In Proceedings of ICME-95, 6th International Conference on Manufacturing Engineering, Melbourne, Australia, November 29–December 1, 1995, pp. 461-466

A TMS-based Approach for Efficient Conflict Resolution and Replanning in Resource Allocation Problems

Nikos I. Karacapilidis
GMD, FIT.KI - Artificial Intelligence Research Division
German National Research Center for Information Technology
Schloss Birlinghoven, 53757 Sankt Augustin, Germany
e-mail: karacapilidis@gmd.de

Dimitris Papadias
Dept. of Computer Science and Engineering
University of California, San Diego
La Jolla, CA, 92093-0114
e-mail: dimitris@cs.ucsd.edu

Summary: This paper describes how results from Truth Maintenance Systems and Temporal Abstraction Techniques can be combined for efficient problem solving. The above combination is proved useful for problems in which temporal abstractions are inherent such as resource allocation problems. The temporal abstraction techniques that are involved here are: *temporal summarization*, that replaces a set of time intervals by a single one, and the *value-assignment delay heuristic* that decomposes the problem into clusters of variables, based on the ease a resource can be assigned. The proposed formalism provides an architecture for the representation of consistency as well as a problem solving framework that efficiently resolves conflicts on resources contention and manipulates rescheduling with the minimum effort. The need for backtracking is limited and rescheduling is rendered a quite easier and faster task.

1. INTRODUCTION

The main tasks in practical planning are the achievement of objectives by satisfying a set of constraints and the reaction to contingencies, i.e., unforeseen events requiring replanning. Replanning becomes necessary in order to precisely react to changed conditions. It is also commonly admitted that the majority of practical planning problems requires search by reduction in their solution spaces and abilities both for incremental replanning and representation of consistency.

Beginning from scratch and reformulating a new problem when contingencies occur is not a feasible strategy. On the contrary, research has been concentrated on the minimization of the perturbation of an existing plan. Truth Maintenance Systems (TMSs) have been designed to support incremental replanning by propagation of perturbations along dependency links, as shown by De Kleer (1) and Doyle (2). However, the efficient replanning of manipulations has not been a reality in practical applications.

Petrie (3, 4) defines Constrained Decision Problems (CDPs) as problems that are characterized by multiple objectives, a large space of acceptable solutions from which only one is sought, general constraints, succinct representation by decomposable goals and replanning. A model for the handling of these problems is also provided to identify the replanning tasks and the dependencies among plan elements sufficient to perform replanning. CDPs are based on a backward chaining rule system, that is a TMS, and on the generation of TMS justifications that cache backward chaining proofs. In this approach a set of predicates, rules, algorithms and dependency links have been built for the representation of consistency as well as the manipulation of contingencies.

Temporal abstraction techniques are generally considered to reduce the complexity of problem solving in domains that they are inherent, such as the allocation of operating rooms and technicians to surgical operations in a hospital or the allocation of machines, workers and tools in a job-shop production system. Another use of the abstraction techniques is to provide useful input to rescheduling and feedback to the planning and scheduling tasks. It was shown by Choueiry and Faltings (5) that these abstractions can be used to improve future predictions and decisions.

In this paper we describe how results from TMSs and Temporal Abstraction Techniques can be combined for efficient problem solving. Although these techniques have been studied independently, in this paper we suggest a new approach that applies them together in resource allocation problems. A TMS-based formalism is outlined that simplifies the representation of consistency and the manipulation of rescheduling. A problem solving procedure is also presented that reduces the problem by making assignments and specifying new goals. The proposed formalism is constructive in the sense that new objects are typically created at runtime, e.g., a specific machine allocation in a specific job. Thus, it avoids defining all combinations of values and variables for all possible objects, whether used in the final plan or not. Additionally, it renders rescheduling an easier task because of the attainment of minimal change to a plan and the formulation of control.

