Local (Human-Centered) Replanning in the SIADEX Framework*

Marc de la Asunción, Luis Castillo, Juan Fernández-Olivares, Oscar García-Pérez, Antonio González, Francisco Palao

Dpt. Ciencias de la Computación e I.A.
ETSI Informática
18071 Granada, SPAIN
siadexwww@decsai.ugr.es

Abstract. This paper presents the replanning capabilities in the SIADEX planning framework[6], which are based in the chronological order of generation of actions during the planning stage. This is a local replanning strategy since the replanning of an action only affects to older actions in the search tree and newer actions (actions that were generated after the replanned action) are maintained unaltered. This process is intended to regenerate the plan taking into account the the chronological steps given to obtain the current plan in a mixed initiative approach with the collaboration of a human expert.

1 Introduction

Planning for real world problems is becoming a need to bridge the gap between theory and practice in the AI Planning community. However this task is full of difficulties that arise from many different sources but one of the most important ones is the uncertainty of the knowledge being used to obtain plans. In this sense, the available knowledge is usually faulty so, there may be knowledge that is only partially known (incompleteness), uncertain knowledge about the effects of an action in its environment (nondeterminism) or knowledge that may not be perfectly known, mainly different types of metric knowledge (imprecision).

There are only two approaches to overcome this uncertainty: a preventive one and a palliative one. Preventive approaches for handling uncertain knowledge in AI planning frameworks try to foresee this uncertainty during the planning process and before the execution of the final plan. Therefore one may find different approaches that build alternative, conditional, branches to foresee any possible contingency during the execution of a plan [7, 11, 3], approaches that take into account the probability of every possible outcome of an action and build different strategies to react upon the detection of unexpected effects [2] or approaches that bound the imprecision of part of the knowledge and obtain plans adapted to these boundaries [8, 5]. These approaches might be called "off-line" approaches since they separate the stages of planning and execution as different, sequential phases of the resolution of a problem like a batch process.

* This work has been partially supported by the Spanish MCyT under project TIC2001-4936-E

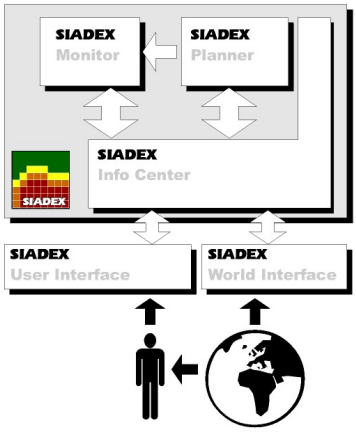
On the other hand, palliative approaches make a simplifying assumption that everything will go as expected so plans are obtained deterministically and, in the case that something could go wrong during the execution, the planner might be invoked again to design a subplan for the new contingency. Therefore, these approaches might be called "online" approaches since they may interleave several planning and execution stages during the resolution of the same problem. Basically one may find approaches for continual planning [9, 10], that interleave the design and execution of pieces of a plan until the complete problem is solved, or replanning approaches [13, 1] that redesign and replace a piece of a plan after a failure has been detected. In these approaches there are three main issues to be solved. The first one is detecting the source of the failure, the second one is delimiting the impact of the failure and the third one is redesigning the part of the plan that has failed.

The framework presented in this paper follows this last replanning approach and gives a solution for the second and third issues but taking into account that the planner is continuously being monitored and validated by a human operator. In this sense, the replanning capabilities of SIADEX must be human centered, that is, they only produce a local redesign of the part of the plan that has failed maintaining the remaining of the plan unaltered. Global replanning is only carried out if the impact of the failure is very deep, and when even the goal of the plan could be put in danger. This replanning strategy is based on the chronological order of generation of actions and it affects either older actions or newer actions that causally depend on the part that has failed. The remaining actions in the plan are maintained unaltered.

