

An Interactive Planning Architecture *

The Forest Fire Fighting case

Anna Perini and Francesco Ricci

I.R.S.T.

38050 Povo (TN)

Italy

e.mail: {perini,ricci}@irst.itc.it

Phone ++39+461-314330

Fax ++39+461-302040

Abstract. This paper describes an interactive planning system that was developed inside an Intelligent Decision Support System aimed at supporting an operator when planning the initial attack to forest fires. The planning architecture rests on the integration of case-based reasoning techniques with constraint reasoning techniques exploited, mainly, for performing temporal reasoning on temporal metric information. Temporal reasoning plays a central role in supporting interactive functions that are provided to the user when performing two basic steps of the planning process: plan adaptation and resource scheduling. A first prototype was integrated with a situation assessment and a resource allocation manager subsystem and is currently being tested.

1 Introduction

This paper presents an interactive planning architecture that shall be used for planning the initial attack to forest fires. The system is part of an Intelligent Decision Support System aimed at supporting the user in the whole process of fire fighting including both situation assessment and planning activities [23].

Developing a practical planning application for a real domain problem raises several dimensions of complexity, as discussed in [24]. The founding assumptions of AI planning have to be carefully analyzed in light of the application requirements. First of all in a classical AI perspective there is a neat distinction of roles between the user and the planner: the user poses the goal, the environment sets the initial conditions and the planner finds the solution. This simplified view of the problem solving activity does not apply in our case. The user is an expert of the domain and wants to control the global information flow. In some cases he is able to solve the current goal, for example mostly regarding strategical decisions, in other cases he wants to set constraints on the search process. Decisions are always taken by the user, using the computer together with other tools and usually following a complex operational flow. So a system that can be used in an unstructured way is need. Assumptions may be made in any moment

*This work has been partially supported by the EspritIII project #6095 CHARADE (Combining Human Assessment and Reasoning Aids for Decision Making in environmental Emergencies). We thank all the partners of the project consortium and in particular Alenia who acted as project coordinator.

and changes in the focus of reasoning are continuous, for example new fire-alarms may arrive at any moment in the planning process.

The above mentioned issues stress various limitations of AI planning techniques (see [27, 22] for a more extensive discussion), in particular those developed in the context of pure formal approaches to planning.

Driven by domain requirements and constraints, a planning approach was defined that rests on the integration of Case-Based Reasoning [11] and Constraint Reasoning. Combining different specialized AI techniques is common in dealing with real planning problems [12]. A remarkable work in this direction is ongoing within the ARPA & ROME Laboratory Knowledge Based Planning Initiative (partially described in [28]).

The case-based module retrieves partial plans from memory using a novel similarity metric that has a local definition and can be adapted by a reinforcement learning procedure [21]. Plans are represented in a hierarchy of parts and partial plans can be completed using the similarity metric by searching for expansions of the leaves in the plan tree structure.

The constraint module adapts the retrieved plan, attaching and propagating constraints. Constraint reasoning is used mainly for managing temporal relations defined on sub-plans or between actions in plans. Constraints are also used in the identification of the sets of applicable actions in relation to particular scenario data. The temporal dimension of the plan is managed with a quantitative approach based on temporal variables over continuous domains. The constraint module supports also the resource allocation process for the chosen plan.

This paper focuses on the role of temporal reasoning in supporting interactive planning functions, an extended description of the CBR module developed for the application can be found in [2].

Section 2 illustrates the forest fire planning problem, Section 3 gives a brief description of the system architecture. The remaining sections are devoted to a presentation of the main interactive planning functions provided by the system: the plan representation is described first (Section 4), the temporal reasoning approach is presented (Section 5) and the main system functions are discussed (Section 6).

A first prototype of the whole Intelligent Decision Support System was completed. Besides the planning module it includes a situation assessment module¹ based on a cartographic module², modules for resource management and allocation³ and a man-machine interface⁴ based on an explicit representation of user tasks, as described in [18, 25, 14].

