl,bachmayer}@informatik.rwth-aachen.de

Abstract

We discuss two algorithms suitable for constructive utility planning using both actions and observations, which are based on iterative plan modification instead of n-step plan generation. We have analysed runtime costs and solution quality both theoretically and on a set of examples. For most situations we analysed, our algorithms perform better than conventional plan generation algorithms using n-step look-ahead, yielding a family of algorithms which can be applied advantageously for constructive utility planning taking both actions and observations into account.

1 Introduction

This paper discusses algorithms presented preliminarily in [NB93] and presents both formal performance analysis and experimental results. The algorithms are part of an approach we call the *constructive utility approach*, which uses ideas from diagnosis, planning and decision theory to construct systems doing both diagnosis and repair.

In this paper we will concentrate on the basic algorithms used to compute diagnosis/repair plans including both observations and actions. The paper [NF93] describes the complementary concepts of utility, plausibility and criticality.

Diagnosis and Repair with purpose goals Let us first repeat the general diagnosis/repair process defined in [FGN92] and [FN92]. The goal of an extended *diagnosis and repair process* is to find and execute a general *diagnosis and repair plan* (DR-Plan). This plan usually integrates observations and actions to (re-) establish certain properties of the system which are represented by a set of goals representing the *system purpose*. Both, the non-achievement of goals (*breakdown costs*) as well as measurements and actions included in diagnosis repair plans (*diagnosis and repair costs*) to achieve these goals have associated costs. The plan should try to be optimal in the sense that it minimizes system breakdown costs as well as dignosis and repair cost.

Diagnosis and Repair with replacement goals In this paper we use no general purpose goals, but goals expressing component replacement to be able to compare it to [SW93], which does not know other repair goals. The algorithm in [SW93] is based on n-step lookahead. The common disadvantage of n-step lookahead methods is the so called horizon effect. The horizon effect causes often unexpected bad decisions, which increases the overall costs. Because of the full consideration of all planning goals during the stepwise DR-plan refinement, we claim that our algorithms avoid the horizon effect. A comparision of the cost evaluations of a specific example support this claim.

2 Basic Concepts

2.1 Context-dependent Costs

As observed also in [SW93], diagnosis and repair costs are often depending on the situation in which the observation is made. These context-dependent costs are accomodated in our framework by attaching a set of preconditions to each observation and repair action. Whenever (some of) the preconditions are not fulfilled in the current situation a sequence of actions has to be performed depending on the current situation. Following the diction in [SW93] we use the term *observation operations* for any actions that lead to new information about the actual state of affairs, while *repair operations* are used to change the actual state. In general, these *diagnostic operations* may consist

of probing goals or replacement goals and context-dependent actions necessary to achieve a state where probing or replacement is possible.

Simple action-costs are used and thus the overall cost of a diagnostic operation can be evaluated by simply summing up these costs. Taking state dependent costs into account (as also Sun and Weld do), costs change dynamically during the construction of a DR-plan. This state-dependence ist the reason, why constructed DR-plans are often complex and consist of several alternate repair and diagnosis phases.

2.2 Diagnosis and Repair Scheme

We have the following definition of an extended *diagnosis and repair algorithm scheme* as used in [FN92].

Algorithm Scheme 1 Characterization of *diagnosis/repair process*:

- Start with an initial set of critical worlds CW representing the inital uncertainity.
- Take the appropriate purpose $\mathcal{T}$ representing the desired functionality after the repair process.
- Loop forever
  - Generate DR-plans consisting of observation/action procedures, evaluate their utility.
  - *If* the best DR-Plan is empty (no advantageous observations/actions exist), *then* stop.
  - Execute the next procedure from the best DR-plan, changing CW to CW'.

2.3 DR-Plan Algorithms

Given that planning in itself is a rather difficult endeavor, generating and evaluating plans from uncertain initial conditions seems to be too complex in general (compare e.g. [CS91]). We therefore adopt the view advocated already in [FN92] and look only for solutions as good as possible rather than optimal ones. However, while retaining the architecture proposed in [FN92] for incrementally generating DR-Plans, we generalize it and the associated algorithms, leading to better plans albeit at higher computational costs.