Section 2 describes the resource assignment problem to be used as a test bed for the proposed approach. Section 3 outlines related work on temporal abstraction techniques and TMSs. The section does not aim at giving an analytical description of these techniques, but at highlighting semantics on the context of the paper. The proposed formalism and problem solving procedure are analytically presented in Section 4. Finally, derived remarks are outlined and discussed in Section 5.

2. PROBLEM DEFINITION

As it is outlined by Baker (6), a solution to a scheduling problem is any feasible resolution of two types of constraints:

- limits on the capacity of available resources, specifying the resource allocation problem, and

- technological restrictions on the order in which tasks can be performed, specifying the sequencing problem.

These two subproblems are often solved simultaneously, although, in a variety of applications the definition of time intervals in which a specific task has to be processed has been decided in advance and, consequently, the only existing problem is the allocation of the resources. The definition of the above intervals is performed either by forward scheduling techniques, based on the release date of a job, or by backward scheduling techniques, based on the due date of the job.

In this paper, it is assumed that the execution times of jobs have been already decided according to one of the above techniques. There is a set of jobs J={J1, J2, ..., Jn} and a set of machines M={M1, M2, ..., Mm} able to perform them. Each job can be performed by a specific subset of M. The assignment of a machine to a job is treated with respect to a set of constraints, such as the existence of necessary tools and/or workers, the assurance of a good machine operation, the machines' suitability for the job and the assumption that each machine can perform one job at a time.

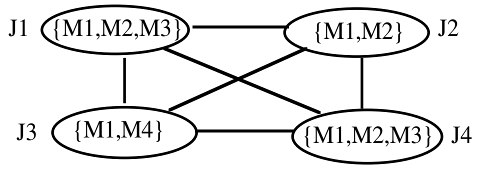

Figure 1: *A constraint graph for the problem.*

A resource allocation problem can be expressed as a constraint satisfaction problem with the aid of a constraint graph, where nodes represent jobs to be scheduled, values represent the suitable machines for each job and arcs denote capacity constraints (i.e., a machine cannot be allocated to more than one job at a time). Figure 1 illustrates an example to be analytically considered in the sequel. After a consistent assignment of machines, a variety of contingencies may occur at any time: a break down of a machine, or an unexpected absence of an appropriate worker or tool. Thus, the initial solution leads to inconsistency and has to be changed to preserve constraint satisfaction.

3. RELATED WORK

Abstractions are widely considered as very good techniques in order to reduce the search space and, consequently, to simplify the problem solving procedure. Abstraction is defined by Giunchiglia and Walsh (7) as the mapping of a problem representation into a simpler one that satisfies some desirable properties in order to reduce the complexity of reasoning. The problem is solved in a simplified abstract space and the solution is then mapped back to the more complex ground space. The main concepts of two temporal abstraction techniques, the *value-assignment delay* heuristic and *summarization*, are outlined with respect to the proposed problem solving formulation:

(i) Value-Assignment Delay heuristic: A value-assignment delay (VAD) heuristic, primarily aimed at list-colouring problems, has been proposed by Choueiry and Faltings (5, 8). The main idea of the heuristic is to delay the assignment of the most solicited values and try to solve the simplified problem without them. The adopted approach lets the variables compete over other possible assignments and, finally, distributes the delayed values over those variables that really cannot do without them.

The choice of values to be delayed is based on measurements of resource contention. The VAD heuristic implicitly causes a partitioning of objects by "grouping values on the trunk of the partitioning tree and by pushing variables into the leaves". That is, when value contention takes place, it decomposes the resource allocation problem in a way that decision making upon resource assignments is performed primarily for the less competitive variables.