2 Planning Forest Fire Interventions

Planning for forest fires requires alternating phases of planning and situation assessment following a precise operational work-flow that is typical of each fire fighting organization. Here we focus on the planning problem. The different levels of abstraction at which reasoning is performed can be described as follows:

- Choosing the intervention strategy. For instance performing fire recognition, attacking the fire simultaneously on different sectors of the fire front ("sectoriza-

¹developed by Inisel

²developed by Thomson CSF/LER

³developed by Italsoft

⁴developed by Thomson CSF/SDC, Italsoft, Inisel, Alcatel-ISR

tion”), or combining basic strategies. Strategic decisions are supported by scenario data resulting from the global assessment of the fire alarm such as physical parameters of the fire, wind speed and direction, accessibility of the burning area, information on the availability of costly resources such as airplanes. This level is called the ”global” or ”fire plan” level.

- Finding tactics that allow a correct strategy implementation. A tactic is a partially ordered set of actions implementing specific fire fighting techniques. Typical actions are the construction of a fire line, the transportation of means, the recognition activity of a scout. The correctness of a tactic refers to the quantity of water, the resources to be employed and the temporal deadlines determined by the way the fire spread out. Tactical decisions require scenario data specific to each sector such as sector accessibility, sector length and deadline, buildings to be defended near the sector. This is the ”sector plan” level.
- Assigning the resources. The actions that require a certain type and quantity of resources. Resources have to be taken from bases or other fires and allocated to actions taking into account time constraints of the plan itself and constraints on resource allocation periods. This is the ”action” level.

3 The System Architecture

A partial view of the architecture of the planning system is depicted in Figure 1. This picture shows the main components: the case-based reasoner, which retrieves (stores) plans from (to) the plan memory; the plan space is the data space in which plans are installed; the action manager, that manages the user requests to modify the plans installed in the plan space; the constraint reasoner, in which variables and constraints are created and constraint reasoning algorithms called. In particular the picture shows the constraint network representing the temporal structure of the plan: temporal variables of the plan and constraints defined on them, as presented in detail in Section 5.

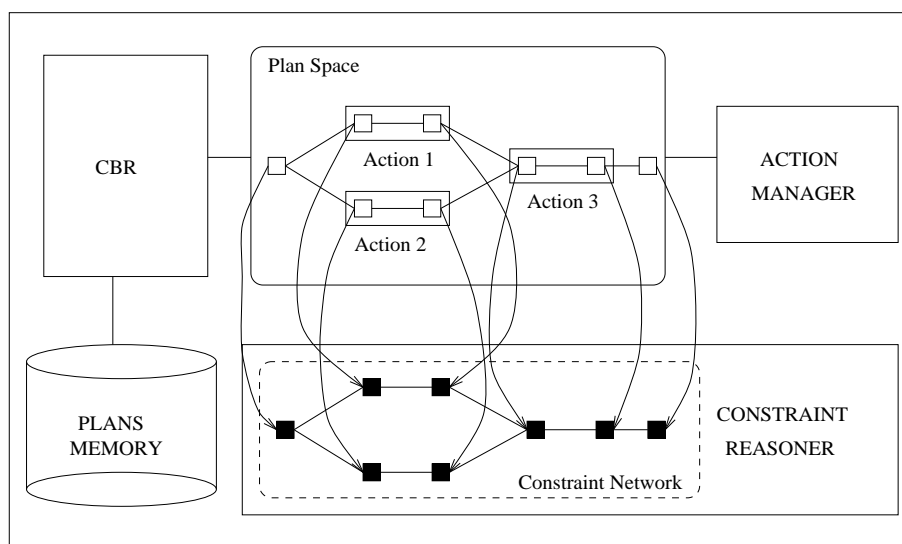


Figure 1: The architecture of the planner.

Plans are retrieved from memory by the case-based module. The temporal structure as stored in memory (constraints between plan’s components) has to be installed, that

is a temporal constraint network has to be defined on the time variables according to a script that is stored in memory. After that step, this data structure has to be fitted and modified to adapt it to the specific scenario data. Part of this process is done in batch mode by the system (fitting), part is performed in collaboration with the user through the action manager (adaptation). Moreover the temporal reasoner supports automatic and manual allocation (the relative modules are not depicted in Figure 1) finding correct schedule times for actions during resource allocation and updating the feasible times for actions not yet allocated.