We will discuss two basic variations of this family of algorithms, the first one centered on successively extending a GDE-DR-Plan and the second one centered on successively extending an R-Plan. A GDE-DR-Plan consists of only one observation phase distinguishing between all critical worlds and a subsequent repair phase, where the replacement of components is accomplished in each world. An R-Plan consists of a sequence of repair actions valid for each plausible world considered without including any measurement actions.

The structure of general DR-Plans is determined by iterating sequences of observation operations distinguishing between sets of worlds and sequences of action operations. As already discussed in [FGN92, FN92, NB93] this interleaving of diagnosis and repair phases can optimize the total cost of the diagnosis and repair process.

3 Computing Diagnosis/Repair-Plans

3.1 Starting from GDE-DR-Plans

3.1.1 Framework

As in [FN92], for each possible world our planner can generate a conventional repair plan containing only action procedures. The possible world is used as initial situation for the planner, a set of repair goals of a specific level as specification for the goal situation.

The planner uses a set of available repair actions, which have associated costs. Additionally, we have a set of possible observation procedures including associated pre-conditions and therefore context-dependent costs.

The structure of DR-Plans is characterized by the iteration of diagnosis and repair phases. We employ following concepts to limit computation complexity:

- The approximate computation of clusters, which are the main units distinguished during diagnosis phases. The main idea here is, that diagnosis phases do not necessarily discriminate between single worlds, but clusters of worlds. These clusters can be viewed as dynamically generated abstractions. The resulting structure of the DR-Plan (alternative phases of discrimination and repair) allows us to break up the planning process into many smaller tasks.
- The use of a bounded number of diagnosis/repair iterations.

The algorithms to construct DR-Plans are a generalization of the ones presented in [FN92], whose motivation was to develop the simplest algorithms possible while still giving adequate solutions. While still using the same framework of DR-Plans, the algorithms here hold a middle ground between solution quality and complexity.

Starting from GDE-Plans the following algorithm ONE2K generates general DR-Plans.

Algorithm 1 ONE2K (or GA if K=2)

- *start from initial-1-step-DR-Plan(GDE-Plan)*
- set current-DR-Plan to initial-1-step-DR-Plan
- loop forever
  - evaluate the merge of every pair of clusters by constructing the sub-DR-plan of every merged pair calling ONE2(K-1) and take the best one.(if $K = 0$, the variation of the DR-plan is constructed - call to the RDR-Plan Algorithm, see below)
  - *if* the best merge of two clusters leads to a better DR-plan
    - *then* set current DR-plan to the better one and loop again
    - *else* output the current DR-plan and exit loop.

This algorithm uses an ID3-style procedure to generate decision trees for discriminating the generated clusters in the diagnosis phase.

The big advantage of such a structure is that the optimality of the computation can be influenced by changing the approximative algorithms for computing diagnosis and repair phases while always taking the complete set of goals into account. This avoids the horizon effect encountered in many traditional planners (e.g. [SW93]) which can lead to neglecting some goals altogether.

As mentioned above, we alternate between diagnosis and repair phases in a DR-Plan. The diagnosis phase distinguishes between clusters of the current partitioning, the repair phase executes a subset of plans in the relevant cluster.

In each cluster, after having executed some repair operations, we again partition the remaining worlds and proceed to distinguish between these clusters, etc. As these DR-Plans are only estimates for an optimal plan and are re-computed after new information is obtained (by probe operations), we simplify their structure by using only a restricted number of diagnosis/repair cycles to make the whole transformation process tractable.

More detailed discussion of the used utility and plausibility concepts as well as cost functions can be found in [FN92] and [NF93].

Looking at our algorithms, the reason for distinguishing observations and actions becomes clear (apart from the conceptual clarity). Alternating observation and action phases gives DR-Plans a structure we can exploit. We are first using approximative algorithms to generate these abstract dynamic plans and fill in the details successively. This breaks up the planning process into many smaller steps, reducing the overall complexity, while avoiding the horizon effect, that each traditional planning strategy starting from the initial situation using a bound on the number of planning steps is going to suffer. A detailed discussuion can be found in [NB93].