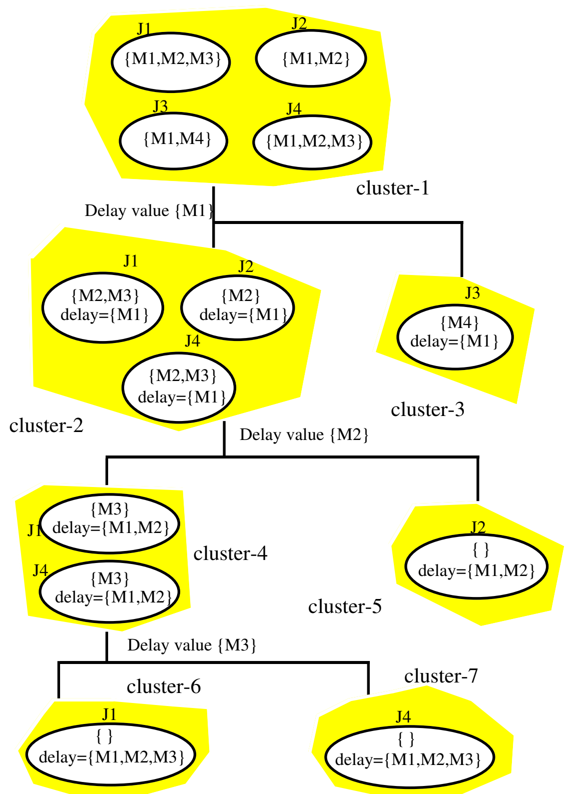

Figure 2: *VAD heuristic for the example of Fig.1.*

Figure 2 outlines the above heuristic for the problem of Figure 1. In cluster-1, each node contains the set of machines that can be assigned to a specific job (in brackets). Using the VAD heuristic, the most common value, M1, is delayed, thus splitting the initial cluster into cluster-2 and cluster-3. In the latter, M4 is assigned to J3. In cluster-2 the three nodes are still competing for common values, having as reserve the value M1. Next, the most common value, M2, is delayed and clusters 4 and 5 arise. In cluster-4 , jobs J1 and J4 are still competing for M3. Thus, M3 is delayed splitting cluster-4 into clusters 6 and 7. Clusters 5, 6 and 7 are left with an empty set of possible values and a reserve set of delayed values (that is, M1 and M2 for cluster-5 as well as M1, M2 and M3 for clusters 6 and 7). This is a *conflict*.

Conflicts are defined by the set of delayed resources at a branching node of the clustering tree. Consequently, decisions have to be made to allocate the delayed resources in the above conflicts to any of the soliciting tasks according to the constraints imposed by the graph in Figure 1. This task is called *conflict resolution*.

(ii) Summarization, as a temporal abstraction technique introduced by Choueiry and Faltings (9), is the replacement of a collection of time tokens by a single one. The interval of the summarization is the interval between the start time of the earliest time token and the end time of the latest one. The derived time token combines the characteristics of the consistuent tokens in a disjunctive way. Summarizations strengthen the above clustering algorithm by reducing the size of the cluster at hand.

Over the last decade, general purpose *Truth Maintenance Systems* (TMSs) have been developed and applied in a number of domains. TMSs have the potential to significantly improve the efficiency of problem solving. However, this potential can only be realized with a suitable interface between the truth maintenance system and the problem solving program.

In many practical applications, the need to make decisions based on limited information means that it is difficult to make the right choices every time. Additionally, contingencies usually cancel our previous decisions. When decisions are found to be wrong they must be revised. Furthermore, it is desirable to avoid repeating such futile choices in the future. One of the motivations for the development of TMSs is to provide a general solution to this problem, as shown by De Kleer (1). In other words, to ensure that search programs are able to carry results obtained in one part of the search space to other parts.

A TMS is intended to be a subsystem of a problem solver that searches for a feasible solution. The problem solver informs the TMS about the belief of problem solver data, the reasons for these beliefs and about sets of mutually inconsistent beliefs. In return, the TMS can inform the problem solver if there is a current set of mutually consistent beliefs. The key reasoning operations performed by a TMS are the *consistent labelling* and the *contradiction resolution*.