2 The SIADEX framework

SIADEX is a planning framework that is being developed under a research contract with the Andalusian Regional Government (Regional Ministry of Environment) [6]. It is intended to assist technical staff in the design of forest fire fighting plans but the ideas presented in this paper are domain independent and they might be applied to any mixed initiative replanning framework. The core of the SIADEX architecture, Figure 1.a), is a generative planner that obtains extended plans following a classical partial order causal link based planning algorithm [12] outlined in Figure 1.b). This algorithm will be explained later. These plans are then executed under the supervision of the monitoring module and a human operator. This is coordinated by the InfoCenter module that interfaces all of the interactions between SIADEX and the outer world:

- Receives planning requests by the user (through the User Interface).
- Ask the planner for new plans.
- Upon notification of the monitoring module, it launches execution orders to the human operator (through the User Interface) or to external agents (through the World Interface)
- Gathers information on the execution of the plan by indirect observation of a human operator (through the User Interface) or by direct gathering (through the World Interface) and translates them into the Monitor.
- Raises alerts about possible execution failures upon notification of the Monitor or upon direct user request.



(a)

SIADEX(Π,$\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$)

1. **If** $\mathcal{A}$ is empty **Return** Π,$\mathcal{O}$, $\mathcal{L}$
2. **Goal Selection**: Extract (l, a, t) from $\mathcal{A}$
3. **Action Selection**: Select an action a' to solve (l, a, t) such that either
   (a) $a' \in \Pi$ (Reuse an action)
   (b) $a' \in \mathcal{D}$ (Insert a new action)
   (c) **Return FAIL**
4. Update
   (a) $\mathcal{L} = \mathcal{L} \cup (a', l, a, NOW())$
   (b) $\mathcal{O} = \mathcal{O} \cup a' < a$
   (c) $\Pi = \Pi \cup a'$
   (d) $\forall l \in Preconditions(a'), \mathcal{A} = \mathcal{A} \cup (l, a', \infty)$
5. Solve causal interferences: if an action a^t threatens a causal link (a^p, l, a^c, t) then either
   (a) $\mathcal{O} = \mathcal{O} \cup a^t < a^p$
   (b) $\mathcal{O} = \mathcal{O} \cup a^c < a^t$
   (c) **Return FAIL**
6. Recursively call **SIADEX(Π,$\mathcal{O}$, $\mathcal{L}$, $\mathcal{A}$, $\mathcal{D}$)**

(b)

Fig. 1. The architecture of SIADEX and the generation of chronologically extended plans. Π stands for a set of actions, i.e., the plan, $\mathcal{O}$ is an order relation over Π, $\mathcal{L}$ is the plan rationale, $\mathcal{A}$ is the agenda of pending subgoals to be solved, and $\mathcal{D}$ is the set of actions schemas in the problem domain

As may be seen this architecture assumes that the execution of a previously designed plan in a dynamic world might not go as expected so it includes two different levels of response to this dynamicity. The first level is the monitoring module, that allows to trace the execution of the plan, detect possible failures and determine the possible impact of the failure. This is explained in sections 2.1 and 3. After the failure of an action has been detected, a mixed-initiative replanning episode is launched. The interaction is coordinated through InfoCenter and it may be as simple as re-executing the failed action (if this is feasible taking into account the temporal constraints in the problem) or as complex as deleting the remaining plan and redesigning a new one. This is explained in section 5. In any case, the new plan is monitored again in a continuous loop that ends with the successful execution of a plan.

2.1 Generation of chronologically extended plans

These extended plans obtained in SIADEX (Figure 1.b) contain a temporally ordered sequence of actions Π and the plan rationale $\mathcal{L}$. The plan rationale records the cause-effect relationships between actions in the plan as well as the chronological point in the resolution process at which every relationship was included. Thus, every record in $\mathcal{L}$ is a tuple (a^p, l, a^c, t) that means that at time stamp t, action $a^p \in \Pi$ was used to solve

the precondition l of action $a^c \in \Pi$. These records are named *causal links*. Taking into account that the planning algorithm follows a backwards search process, the age of an action $a \in \Pi$ in the plan, that is, the chronological point at which the action was included is given by $\tau(a) = \min_{(a,l,a',t)\in\mathcal{L}} t$.

The agenda $\mathcal{A}$ that contains the pending subgoals to be solved is also annotated with time stamps. A goal (l, a, t) in the agenda is a precondition l of an action a that is to be solved. If $t = \infty$ then the goal is being newly generated, otherwise $t \neq \infty$ means that it is a goal being replanned and it was originally solved at time stamp t.