4 Plan Representation

A hierarchical representation of plans allows a uniform description of the different abstraction levels to be given (Section 2) and makes the decomposition of reasoning problems into subproblems straightforward. The basic element of the hierarchical representation is the `action_net` structure that contains references to the plan it belongs to (`an_super`) and to its sub components (`an_subs`), as shown in the example depicted in Figure 2. The `action_net` structure represents also the temporal dimension of a plan component: the start and the end time variables that define the time interval during which the plan component is performed, the temporal relation between sub plans, constraints on action duration and deadline. The hierarchical structure was exploited for decomposing the temporal reasoning problem underlying the interactive planning process allowing for efficiency improvement, i.e. the temporal constraint network of an `action_net` contains temporal variables and constraints that are relevant for the represented plan element only.

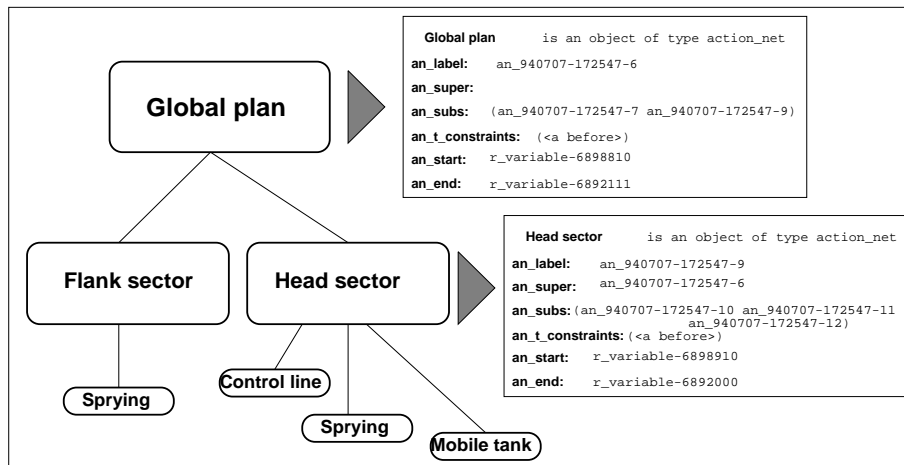


Figure 2: Part of hierarchy in plan representation. Two examples of `action_net` object are shown, only the main attributes are reported.

The `action_net` structure contains also a descriptive, domain dependent, component that collects the scenario data (resulting from the assessment of the fire alarm) that are relevant at a specific abstraction level, as described in Section 2. For instance the scenario data of an action contains information on the fire fighting technique such as the type and the number of resources to be used, the scenario data of a sector plan contains information on the fire situation, on the accessibility, etc. relative to the sector. The scenario data and the plan structure enter in the computation of the similarity metric that drives the search of the plan in memory.

At each plan level the system provides the user specialized functions that are suitable to support reasoning task specific to the abstraction level. So, for instance, action allocation can be performed only at the action level, insertion/deletion of actions and relations between actions can be performed only when working on sector plans, a sector plan can be deleted only when working on the global fire plan. The user can easily shift from one level to the other.

5 Temporal Data

When planning an intervention the operator uses time estimates and takes into account time constraints. Minimum duration estimates for actions are computed using heuristic knowledge on the work capacity of resources and squads when performing a specific fire fighting technique. Temporal relations can exist between pairs of actions, for instance the action of spraying retardant with the helicopter must be completed before starting a ground attack. Moreover all the actions of a sector plan must be accomplished within a given time, i.e. the sector deadline that represents the estimated time at which the fire will reach that sector. This brief analysis shows that the temporal information to be dealt with in the application is essentially metric.

Constraints on time points allow representing and reasoning about quantitative information, as discussed more deeply below, but cannot be easily grasped by the user. Intervals and constraints between intervals are more easily managed by the user. So the temporal dimension of a plan was represented at two levels:

- **The system’s user level.** Here time intervals are considered and constraints between time intervals are handled.
- **The constraint reasoner level.** Here time points are defined and time constraints representing bounded differences between time points are implemented.

From a user point of view, the temporal structure of a plan is represented through the temporal relations that can be defined between these time intervals in the form of single Allen’s interval relations [1]. That is, given two plan components, one of the thirteen Allen’s temporal relations can be stated for them. Mainly qualitative temporal relations are represented in this way with the exception of the interval relations “before” and “overlaps” (and their inverses) the extent of which can be further specified by a minimum and a maximum numerical parameter. The interval relations are mapped to a set of metric constraints representing bounded differences between the interval endpoints.