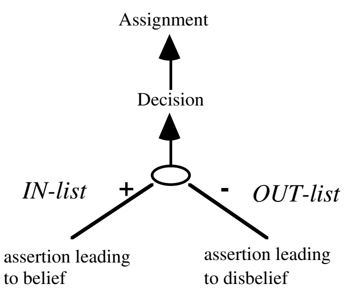

Figure 3: *A Justification scheme.*

A TMS dependency network offers a purely syntactic, domain-independent way to represent belief and change it consistently. Figure 3 shows how justifications operate in a TMS. Each justification comprises two parts: an IN-list (whose assertions are connected by "+" links) and an OUT-list (whose assertions are connected by "-" links). The justification is connected by an arrow to the decision and, consequently, to the assignment that it supports. In other words, in a TMS dependency network nodes (i.e., assignments) are believed when they have a valid justification. Finally, a justification is valid if every assertion in the IN-list is believed to be true (is IN, or marked "+") and every assertion in the OUT-list is believed to be false (is OUT, or marked "-").

In a TMS network, nodes are labelled with a *belief status*. The *labelling task* of a TMS is to label each node in such a way that assures consistency for the dependency network structure, as shown by Rich and Knight (10). The other major task of a TMS is resolving contradictions. A contradiction node does not represent a logical contradiction but rather a state of the database explicitly declared to be undesirable. The TMS network has to make all contradictions OUT.

The TMS described above is known as Justification-Based Truth Maintenance System (JTMS), and it was introduced by Doyle (2, 11). Some important reasoning operations that a JTMS does not perform are: a) applying rules to derive conclusions, b) creating justifications for the results of applying rules, c) choosing among alternative ways of resolving a contradiction, and d) detecting contradictions. The Assumption-Based Truth Maintenance System (ATMS) is another way of implementing non-monotonic reasoning. While in a JTMS a single link of reasoning is pursued at a time, in an ATMS alternative paths are maintained in parallel. In a JTMS, dependency-directed backtracking is performed whenever there is a need to change the assertions of the system. On the other hand, in an ATMS backtracking is avoided at the expense of maintaining multiple contexts, each associated to a set of consistent assumptions. It has been shown by Smith and Kelleher (12) that an ATMS perform breadth-first analysis while a JTMS follow a depth-first way of operation. The corresponding problem solver has to create nodes to represent the assertions that are derived either by itself or are given as premises (axioms), to associate various justifications with each of the above nodes and, finally, to inform the ATMS about inconsistencies. There is no need to construct rules that explicitly select the path to be followed in the above procedure. On the contrary, in a ATMS, reasoning proceeds by creating and maintaining multiple contexts and, consequently, by pruning the universe of inconsistent contexts whenever a contradiction is discovered.

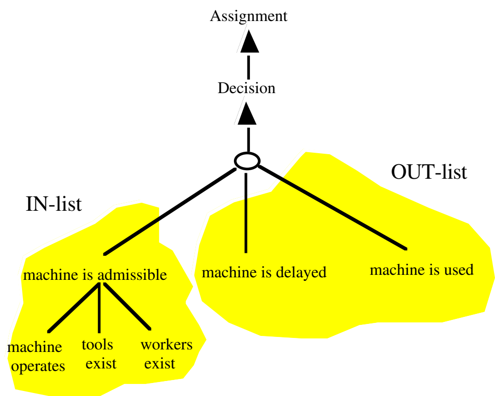

Figure 4: *The proposed justification*

4. SOLUTION METHOD

In order to preserve temporal abstractions in the reasoning process of the problem, a new, problem-oriented justification scheme is formulated. In this scheme, illustrated in Figure 4, in addition to assertions that lead to the belief or the disbelief of a decision (*machine is admissible*), *delay assertions* are attached. The delay assertions comprise assertions that lead to the consideration of the specific assignment as *reserve* and result in delayed justifications. In order to follow the classical TMS terminology, delay assertions are considered to belong in the OUT-list of the justification. Assertions leading to disbelief aid to the representation of contingencies. In addition, in order to efficiently handle capacity constraints, concerning possible job overlapping, assertions for the usage of a specific machine (*machine is used*) are attached. They also belong in the OUT-list of a justification.

Each kind of assertion in the proposed justification scheme can be supported by a set of *admissibility conditions* that may be premises, assumptions or justifications leading to the belief, disbelief, or delay of an assertion.