For monitoring purposes only the plan Π, the temporal constraints defined over the actions $\mathcal{O}$ and the plan rationale $\mathcal{L}$ are needed. The time stamp t in every record of the plan rationale is only used for replanning.

3 Monitoring in dynamic environments

The monitoring module of SIADEX is based on a temporal scheduling and rescheduling policy over temporal plans [5] so that actions in a plan are continuously being selected for execution following the best temporal ordering, their execution is monitored and possible faults are detected (Figure 2). When a fault is detected its impact is calculated, and a replanning episode starts in which the user may interact with SIADEX, making some suggestions (delete or add actions by hand, add or delete goals or literals). These suggestions are processed by the planning algorithm and new interactions are requested to the user until a valid plan is obtained, that is scheduled again for its execution and monitoring.

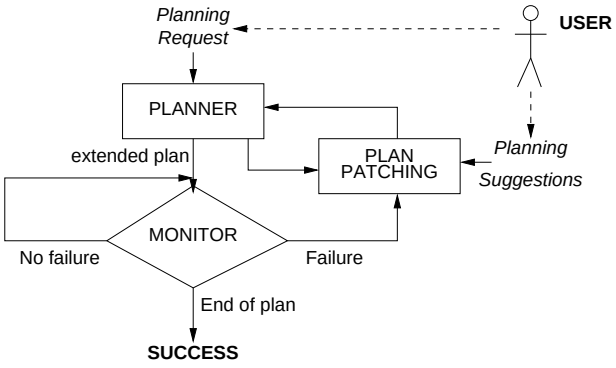


Fig. 2. The mixed-initiative planning and replanning loop in SIADEX

In order to correctly monitor the execution of a plan, SIADEX is based on two main constructs, the *Currently Known Horizon* (CKH) and the *Current Planning Horizon* (CPH). CKH Is the current state of the world since the initial state of the problem along actions have been completely executed. Initially CKH is the initial state of the

problem. CPH is a prediction about the next changes to come in the near future. The monitoring procedure is shown in Figure 3.

1. Let us consider
   - $\mathcal{X}$ The set of actions already executed
   - Π The set of actions that are to be executed
   - $\mathcal{Q}$ The queue of actions that are currently under execution
2. Initially $X = \emptyset$, $Q = \emptyset$ and $\Pi =$ the plan, $CKH = Initial - State$, $CPH = \emptyset$
3. Loop
   (a) Extract from Π the next actions to be executed and insert them into Q
   (b) Insert all of their effects in CPH
   (c) If a literal in CPH has been achieved in the real world, move it from CPH to CKH
   (d) If an action has achieved all its effects, move it from $\mathcal{Q}$ to $\mathcal{X}$
   (e) If an action in $\mathcal{Q}$ cannot achieve some of its effects or a literal in CKH has been deleted then return FAILURE
   (f) If $\Pi = \mathcal{Q} = \emptyset$ then return SUCCESS

Fig. 3. The monitoring steps

Initially, CPH is empty and every time a action is launched for its execution, all its effects are included in CPH. Every effect takes a different time to be achieved [4, 5] so the execution is continuously monitored and, once an effect has been achieved in the real world, it is moved from CPH to CKH.

If everything goes well, all of the actions are correctly executed, CKH contains the description of a goal state and CPH is empty. However, a plan may fail at some point in its execution. The source of this failure may be either a missing condition, i.e., a literal that suddenly dissapears from CKH, or a missing effect, that is a literal that cannot be removed from CPH. A missing condition may be due to an external event that deletes a previously achieved effect or an initial condition. A missing effect may be due to the failure of an action (or the agent that executes it) so that part of its effect cannot be achieved.

Both cases produce a missing literal that, taking into account the plan rationale $\mathcal{L}$, might invalidate the causal dependencies of a set of actions. So, after the detection of the failure, the next step is determining its impact in the causal structure of the plan. This is done by function **DeleteLiteral(l)** shown in Figure 4. Additionally, during the monitoring of the plan, new unexpected literals might arise. In these cases function **AddLiteral(l)** is easily used.