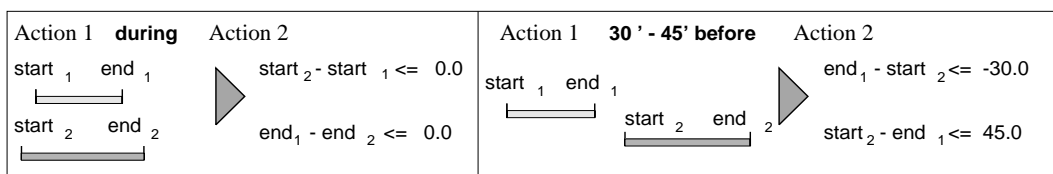


Figure 3: Examples of correspondence between interval relations and bounded difference constraints between the points representing the start time and the end time of actions.

Figure 3 shows an example of how Allen’s relations are translated into bounded difference constraints between time points. A bounded difference constraint between two time points X_i and X_j can be represented by the inequality $X_j - X_i \leq a_{ij}$, where a_{ij}

represents the bound on the distance between the two points. Qualitative relationships can be represented using the infinity and 0 values as distance bounds. For example “ X_i before or equal to X_j ” is expressed constraining the distance between X_i and X_j to be between zero and infinity ($0 \leq X_j - X_i \leq \infty$). Estimates on the minimum duration of actions can be represented directly as bounds to the distance between the start and end times of the actions.

5.1 Temporal Constraint Satisfaction Problems

A set of bounded difference constraints defined on a set of time points can be cast into a CSP problem defined on variables with continuous domains according to the formalization given in [9], the so called Temporal CSP (TCSP). We recall here some important properties of TCSP. In the TCSP formalism time points are represented by the variables of a constraint network. The variable domains are sets of real numbers. Unary and binary constraints on these variables correspond to temporal distances between pairs of variables (or between a variable and a reference time). Namely, given two temporal variables X_i, X_j a binary *temporal constraint* T_{ij} between them is a disjunction of intervals indicating the allowed values for the distance $X_j - X_i$, i.e. $T_{ij} : (a_1 \leq X_j - X_i < b_1) \vee \dots \vee (a_n < X_j - X_i \leq b_n)$.

The solutions of a TCSP with n variables are the n -tuples of floats corresponding to variable values that do not violate the constraints of the network. A TCSP is consistent if it has at least one solution.

The minimal network representation contains the network information in the most explicit form, i.e. all the possible values of the distance between the variables X_i, X_j are directly available from the label of the edge that connects X_i, X_j and considering the rest of the network does not produce any further reduction.

TCSP language allows representing disjunctive information like “event A ends about 10-20 seconds or about 40-60 seconds before event B starts”. Moreover qualitative point-interval relations and interval-interval relations can be represented [15, 26].

It is well known that determining the consistency and computing the minimal network of a general TCSP is not tractable. Current research focuses on finding polynomial algorithms that can give approximate solutions to these problems [9, 26] and recognizing subclasses of the language that are tractable [15, 10].

A tractable subclass is the Simple Temporal Problem (STP) in which single disjunct constraints can be represented [16, 8, 9]. The class of constraints dealt with in our application corresponds to this class.

Enforcing path-consistency on a STP gives the minimal network representation and is equivalent to computing all the shortest paths on the distance graph representing the STP [9], that is the directed edge-weighted graph $G = (N, E)$, in which each node $i \in N$ represents the temporal variable X_i of the STP network, and each edge $(i, j) \in E$ represents the linear inequality $X_j - X_i \leq a_{ij}$ (it is labeled by a weight a_{ij}). So the minimal network can be computed by the all shortest paths algorithm of Floyd-Warshall that works in $O(n^3)$ time [7]. An important property of the minimal network of STPs is that it is decomposable, i.e. any local assignment to any subset of variables, such that it satisfies the constraints involving only the variables of this subset, can be extended to a global solution. It follows that a solution of a decomposable network can be found with a back-track free search.

5.2 Shortest Paths and Arc Consistency

The common way to check both consistency and to compute the minimal network is path consistency, but one can use more economical procedures for checking the consistency and computing the feasible times for the network variables.

In [8] Davis proved that determining the consistency of an STP is equivalent to computing single-source shortest paths. The Bellman-Ford algorithm computes single-source shortest paths in $O(ne)$ time on a graph with both positive and negative weights.⁵

Computing the single-source shortest paths also gives the earliest times for the network's variables in respect to the source variable. The latest times can be computed by a further computation of the single-source shortest paths on the transpose graph. Note that for an STP the earliest and the latest times bound the (convex) interval of the feasible values of a variable.