The role of the proposed formulation is to:

- Label each node with the contexts in which it has a delayed justification;
- Label each node with the contexts in which it has a valid justification;
- *Propagate delays*, that is, to maintain as reserve those contexts that include subcontexts which have been delayed;
- *Propagate inconsistencies,* that is, to rule out contexts that include subcontexts which have been known as inconsistencies.

Each cluster derived from the VAD heuristic described above can be depicted as a node of a classical TMS. The heuristic is exploited in the problem solving procedure of the system. In the proposed system, each node may have one or more justifications. If all of the assertions in the IN-list are valid while none of the assertions in the OUT-list is, then the justification is valid and, consequently, the node is supported. For a node to be valid, it must have at least one valid justification. A link of an assertion leading to disbelief initially has no justification and so cannot be valid.

For each node of the TMS-like architecture a set of delay operators, *del-operators*, are defined upon the resources that participate in the existing assignments of the node. The rationale for choosing a *del-operator* instance aims at a valid justification of the *best assertion* for each assignment. Each time the problem can be reduced, a *preference procedure,* based on measurements of resource contention, determines the del-operator which will be assigned. When this assignment is performed, all assertions leading to the delay of the specific machine become IN.

Each *del-operator* is also associated with a set of admissibility conditions, a set of conditions such that if all are valid an operator can be applied for the decision reduction. Admissibility of an operator depends only on assumptions about the scheduling environment (e.g., the availability of a resource) and declares that this resource will be available in the future. Events may lead such conditions to falsehood (e.g., a machine break-down or an unforeseen lack of skilled workers). In these cases some decisions become invalid and, consequently, there is a need for rescheduling.

Figure 5 outlines the problem solving procedure while Figure 6 illustrates how the proposed problem solving works. There is a set of operators aiming at decision reduction by delaying a value, *del-operators*, and a set of procedures aiming at conflict resolution, *core-procedures*. The former set decides the value that has to be delayed first (e.g. the value that participates in most variable domains) and reduces the problem to a new goal and a set of possible assertions for each assignment. *Core-procedures* decide the pattern of value assignment in order to achieve conflict resolution. It is clear that summarization, as it was described in Section 3.2, may aid to the conflict resolution and it can be considered as a *core-procedure*.

```
1:For each reducible goal
       /* there is at least one cluster with jobs
       /* competing for the same resource(s)
2:{apply a del-operator
3:     {consider the first one (out of an order
              defined by a preference procedure),
4:      check its admissibility conditions for
              consistency,
5:      if consistency holds, go to 9:
6:            else,
               while there are more del-operators
7:             { consider the next del-operator
8:               go to 4: }
9:      delay the most common resource
       /* mark "+" the associated machine_is_
       /* delayed links in each derived cluster
10:    }
11:if all del-operators are lead to inconsistency
       then failure
12:isolate in a cluster jobs not competing for
       any value
       /* having all values delayed
13:make clusters of jobs competing for the same
       value
14:if a cluster has more than one job
15:    {define a new reducible goal
16:     go to 1:}
17:for each remaining cluster:
18:    if there is a conflict
19:           apply a core-procedure to achieve
           conflict resolution
20:    make final assignments
21:           {find: best assertion
22:                  delay assertions}
23:}
```