Function **DeleteLiteral(l)** carries out three main activities. On the one hand, it start to regenerate the part of the plan rationale that had been altered by the deletion of l. In order to do so, it includes in the agenda $\mathcal{A}$ a new replanning subgoal for every altered causal link. These subgoals are not new subgoals but subgoals to be re-satisfied. Therefore they will have a time stamp $t \neq \infty$. Secondly, every action with a missing precondition due to the deletion of l is labeled as $OPEN$. These actions cannot be executed without re-satisfying the missing precondition. And finally, every action that

DeleteLiteral(l)	**AddLiteral(l)**
1. If $\nexists(a, l, a', t) \in \mathcal{L}$ do nothing 2. Otherwise $\forall(a, l, a', t) \in \mathcal{L}$ do (a) Remove (a, l, a', t) from $\mathcal{L}$ (b) Add (l, a', t) to $\mathcal{A}$ (c) Label a' as $OPEN$ (d) $\forall(a', l', a'') \in \mathcal{L}$ i. Label a'' as $UNSTABLE$	1. $CKH = CKH \cup l$

Fig. 4. Two functions that may be used after the detection of a failure.

depends directly or indirectly on an $OPEN$ action is labeled as $UNSTABLE$ meaning that their preconditions are fully satisfied but that there may be some supporting action with a missing precondition.

4 User centered plan patching

This labeling of the action in the plan determines the future impact of the missing condition detected by the monitoring module. The decision to be taken is not easy, and it is left to the judgment of the human operator since it may be a very domain dependent decision. The most conservative decision is to re-execute the failed action. The most aggressive decision is to purge the plan, that is, the deletion of all $OPEN$ actions and all $UNSTABLE$ actions and replan them completely. All of these possibilities may be carried out through InfoCenter in a plan patching stage that is driven under the control of a human operator (Figure 2). This transition in the loop of planning and replanning may be carried out in two different ways (depending on the problem domain).

- The execution of the plan is fully stopped. Then CPH and CKH are immediately stopped until the plan patching ends and the (possibly new) plan may continue its execution. This corresponds with a batch replanning system.
- The execution of the plan is not stopped and every action that were not altered by the failure is continued. Then CPH and CKH continue changing while there is any executable action not labeled as $OPEN$ either as $UNSTABLE$. This schema corresponds with an asincronous replanning system and requires a very subtle synchronization since CKH and CPH might change or new failures might occur.

In any case, these are the suggestions that may be made by the human operator.

DeleteAction(a) This suggestion deletes the desired action a (only if it has not been executed yet), it deletes all of its effects in the plan, and recursively deletes all of the actions that were included explicitly to solve any precondition of a[1]. So this suggestion deletes the causal structure supporting a that is no longer needed,

[1] These are actions newer than a in the chronological order that solves a precondition of a. Actions older than a are not deleted since they were introduced for other purpose and they were later reused by a during the planning process.

labels as $UNSTABLE$ the actions that causally depend on a, and generates some replanning goals.

DeleteAction(a)

1. Only if $a \in \mathcal{Q} \cup \Pi$
2. Remove a from $\mathcal{Q}$ or Π
3. $\forall l \in effects(a)$ **DeleteLiteral(**l**)**
4. $\forall (a', l, a, t) \in \mathcal{L}$
   (a) Remove (a', l, a, t) from $\mathcal{L}$
   (b) If $a' \in \mathcal{Q} \cup \Pi$
      i. If $\tau(a') > \tau(a)$ **DeleteAction(**a'**)**

AddAction(a) This suggestion creates a set $\mathcal{P}$ of patches of the user that contains actions explicitly added by the user. These actions will be eventually reused later by the planner during the replanning stage.

AddAction(a)

1. $\mathcal{P} = \mathcal{P} \cup a$

DeleteGoal(g) This suggestion is very strong. It deletes all of the actions that were exclusively added to solve that goal and generate a replanning subgoal and, possibly, many replanning subgoals.

DeleteGoal(g)

1. Let (a, g, END, t) the causal link that records the satisfaction of the goal g by means of the action a.
2. **DeleteAction(a)**
3. Insert (g, END, t) in $\mathcal{A}$

Addgoal(g) This action simply adds a new primary goal to the agenda.