An interesting point is the role of arc consistency on a STP. Arc consistency can be computed with algorithms that perform linearly with the number of edges in a discrete domain CSP, but it suffers from the termination problem in continuous domain CSP [8], i.e. in a not-consistent network arc consistency can enter in an arbitrary long loop and recognize inconsistencies only after a number of steps equal to D/p , where D is the maximum domain size and p the precision [13]. Basically arc consistency and single-source shortest paths algorithms exploit the same refining procedure. Arc consistency algorithms make networks stable with respect to the **arc-refine** operator⁶. Shortest paths algorithms make networks stable in respect to the Bellman-Ford operator, which is usually called **relax** [7]. It can be shown [3] that two applications of **relax** correspond to the application of the **arc-refine** operator and that when a network is consistent arc consistency and two applications of the Bellman-Ford algorithm can be equivalently used to compute the minimal⁷ feasible time interval for variables in an STP. This fact can be exploited in several ways. For instance, in a consistent network arc consistency should be preferred to shortest path computation.

In summary the expressiveness of the STP language allows us to represent the temporal information of the application domain. Disjunctions should be useful for representing information that is dealt with during the allocation process, such as disjointness between two actions performed by the same resource. At present a sequencing of parallel actions that can be performed by the same resource is provided by a search algorithm that uses the temporal reasoner for propagating the effect of this choice to the other actions.

The system requirement of supporting an interactive process poses the problem of using efficient computations. So a careful analysis of the system functions was performed in order to better exploit the arc consistency versus the shortest paths computation. An analogous approach was followed also in other works [4, 6].

⁵Other shortest-path algorithms can have a better performance, for instance Dijkstra's algorithm takes $O(n^2)$ time, but they require that all edges in the distance graph be nonnegative [7]. This corresponds to the case where only maximum duration constraints are considered.

⁶**arc-refine**, given a directed edge connecting the variables X_i and X_j , removes a value x_i from the domain of X_i if for all values x_j in the domain of X_j the pair of values (x_i, x_j) doesn't satisfy the constraints on the directed edge

⁷That interval cannot be further reduced without eliminating solutions of the network

6 System Functions

This section is focused on the functions that can be called by the user during plan adaptation and resource allocation. The temporal information relative to a sector plan is coded in a temporal constraint network (STP) associated to the `action_net` representing a sector plan. The variables of this network are the start and the end time points of the actions and of the sector plan itself. The constraints are those representing the minimum duration of actions and those representing the interval relations between actions. Moreover the part-of relation between the sector plan and each one of its actions induces a pair of temporal constraints between the sector start and the action's start and respectively between the action's end and the sector plan's end ($sector_{start} - action_{start} \leq 0$, $action_{end} - sector_{end} \leq 0$).

A brief note on variable representation and on two basic procedures can help in making clear the description of the functions given below (a more extensive description can be found in [19]). A time variable is represented by a set of features such as the value, the domain and the default domain. In particular the default domain which is a convex interval of real numbers, represents the maximum time window on which the variable can take values, independently from the constraints that are defined on it. Default domains are initialized to the whole plan time window and can be reduced for representing explicit assumptions, made by the user, on the a priori time window of a specific variable. Two basic procedures were implemented on a temporal network:

- the `propagate/ repropagate` procedure that computes arc consistency on the directed edges involving the set of variables specified in input (on all the edges of the network, by default). It updates the variables domains. The `repropagate` procedure sets the variables domains to their default domains first, as required when the new variables domains can be bigger than the previous one.
- the `check /recheck` procedure that implements the Bellman-Ford algorithm for computing the single source shortest paths on a directed arc graph. If the network is not consistent it returns false and restores a previous consistent network state, otherwise it updates the variables domains ⁸. The `recheck` procedure sets the variables domains to their default domains first.

The main interactive functions follow:

Inserting new actions or interval relations between actions in a sector plan.