Figure 5: *The problem solving procedure*

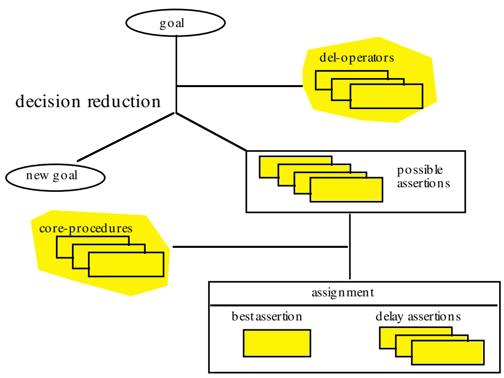

Figure 6: *The problem solving architecture*

In Figure 7 the proposed problem solving procedure is applied for the example of Figure 2. The considered admissibility conditions for each *del-operator* are the proper operation of the machine and the assurance of the appropriate tools and workers. All of them are valid (in boxes with grey shadow) and the *del-operator* M1 is primarily applied to the initial goal "allocate J1, J2, J3, J4". The problem is reduced to the goal "allocate J1, J2, J4" by attaching a final assignment for job J3, since there is no conflict for it. Thus, in this step there is no need for the application of *core-procedures*. In a similar way, the *del-operator* M2 is applied and leads to the reduced goal "allocate J1, J4" and an (not final) assignment for J2. The *del-operator* M3 is finally applied producing non-final assignments for J1 and J4. Jobs J1, J2 and J4 are in conflict and, according to the procedure described above, a *core-procedure* has to be selected in order to provide conflict resolution.

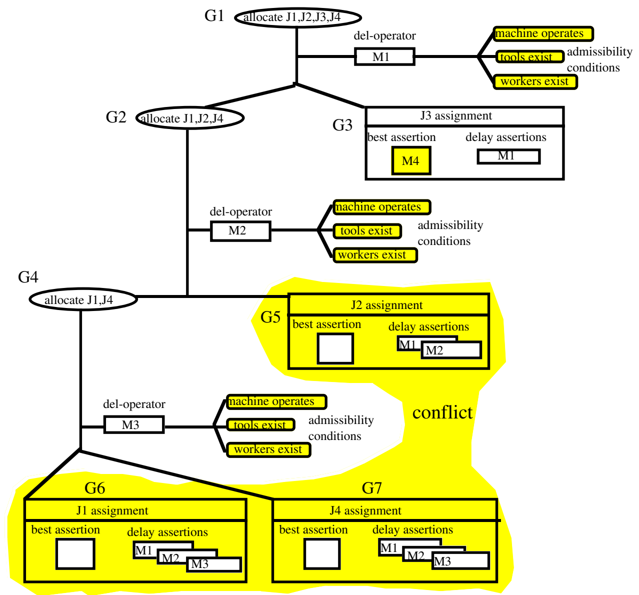

Figure 7 : *The TMS-based architecture for the example of Fig.2.*

The proposed core-procedure is as follows:

```
Working in a bottom-up way
{find the most recently delayed resource
       if      machine_is_admissible is "+"
               and machine_is_used is "-"
       then    mark "-" the machine_is_delayed
               link
               propagate machine_is_used as "+",
                      for the specific resource,
                      in the whole tree
       else    continue with the next delayed
                      resource.
 if there is not a resource with
               machine_is_admissible "+" and
               machine_is_used "-"
       then    there is not solution for the
               specific job}
```

With the proposed problem solving procedure, when a contingency occurs the elimination of inconsistencies is an easy task. Each assignment has a set of delay assertions, thus when its *best assertion* is retracted the problem solver has to search in the set of *delay assertions* to find an alternative solution.

Figure 8 illustrates the initial status of the problem after the labelling of the links corresponding to the application of a *del-operator*. Initially, all *used* links are OUT. It can be assumed that all *adm* (stands for admissible) links are valid. Figure 9 illustrates the status after the application of the proposed problem solving technique for conflict resolution. Starting with J1 and assigning M3 to it, the propagation of *M3 used* as IN retracts the need of consideration of all branches associated to M3 (i.e., branch 1). In a similar way, assigning M2 to J4 retracts the need of consideration of branch 2, and assigning M1 to J2 that of branch 3. Suddenly a contingency occurs: The only worker that can operate M3 for job J1 is absent. Our algorithm picks up the second choice and works similarly. The shadow area of Figure 10 indicates the perturbation to the existing problem solution caused by the occurrence of the above contingency.