AddGoal(g)

1. $\mathcal{A} = \mathcal{A} \cup (g, END, \infty)$

5 Local replanning in SIADEX

After the plan patching stage, the plan contains a damaged causal structure, with many open conditions still to be re-solved, some actions are new and others have dissapeared. The replanning episode has to regenerate the missing pieces of the plan to obtain a complete, valid plan. In order to do this, instead of replanning completely the missing actions, a local replanning procedure is carried out taking into account the time stamps

in the newly generated replanning subgoals. This is a very close procedure to a human revision of a plan since a global redesign of a plan is only carried out if it is strictly necessary given that the execution of a completely new plan may be very costly in terms of reconfiguration of the agents and actuators in the real world. In this sense, local redesigns of a globally valid plan are more preferred. However, in the case that a local replanning is infeasible, a global replanning must be carried out.

Instead of using a specialized replanning algorithm, the original algorithm in Figure 1.b) was redesigned so that it is able to work in a purely generative episode and also in a replanning episode. The main additions are

- It is able to replan over a plan that is being executed, taking into account the existence of already executed actions and CKH. One of the main difference with the original algorithm (step 3) is that a subgoal may only be solved by a previously existing action when the age of the reused action is lower that the age of the subgoal. For newly generated subgoals there is no difference since their time stamp is ∞ and all of the actions are potentially candidate. For replanned subgoals, actions newer that the subgoal are not considered since they did not exist in the chonological point at wich the goal was introduced.
- It reuses and extends the remaining plan rationale $\mathcal{L}$ and reorder chronologically every newly inserted causal link maintaining the original order (step 4a).
- It is able to include the actions suggested by the user in $\mathcal{P}$ although, due to the search process these actions might be finally rejected (step 3b).
- The role of the time stamp in the causal structure is increased to provide a local replanning capability. In fact subgoal resolution and causal link generation depends on the order imposed by existing time stamps.
- In the case of a backtracking during a local replanning (steps 3e and 5c), all of the actions newer than the current problem being replaned are deleted triggering a global replanning process.

Therefore, during a generative episode $\mathcal{Q} = \mathcal{P} = CKH = \emptyset$ and all of the goals in $\mathcal{A}$ are of the form (l, a, ∞). On the other hand, during a replanning episode $\mathcal{Q}$, $\mathcal{P}$ and CKH may be non empty, and goals in $\mathcal{A}$ may have a time stamp different than ∞

6 Concluding remarks

This paper has shown the replanning capabilities of SIADEX, a system being developed under a research contract with the Andalusian Regional Government to assist technical staff in the design of forest fire fighting plans. Although this a very specific domain, the techniques shown in the paper are domain independent and they might be applied to any domain. These techniques are based on a local replanning capability with the following features

- It is a local replanning procedure since it does not produce a global replanning process if it is not necessary, so only the neighborhood of the failed action is replanned.
- It is a mixed initiative approach in which the user is informed of the existence and the impact of the failure and he/she is able to patch the plan by adding/deleting either actions or goals.

SIADEX-R(Π**,**$\mathcal{Q}$**,** $\mathcal{P}$**,** $\mathcal{O}$**,** $\mathcal{L}$**,** $\mathcal{A}$**,** $\mathcal{D}$**,** CKH**)**