Trying to insert a new action in a sector plan requires updating the underlying temporal network by adding two new variables (the start and end variables of the new action) and at least three new constraints (the minimum action duration and the two constraints induced by the part-of relation). Analogously inserting an interval relation between two actions requires adding the corresponding distance constraints between the start/end variables of the two actions. These modifications can cause a violation of the sector deadline so the consistency of the updated temporal constraint network associated to the sector plan must be checked. If no inconsistency is produced the `action_net` structure representing the sector plan is updated and the earliest and latest times of the variables of the sector temporal network are recomputed. This function is implemented by calling the procedure `check`.

⁸In fact the Bellman-Ford algorithm was implemented as an arc consistency algorithm plus a check on the existence of loops.

Inserting a new tactic in a sector plan.

A tactic is a partially ordered set of actions. For instance performing a "direct attack with the helicopter" requires setting up an artificial water reservoir first (by building a pipe line or by using tank trucks for supplying water) and then dispatching the helicopter with an appropriate bucket.

When inserting a tactic into a sector plan the constraints and the variables created during the instantiation of actions and relations of the tactic are added at the same time to the sector temporal network and only one call to the `check` procedure is performed.

Deleting actions or interval relations of a sector plan.

Deleting an action or an interval relation between two actions corresponds to removing the associated constraints (and the variables that are no longer linked) from the sector temporal constraint network. This can not produce inconsistency. The feasible time intervals of the network variables can increase so the variable domains have to be restored to their default values before refining them with arc consistency, that is the `repropagate` procedure is called.

Delaying action's start time or anticipating the end time.

Delaying the earliest start time of an action (no later than the latest possible value for the start variable) doesn't cause inconsistency since it means considering a sub-interval of the feasible values of the variable. But the feasible times of the variables connected to the action's start have to be updated. This is obtained by calling the procedure `propagate` only on the edges leaving from the start variable. Analogously an action latest end can be anticipated (no earlier than the earliest value of the end variable), by calling the procedure `propagate` only on the edges leaving from the end variable.

Modifying the duration of an action.

Modifying the action minimum duration corresponds to a change of the relative constraint. Weakening the constraint (i.e. reducing the minimum duration) doesn't produce inconsistency, the previous feasible values of action's start and end are still good values. These intervals must be updated including new consistent values, and analogously the feasible values of the variables connected to the action's start and end can be enlarged. This requires calling arc consistency after having reset the variable domains to their default domains (`repropagate`). When trying to increase the minimum duration the consistency has to be checked against sector deadline violation, so the procedure `check` is called on the sector temporal network.

Manual allocation of actions.

The manual allocation of an action consists of choosing a specific resource that is available at a given base and in dispatching it to the sector in which the action has to be performed. The time needed by the resource to reach the sector is computed (it is a situation assessment datum) and this allows defining a precise start time for the action. The temporal reasoner supports this task by assigning this value to the start variable of the action and propagating it along the edges connecting the action's start to other temporal variables of the sector constraint network (i.e. the procedure `propagate` is called on the edges leaving from the start variable). Undoing a manual allocation requires removing the value given to the start variable and calling `repropagate` for computing the feasible values.

Automatic allocation on the sector plan.

When plan adaptation has been completed the system can be required to compute resource allocation on the sector plan. Some of the actions may be already allocated

(by manual allocation). The Allocator orders the actions to be allocated according to some heuristics and schedules them setting the action start time to the earliest possible time updated according to the already scheduled actions. This corresponds to computing a solution (or completing a partial one) for the sector temporal network. Here the decomposability property of the minimal network of an STP is exploited. This property allows building a solution with a backtrack-free search propagating the values assigned to a subset of variables to the not yet assigned variables. So the temporal reasoner computes the minimal network of the sector temporal network before starting automatic allocation and supports the process of building a solution by updating the feasible values to the set of values that can complete a current partial solution.

7 Conclusion and Future Work

This paper presents a partial view of a system for planning first intervention attacks to forest fires. Constraint reasoning plays a central role in managing the temporal information related to a plan and in supporting interactive planning functions to be used during plan adaptation and resource allocation.

The temporal reasoning system implemented so far corresponds to the Simple Temporal Problem formalism defined in [9]. An interesting aspect of the work here described is the careful use of the different temporal reasoning computations that can be performed on a STP: determining the network consistency, computing the feasible times for the variables of the network, computing the minimal network representation. We are now focusing on another problem suggested by the strong requirement of supporting an interactive process, i.e. that of developing incremental algorithms. It is an interesting and open field of research to provide tools for dynamic updating of a constraint network and for checking the constraint satisfiability [17, 6]. Two lines of research are under investigation; first, to develop incremental graph based algorithms for shortest paths; second, to separate the satisfiability of the network from the search for the minimal network representation. In this second approach one can use an algorithm for dynamically maintaining the loops in the constraint graphs for the first task, and arc consistency for dynamic networks for the second task [17, 5, 20].