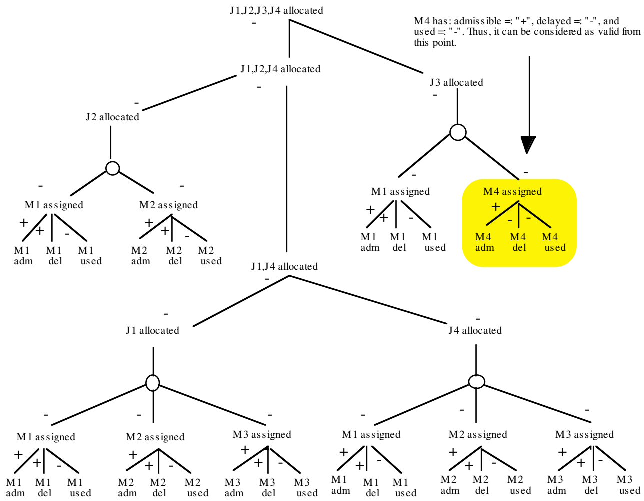

Figure 8: *Initial status of the problem.*

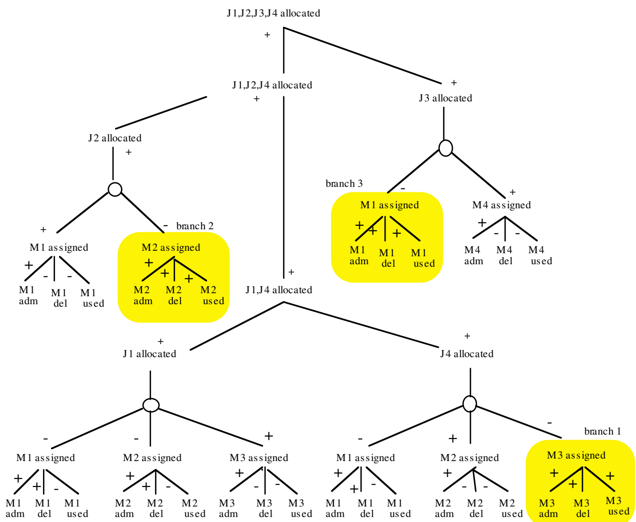

Figure 9: *The problem status after the application of the proposed technique.*

5. REMARKS AND CONCLUSION

According to the problem solution formalism, contexts are delayed or eliminated as the problem solver asserts delays and finds inconsistencies. Each of the nodes of the TMS network has a set of possible assignments indicating that a specific resource (machine) can be allocated to a job. By applying a *del-operator* with the aid of a predefined procedure, the delay is propagated downwards and, simultaneously, it reduces the search space. In this way, each node of the above figure has a set of delayed assignments and/or a set of possible assignments. According to procedures based on resource contention, the problem solver decides whether more delayed assignments have to be created. If no more delayed assignments are needed, the problem solver again decides the final allocation of *best assertions*.

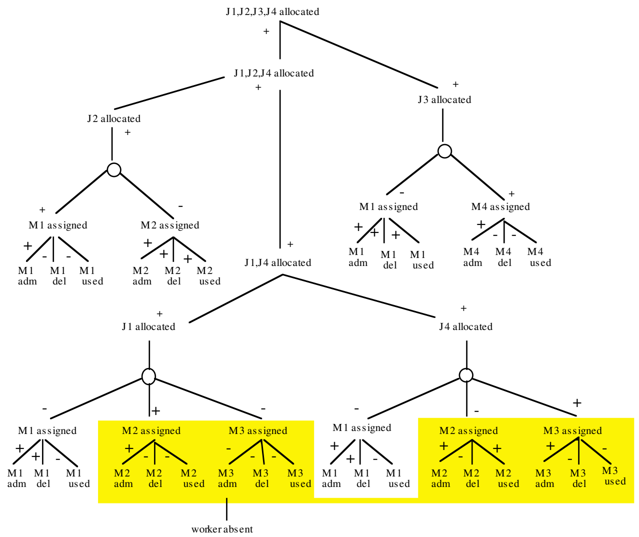

Figure 10: *The problem status after the alteration caused by a contingency.*

The following remarks are revealed from the proposed approach:

i) The resource allocation problem is undoubtfully a Constraint Satisfaction Problem. However, constraint satisfaction or integer programming techniques are unsuitable for these problems for the following reasons (see also the work of Petrie (4)):