3.2 Starting from R-Plans

In the previous section, we started from the GDE-DR plan, which was used as upper bound for cost minimization. While this has the advantage of never leading to a worse plan then a diagnosis systems, which does repair only after diagnosis, it has one disadvantage: Clusters can be generated, which cannot be distinguished by existing observation procedures. Though this case is handled by our algorithm, it still leads to some inefficiency.

To avoid this we have implemented another variation for constructing DR-Plans which use pure repair-plans as starting point. The costs of these pure repair-plans are the upper-bound of our new algorithm. We will discuss this variant in detail in the following section.

This approach is useful if the repair plans for different diagnosis are not incompatible with each other. This is the case, for example, when we concentrate only on replacement actions as repair actions and additional preconditions for these repair actions (such as location changes), which do not longer have to hold once the repair action has been accomplished (see also [SW93]).

Starting from a pure R-Plan, we can try to lower the costs of this plan by inserting an observation operation. The RDR-Plan Algorithm inserts an observation operation(D) into the given R-Plan in such a way that the overall costs of the resulting RDR-Plan are as low as possible.

So the general structure of a repair-plan with an inserted observation consists of three phases:

- 1. Repair-Phase: Consists of all repair operations before an observation is made.
- 2. Diagnosis-Phase: Consists of the observation procedure.
- 3. Repair-Phase: Consists of v repair plans, each executed with propability p. (v is the number of values observable by the observation procedure)

As mentioned above the ONE2K-Algorithm uses an entropy based ID3-style algorithm to estimate the diagnostic cost and to distinguish clusters.

In the algorithm starting from R-plans no entropy based evaluation of the diagnosis costs is necessary. All possible diagnosis procedures are explicity known

and they are explicity incooperated in the RDR-plans or DR-plans. Therefore the costs are evaluated without an extra calculation of the entropies and/or decision trees.

Algorithm 2 RDR-Plan-Iteration-Algorithm

- Start from a R-Plan
- Take an observation operation obs_i $(1 \leq i \leq m)$
  - For n_x R-operation in Phase 1 do (n_x times)
    - The R-operation of the first repair phase are tested for the execution in Phase 3. The overall best R-operation which yields the overall best RDR-Plan costs is choosen and shifted to Phase 3
- The Iteration generates a set of RDR-Plans. The overall best RDR-Plan of this set is applied for the DR-Plan-construction algorithm.

The replacement-subplan have to be inserted into one cluster which contains the corresponding world of the replacement subgoal (otherwise known correct components are replaced). For example, if the replacement goal g_k is shifted to phase 3 of the RDR-Plan, the goal is inserted in cluster obs_{ui} if $w_k \in obs_{ui}$ $(1 \leq k \leq n$, where n is the number of worlds).

R-Plan generation algorithm Note, that the R-operations of phase 1 and phase 3 corresponds to state-dependent subplans. The goals of this plans are the replacements of probably defect components. After every shifting operation several calls to a planning function are required, because all R-operations have to be replanned, which could have been affected by the shifting operation.

Under the assumption, that the R-operations R-plans are freely permutable, that is, no execution of an R-operator is a condition of the execution of another R-operator, a simple greedy algorithm can be designed, which returns a satisfying minimum cost solution:

1. Take an R-operation from the set of R-operations of the R-plan
2. Take the next R-operation and check, if the insertion of that Operation behind or before the last operation generates a cheaper R-Plan. The cheapest operation order will be used by further steps. (Notice, that actually goals are inserted and a planner evaluates the corresponding operators)
3. Execute this insertion procedure for every R-operation in the set generating all insertion variants to get the best one. Stop until all R-operations are inserted.

This algorithms needs $g * (g + 1)/2$ basic steps. g is the number of R-operations (corresponds to the number of plausible worlds if replacement-goals are used).