References

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communication of ACM*, 26(11):832–843, 1983.
- [2] P. Avesani, A. Perini, and F. Ricci. Combining CBR and constraint reasoning in planning forest fire fighting. In *Proceedings of the First European Workshop on Case-Based Reasoning*, pages 235–239, Kaiserslautern, 1993.
- [3] P. Avesani, A. Perini, and F. Ricci. The intervention planning subsystem. Technical report, IRST, 1995. CHARADE Restricted Report #50B.
- [4] F. A. Barber. A metric time-point and duration-based temporal model. *SIGART Bulletin*, 4(3):30–49, 1993.
- [5] C. Bessière. Arc-consistency in dynamic constraint satisfaction problems. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.
- [6] R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks. In K. Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, pages 196–201. 1994.

- [7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. The MIT Press, 1990.
- [8] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.
- [9] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49, 1991.
- [10] A. Gerevini and M. Cristani. Reasoning with inequations in temporal constraint networks. Technical report, IRST, Trento, Italy, 1995.
- [11] K. J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, Boston, 1989.
- [12] T. R. Hinrichs. Towards an architecture for open world problem solving. In J. L. Kolodner, editor, *Proceedings of the 1988 DARPA Workshop on Case-Based Reasoning*, pages 182–189, 1988.
- [13] O. Lhomme. Consistency techniques for numeric CSPs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chambéry, France*, pages 232–238, 1993.
- [14] P. Marti. The interface design of an integrated system for handling environmental emergencies. In *Cognitive Ergonomics and User-Centered Design*, Padova, Italy, Mar. 1995.
- [15] I. Meiri. Combining qualitative and quantitative constraints in temporal reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 260 – 267, 1991.
- [16] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.
- [17] B. Neveu and P. Berlandier. Arc-Consistency for dynamic constraint satisfaction problems: an rms-free approach. In *ECAI94 Workshop, Constraint Satisfaction Issues Raised by Practical Applications*. 1994.
- [18] V. Normand. Task modelling in the design and implementation of interactive systems. In *Proceedings of the 5th Conference on the Engineering of Man-Machine Interfaces*, Lyon, France, Oct. 1993.
- [19] A. Perini and F. Ricci. Constraint reasoning and interactive planning. In *Workshop on Constraint Languages-Systems and Their Use in Problem Modelling - International Logic Programming Symposium*, Ithaca, New York, 1994.
- [20] P. Prosser, C. Conway, and C. Muller. A distributed constraint maintenance system. In *Proceedings of Expert Systems and their Applications*, Avignon, France, 1992.
- [21] F. Ricci and P. Avesani. Learning a local similarity metric for case-based reasoning. In *International Conference on Case-Based Reasoning (ICCBR-95), Sesimbra, Portugal, Oct. 23-26, 1995*, Oct. 1995.
- [22] F. Ricci, P. Avesani, A. Perini, and M. Stocchero. Approaches and techniques for a.i. planning. Technical Report TR9307-32, IRST, 1993. CHARADE Restricted Report #51.
- [23] F. Ricci, S. Mam, P. Marti, V. Normand, and P. Olmo. CHARADE: a platform for emergencies management systems. Technical Report 9404-07, IRST, 1994.
- [24] F. Ricci, A. Perini, and P. Avesani. Planning in a complex real domain. In *Proceedings of the Italian Planning Workshop*, pages 55–60, Rome, 1993.
- [25] J. M. Rousseau and V. Normand. Task modelling: a common framework for the ergonomist and the computer scientist. In *Proceedings of ERGOIA*, Biarritz, France, Oct. 1994.
- [26] E. Schwalb and R. Dechter. Processing temporal constraint networks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.
- [27] W. Swartout. DARPA Santa Cruz workshop on planning. *The AI Magazine*, 9(2):115–131, 1988.
- [28] D. E. Wilkins and R. V. Desimone. Applying an a.i. planner to military operation planning. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*. Morgan-Kaufmann, San Mateo, 1992.