- In order to formulate a constraint satisfaction or integer programming problem, one must enumerate all possibilities (in general, such problems are known to be NP-complete). Thus, defining all variables means anticipating the creation of all possible contexts, whether used in the final plan or not. This is a cumbersome and usually needless task in the formulation of constructive problems. However, the proposed formalism is constructive in that new nodes are typically created at runtime, based on the operator selected.
- Replanning from scratch is undesirable because of the actual target of minimal change to an existing resource allocation plan.

ii) The use of a TMS architecture provides a convenient and efficient tool for the implementation of the proposed formalism. It facilitates the automatic generation of dependencies preserving semantics about them.

iii) The use of temporal abstraction techniques and especially of the VAD heuristic reduces the size of the search space and prunes the search tree in a significant way.

iv) The admissibility conditions both for the *del-operators* and the assertions leading to belief provide a suitable way of representing a set of conjunctive constraints related to the proper use of the eligible resources.

v) The application of the proposed formalism in relatively small problems can be considered as a suggestive evidence to warrant successful development of larger ones.

Concluding, the partial advantages of temporal abstraction techniques and truth maintenance based structure are exploited in this paper, aiming at the solution of a resource allocation problem. A TMS-like justification and problem solving schemes are proposed in order to achieve a more efficient search reduction and associated pruning of the search space. The proposed combination leads to the limitation of backtracking, and rescheduling becomes a quite easier task. It can be considered as a trial to eliminate the essential difference between problem reduction and problem solving. Extension of the proposed formalism in a general scheduling problem, considering both resource allocation and sequencing, is proposed as an essential direction for further research.

ACKNOWLEDGEMENTS: Dimitris Papadias is supported by National Science Foundation grant IRI-9221276.

REFERENCES

1. J. De Kleer, “An Assumption-Based TMS”, Artificial Intelligence, Vol.28, No 2, 1986, pp. 127-162.
2. J. Doyle, “A truth maintenance system”, Artificial Intelligence, Vol.12, 1979, pp. 232-272.
3. C. Petrie, “Planning and Replanning with Reason Maintenance”, Technical Report EID-385-91, MCC Enterprise Integration Division, Austin, TX, 1991.
4. C. Petrie, “Constrained Decision Revision”, Technical Report EID-414-91, MCC AI Lab, Austin, TX, 1992.
5. B.Y. Choueiry and B. Faltings, “Interactive Resource Allocation using Temporal Abstractions and Value-Assignment Delay Heuristic: A CSP Perspective”, Technical Report TR-92/17, EPFL, Lausanne, Switzerland, 1992.
6. K.R. Baker, “Introduction to Sequencing and Scheduling”, J. Wiley & Sons, New York, 1974.
7. F. Giunchiglia and T. Walsh, “A Theory of Abstraction”, Technical Report IRST 9001-14, Istituto per la Ricerca Scientifica e Tecnologica, University of Trento, Trento, Italy, 1990.
8. B.Y. Choueiry and B. Faltings, “Temporal Abstractions and a Partitioning Heuristic for Interactive Resource Allocation”, in Working Notes of the Workshop on Knowledge-Based Production Planning, Scheduling and Control, IJCAI-93, Chambery, France, 1993, pp. 59-72.
9. B.Y. Choueiry and B. Faltings, “Building Temporal Abstractions”, in Working Notes of the Workshop on Approximation and Abstraction of Computational Theories, AAAI-92, San Jose, CA, 1992, pp. 47-51.
10. E. Rich and K. Knight, “Artificial Intelligence”, Mc Graw-Hill, New York, 1991.
11. J. Doyle, “Truth maintenance systems for problem solving”, Technical Report AI-TR-419, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1978.
12. B. Smith and G. Kelleher, “Reason Maintenance Systems and their applications”, Ellis Horwood Ltd., Chichester, England, 1988.