Algorithm 3 DR-Plan construction algorithm, first variant

1. Start from a pure R-Plan as upper bound. No world is distinguished by an observation. All worlds are in one cluster.
2. Calculate for all observations(obs_i,$1 \leq i \leq m$), which have to be applied on a cluster, the corresponding best RDR-Plan (call to the RDR-Plan-Iteration-Algorithm). Determine the new DR-plan costs and take the corresponding best observation as new insertion in the existing DR-Plan.
3. If this operation is prefered to all other according to the cost function, the result are v new clusters. Each of these can be refined by going to step 2.
4. Stop and return the DR-plan

This algorithm has $O(n) \times O(m)$ basic steps (calls to the RDR-Plan-Iteration-Algorithm), where n is the number of plausible worlds in PW and m is the number of observations. Note, that the first repair-diagnosis steps of the returned DR-plan are generated by the RDR-Plan algorithm and thus they are choosen very locally.

To improve their quality, the basic version of the algorithm can be extended in various ways to achieve better estimates of the DR-plan-costs. Notice, that only the RD-part of the RDR-plan is used for the construction of the DR-plan. The remaining R-part may be changed by further RDR-plan constructions.

A variant of the algorithm is the application of this basic version on the repair cluster for every first observation. This means, that for every possible first observation an entire DR-plan is constructed and overall costs are calculated. The best DR-plan with the corresponding best observations is then choosen as the best estimate (DR-Plan construction algorithm 2).

This increases the complexity and we have $O(n) \times O(m^2)$ basic steps. We will use this kind of algorithm for our comparisons. So far it has shown the best trade-off between quality of results and complexity in our examples.

Algorithm 4 DR-Plan construction algorithm, second variant (RA)

1. Start from a pure R-Plan as upper bound. No world is distinguished by an observation. All worlds are in one cluster.
2. Calculate for all observations, which have to be applied on the start-cluster, the corresponding DR-Plans with DR-Plan construction algorithm 1. Determine the new DR-plan costs and take the best DR-plan.
3. Stop and return the best DR-Plan.

3.3 Comparison of Iteration Planning and N-Step Look Ahead Planning - Experimental Results

We have applied our DR-Plan construction algorithm 2 and the ONE2K-algorithm on a parameterizable modified STRIPS-example ([FN71]). A robot is able to move in different rooms, to push boxes, to climb on boxes to repair a wire part and so on (see figure 1). In the different rooms a network is installed, which connect sockets(1 - 7). Certain wire parts(A - G) can be faulty. The task of the robot is to repair the faulty network as cost effectively as possible. All actions, which the robot can perform, cause costs.

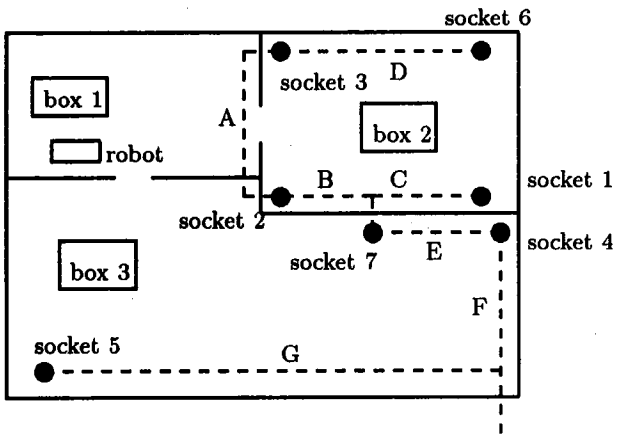

Figure 1: Robot example with $n = 7$ plausible worlds and $m = 7$ probing points

In the example m probe points (sockets) are used to discriminate the worlds. These probe points correspond to goals of observation-operators. The goals of the repair operators are the replacements of the n wire components. We sum up the action costs of the diagnostic operators and yield state dependent operator costs. The planning of the diagnostic operations(repair and observation) is performed by the linear planner WARPLAN of [War74]. To repair a faulty component a sequence of diagnostic operations has to be executed. The diagnostic operations in this sequence are determined by repeated calls to the N-step lookahead and DR-Plan algorithm. After every evaluation of a DR-Plan the front operator is executed, new information is gained, and the DR-plan algorithm is called again. In this manner execution traces are evaluated and performed until the faulty component is replaced. We have applied our algorithm on the robot example and have generated execution traces for several variantes. To compare the achieved costs of the execution traces of the different methods, we have generated all possible execution traces and multiplied the corresponding costs by the faulty probability of the wrong component(see table 1). The sum of these weighted costs is a measure of the goodness of the results. The meaning of the columns of the table 1:

- n: Number of worlds .
- m: Number of observations.
- start: Room in which the robot starts.
- Obs: "A"and "B" represent different types of the installation network, which affect the discrimination capability of the observations.
- InC: Are the costs of certain actions increased if these actions are repeated in successive diagnostic operators?
- N=i: N-step-lookahead Algorithm with i steps lookahead.
- RA: Costs of DR-Plan Construction Algorithm 2 starting from a R-Plan.
- GA: Costs of DR-Plan Algorithm ONE2K starting from a GDE-Plan.

If the number of worlds is small ($n \leq 6$) then the space of all possible execution traces is also small. The consequence is that our algorithms do not get better results because 3-step lookahead is exhaustive ($n = 4$ but not $n = 5$). Our algorithms are superior in most cases if the average length of the execution trace is increased. Specially the above results show that changing action costs (column $InC = yes$) cause N-step look ahead to compute worse results than GA and RA in almost all cases.

Additionally we compared the different runtimes and included a table of relative time measures (see table 2). Both tables include results for n between 4 and 9 and for m between 4 and 7. For the parameters tested, the performance of the algorithm RA lies between 3-and 4-step look ahead. For larger n its relative performance slowly gets worse (up to 6- or 7-step look ahead). However, as the usual number of

n	m	*start*	*Obs*	*InC*	$N=1$	$N=2$	$N=3$	*RA*	*GA*	$N=4$
4	4	room(1)	A	yes	601	585	559	559	559	559
4	4	room(1)	A	no	511	503	495	495	495	495
5	5	room(1)	A	yes	716	694	652	628	628	628
5	5	room(1)	A	no	589	558	543	536	543	536
6	5	room(1)	A	yes	706	727	679	679	679	679
6	5	room(1)	A	no	589	530	530	530	530	530
7	6	room(1)	B	yes	830	1107	775	725	761	761
7	6	room(1)	B	no	589	573	568	566	568	568
7	6	room(1)	A	no	581	539	560	539	539	539
7	6	room(3)	A	no	536	497	497	497	497	497
7	7	room(1)	B	yes	830	1107	775	725	761	761
7	7	room(1)	B	no	589	573	568	566	568	568
7	7	room(1)	A	no	591	539	572	539	539	539
7	7	room(3)	A	no	536	497	497	497	497	497
8	6	room(1)	B	yes	920	1258	861	833	833	833
8	6	room(1)	B	no	718	650	664	659	649	660
8	6	room(1)	A	no	597	642	589	581	581	581
8	6	room(3)	A	no	609	654	601	593	617	593
8	7	room(3)	B	yes	933	862	818	780	780	780
8	7	room(3)	B	no	617	608	608	608	608	613
8	7	room(3)	A	no	623	655	620	620	620	629
8	7	room(1)	A	no	611	643	608	608	608	608
9	7	room(3)	A	yes	891	878	875	875	875	*
9	7	room(3)	A	no	677	709	674	674	674	*

Table 1: Average costs of all possible execution traces.

plausible worlds will in most cases not be greater than 8 to 10, worst case results for very large n are not relevant for practical applications. The performance of ONE2K - Algorithm (GA) is not worse than 3-step look ahead for almost all parameter configurations and better in many cases. The theoretical preformance results presented in [NB93] correspond to the experimental results of table 2.