1. **If** $\mathcal{A}$ is empty **Return** Π,$\mathcal{O}$, $\mathcal{L}$
2. **Goal Selection**: Extract (l, a, t) from $\mathcal{A}$
3. **Action Selection**: Select an action a' to solve (l, a, t) such that either is a new action or a previously existing action with a time stamp older than the goal[a].
   (a) $a' \in \mathcal{X}$ and $l \in CKH$ Reuse an already executed action whose effects are still in CKH
   (b) $a' \in \mathcal{P}$ Reuse an action suggested by the user
   (c) $a' \in \mathcal{Q} \cup \Pi$ and $\tau(a') < t$ Reuse an action still to be executed. This corresponds to a delay in the execution of a that was originally scheduled to be executed before, but, due to the failure it is still possible to execute it later reusing part of the existing plan rationale.
   (d) $a' \in \mathcal{D}$
   (e) **Return FAIL** and
      i. $\forall a \in \Pi \cup \mathcal{Q}$ such that $\tau(a) > t$ **DeleteAction(a)**
4. Update
   (a) if $t = \infty$ (new goal) $\mathcal{L} = \mathcal{L} \cup (a', l, a, NOW())$
   otherwise (replanning goal)
      i. $\mathcal{L} = \mathcal{L} \cup (a', l, a, t)$
      ii. Increase in 1 time unit the time stamp of every goal $(l, a, t') \in \mathcal{A}$ and every causal link (a_1, l, a_2, t') such that $t < t'$
   (b) $\mathcal{O} = \mathcal{O} \cup a' < a$
   (c) $\Pi = \Pi \cup a'$
   (d) $\forall l \in Preconditions(a'), \mathcal{A} = \mathcal{A} \cup (l, a', t + 1)$
5. Solve causal interferences[b]: if an action a^t threatens a causal link (a^p, l, a^c, t) then either
   (a) $\mathcal{O} = \mathcal{O} \cup a^t < a^p$
   (b) $\mathcal{O} = \mathcal{O} \cup a^c < a^t$
   (c) **Return FAIL** and
      i. $\forall a \in \Pi \cup \mathcal{Q}$ such that $\tau(a) > t$ **DeleteAction(a)**
6. Recursively call **SIADEX-R(**Π**,**$\mathcal{Q}$**,** $\mathcal{P}$**,** $\mathcal{O}$**,** $\mathcal{L}$**,** $\mathcal{A}$**,** $\mathcal{D}$**,** CKH**)**

[a] For newly generated goals (with time stamp ∞) every action is a possible one. For replanning goals with a time stamp t only actions whose time stamp is lower are considered, that is, actions that were previously generated in the search tree. Newer actions, i.e., actions that were generated later in the search tree are not considered.

[b] This step does not consider time stamps since interferences may appear between actions and causal links of any age.

Fig. 5. The planning and replanning algorithm of SIADEX

– It might be used to start a planning process either from scratch or from initial suggestions of the user and regenerating a plan over them. This is because this process is inherently the same to replanning so that the interaction with the user may be even closer.

On the opposite hand, it must be said that these techniques are not complete since after a failure in the execution, only the plan rationale is deleted and regenerated. Ordering and binding constraints posted by failed actions are not deleted neither regenerated though they are being studied for a more comprehensive replanning procedure in the SIADEX project.

References

1. J.A. Ambros-Ingerson and S. Steel. Integrating planning, executiong and monitoring. In *Proceedings of AAAI-88*, pages 83–88, 1988.
2. J. Blythe. Decision-theoretic planning. *AI Magazine*, 20(2):37–54, 1999.
3. L. Castillo, J. Fdez-Olivares, and A. González. A conditional approach for the autonomous design of reactive and robust control programs. In *Proc. of AIPS02 Workshop on Planning for Real World*, pages 59–68, 2002.
4. L. Castillo, J. Fdez-Olivares, and A. González. A temporal constraint network based temporal planner. In *Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG 2002*, pages 99–109, 2002.
5. L. Castillo, J. Fdez-Olivares, and A. González. Some issues on the representation and explpoitation of imprecise temporal knowledge in an AI planner. In *Knowledge-Based Intelligent Information and Engineering Systems*, Lecture Notes in Artificial Intelligence, LNAI-2774, pages 1321–1328. Springer-Verlag, 2003.
6. M. de la Asunción, L. Castillo, J. Fdez-Olivares, O. García-Pérez, A. González, and F. Palao. SIADEX: Assisted Design of Forest Fire Fighting Plans by Artificial Intelligence Planning techniques. `http://decsai.ugr.es/siadex`, 2003.
7. O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An approach to planning with incomplete information. In *Proc. Third. Int. Conf. on Principles of KRR-92*, pages 115–125, 1992.
8. P. Morris and N. Muscettola. Execution of temporal plans with uncertainty. In *AAAI 2000*, pages 491–496, 2000.
9. N. Muscettola, P. Pandurang Nayak, B. Pell, and B. C. Williams. Remote agent: to boldly go where no AI systems has gone before. *Artificial Intelligence*, pages 5–48, 1998.
10. K. L. Myers. CPEF: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, 1999.
11. M. A. Peot and D. E. Smith. Conditional nonlinear planning. In *Proc. First Int. Conf. of AIPS*, pages 189–197, 1992.
12. D. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.
13. D. E. Wilkins. *Practical planning: Extending the classical AI planning paradigm*. Morgan Kaufmann, 1988.