4 Conclusion

We have presented a family of algorithms to compute diagnosis/repair plans consisting of both observation (diagnosis) and action (repair) sequences. These algorithms use a new concept we call *iteration* planning to handle the complexity of this task, as they are based on distinguishing and iterating between diagnosis and repair phases. We described two basic variants in more detail, the first one starting from a GDE-DR-Plan consisting of one observation and one repair phase (separate for each world), and the second one starting from a R-Plan consisting of only repair actions (valid for all worlds considered).
We have implemented a replacement version of both variants, which solve exactly the same problem as the n-step look ahead planning algorithm described by Sun and Weld [SW93]. It is thus suitable for detailed comparison with this more conventional approach. We have analysed its performance both theoretically and experimentally on a robot diagnosis/repair example with varying parameters. Theoretical analysis and experimental results show our algorithms to be superior in performance to a 4-step resp. 3-step look ahead planning algorithm a la Sun and Weld, while yielding better (lower cost) plans in many situations. We expect still better results when we take breakdown costs into account, which we described in [FN92] ([SW93] is not able to include such time dependent breakdown costs).

0* No results because the evaluation takes too long

References

[CS91] Lonnie Chrisman and Reid Simmons. Sensible planning: Focusing perceptual attention. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 756–761, Los Angeles, July 1991. Morgan Kaufmann Publishers, Inc.

n	m	*start*	*Obs*	*InC*	$N=1$	$N=2$	$N=3$	*RA*	*GA*	$N=4$
4	4	room(1)	(A)	yes	1	9	20	19	22	39
4	4	room(1)	(A)	no	1	8	23	19	28	43
5	5	room(1)	(A)	yes	1	12	35	71	45	118
5	5	room(1)	(A)	no	1	10	40	38	46	134
6	5	room(1)	A	yes	1	10	56	140	38	268
6	5	room(1)	A	no	1	9	59	85	33	270
7	6	room(1)	B	yes	1	23	64	154	114	404
7	6	room(1)	B	no	1	11	90	189	54	588
7	6	room(1)	A	no	1	10	84	187	69	536
7	6	room(3)	A	no	1	11	91	176	64	567
7	7	room(1)	B	yes	1	22	63	151	112	396
7	7	room(1)	B	no	1	10	84	186	54	578
7	7	room(1)	A	no	1	10	82	155	65	501
7	7	room(3)	A	no	1	11	92	176	65	584
8	6	room(1)	B	yes	1	26	76	169	105	561
8	6	room(1)	B	no	1	8	106	147	64	756
8	6	room(1)	A	no	1	13	118	348	72	894
8	6	room(3)	A	no	1	13	116	252	67	937
8	7	room(3)	B	yes	1	13	78	184	40	598
8	7	room(3)	B	no	1	13	118	229	110	908
8	7	room(3)	A	no	1	12	122	351	155	957
8	7	room(1)	A	no	1	12	124	360	107	912
9	7	room(3)	A	yes	1	13	152	455	119	*
9	7	room(3)	A	no	1	13	161	820	121	*

Table 2: Average relativ runtimes of all possible execution traces.

[FGN92] Gerhard Friedrich, Georg Gottlob, and Wolfgang Nejdl. Formalizing the repair process. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 709–713, Vienna, August 1992. John Wiley & Sons, Chichester. Also appeared in the Proceedings of the Second International Workshop on Principles of Diagnosis, Milano, 1991.

[FN71] R. E. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving in problem solving. *Artificial Intelligence*, 2, 1971.

[FN92] Gerhard Friedrich and Wolfgang Nejdl. Choosing observations and actions in model-based diagnosis-repair systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 489–498, Cambridge, MA, October 1992. Morgan Kaufmann Publishers, Inc.

[NB93] Wolfgang Nejdl and Jörg Bachmayer. Diagnosis and repair iteration planning versus n-step look ahead planning. In *International Workshop on Principles of Diagnosis*, Aberystwyth, Wales, September 1993.

[NF93] Wolfgang Nejdl and Gerhard Friedrich. Constructive utility in model-based diagnosis/repair systems. In *International Workshop on Principles of Diagnosis*, Aberystwyth, Wales, September 1993.

[SW93] Ying Sun and Daniel S. Weld. A framework for model-based repair. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 182–187, Washington, July 1993. AAAI Press/MIT Press.

[War74] D. Warren. WARPLAN, a system for generating plans. Memo 76, University of Edinburgh, Department of Computational Logic, 1974.