Interactive Case-Based Planning for Forest Fire Management

PAOLO AVESANI, ANNA PERINI AND FRANCESCO RICCI*
Istituto per la Ricerca Scientifica e Tecnologica, 38050 Trento, Italy
avesani@irst.itc.it
perini@irst.itc.it
ricci@irst.itc.it

Abstract. This paper describes an AI system for planning the first attack on a forest fire. This planning system is based on two major techniques, case-based reasoning, and constraint reasoning, and is part of a decision support system called CHARADE. CHARADE is aimed at supporting the user in the whole process of forest fire management. The novelty of the proposed approach is mainly due to the use of a local similarity metric for case-based reasoning and the integration with a constraint solver in charge of temporal reasoning.

Keywords: case-based reasoning, planning, local similarity metric, learning, temporal reasoning, constraint satisfaction

1. Introduction

This paper presents an interactive planning system that has been embedded in an intelligent decision support system aimed at supporting firemen in the whole process of fire fighting, including situation assessment and planning activities.

Classical AI approaches to planning rely on the view that the problem solving activity is predominantly autonomous. This simplified perspective doesn’t apply in the fire fighting domain. In some cases, the user is able to solve his current problem, for example, mostly using the strategical level. In other cases, the user wants to constrain the search process. Decisions are always made by the user using the computer together with other tools and usually following a complex operational flow. The above mentioned issues stress various limitations of classical AI planning techniques (for a more extensive discussion, see [1]), in particular of those developed in the context of pure formal approaches to planning.

An interactive planning approach driven by domain requirements and constraints that rests on the integration of case-based reasoning [2] and constraint reasoning [3] was developed. In this approach, plans are represented with two components: a set of indices derived from situation assessment, also called predictive features, and a time referenced network of actions: this second plan component is the target in the retrieval step.

Case-based reasoning techniques used for retrieving partial plans from a case memory were enhanced with a novel similarity metric that has a local definition and is tailored to the case base by a reinforcement learning procedure. This new local metric is defined only on a subset of the original case base (prototypes). The remaining cases are used for adapting the local definition of the metric on the selected prototypes. One advantage derived from this approach is that it is possible to reduce the size of case memory still maintaining optimal accuracy while increasing the time response for case retrieval. Moreover, the learning method is robust against the selection strategy of prototypes and an almost random selection can be simply used.

Constraint reasoning techniques are exploited in the plan adaptation phase when constraints are attached to the retrieved plan and propagated. Constraint reasoning

*Present address: Sodalia, S.p.A, via V. Zambra 1, 38110 Trento, Italy. Email: ricci@sodalia.it.
is used mainly for managing temporal relations defined on sub-plans or between actions in plans. Constraints are also used in the identification of all the actions that could be potentially applicable in a given emergency scenario. The temporal dimension of the plan is managed with a quantitative approach based on temporal variables over continuous domains. Constraint reasoning also supports the resource allocation process for the chosen plan. We designed new and efficient temporal reasoning procedures, in particular a fast incremental algorithm for updating temporal constraint networks.

The paper is organized as follows. First, we describe the forest fire fighting domain, stressing its complexity. Then we illustrate the operational context where the CHARADE system is used and the global system architecture. In the following sections, we focus on the interactive planning problem and on the proposed solution. Section 3 describes the decision process related to the interactive planning problem. Section 4 illustrates the plan representation adopted. A more detailed view of plan retrieval and plan reuse is given in Section 5 and Section 6 respectively. The first describes the local similarity metric introduced and the second the efficient incremental algorithms for temporal reasoning developed. We conclude by reviewing a few related works that share with our system both the techniques (case-based reasoning) and the application domain (crisis response management).

2. Planning for Fire Fighting

The complexity of the forest fire domain comes from features that are typical of environmental problems. Fire is a dynamic phenomenon whose evolution is determined by weather conditions, in particular wind intensity and direction, by humidity, and by fuel type. These variables usually change rapidly and sometimes in an unpredictable way. Moreover, relevant fire events can happen on very different time and spatial scales, from seconds to days, from meters to kilometers, determining a large variety of world states. Data are always incomplete and uncertain, and, in some cases, totally absent.

Operational constraints often impose quick decisions that drastically limit the possibility to build an optimal plan. Sub-optimal solutions are often adopted and are strongly biased by the past experience.

The management of forest fires, as in general environmental emergencies, involves organizations that have decisional and operative centers distributed on the territory. So managing a forest fire can require several centers to cooperate. Moreover, complex coordination problems can arise when resources from different organizations, such as forestry, police, and physicians, are used.

These are all general features of the fire domain. To be more specific, the management of forest fires follows a precise operational work-flow that is typical of each fire fighting organization. Here we focus on the work organization in a French regional center.

2.1. The Operational Context

To understand how the user can be supported by the CHARADE system in planning an intervention, a typical session of crisis management is illustrated.

The user of the system is a fireman working in a regional center in charge of supporting local fire fighting. His tools are a workstation, a dedicated line to acquire data from infrared sensors and meteo sensors, a radio, a fax, a telephone and a printer.

The CHARADE system is running on a workstation and comprises a geographic information system (GIS), a graphical simulator of the fire evolution, tools for territorial, meteo and resource assessment, and a module for supporting the intervention planning and control.

When a new fire is reported, the alarm is promptly validated and the situation assessed by the user, who possibly runs a mathematical fire spreading model. On the screen, the operator can look at the output of the fire spreading model and access, through a graphical user interface, information on the graphical symbols shown by the map. At the end of this phase, the operator has acquired enough information for drawing on the map a number of line sectors that subdivide the original fire front.

This operation, called sectorization, involves crucial and strategical decisions: on which fire front to attack the fire; what specific technique to use on the selected fire fronts; how far from the fire epicenter to locate the resources; what kind of supplies need particular attention (for example inhabited houses, railway lines, etc).

Once the sectors have been identified on the map, the operator looks for a plan to fight the fire in each sector. The plan may use air forces and/or ground means, and adopt a specific course of action. Searching in a database of past sector plans, the system retrieves a set of plans that in a similar context worked successfully. Then the retrieved plans are modified and adapted to the current situation. These plans are evaluated by the
2.2. System Overview

The CHARADE system supports decision making in forest fire management, enhancing the cooperation of human and machine reasoning capabilities. The design of the interactive system functions has been performed following a task model of the forest fire management activities that was built upon a careful analysis of the operator activity conducted by following a task-oriented methodology [4]. This task model is embedded into the man-machine interface (MMI) subsystem and it is exploited to drive and control the overall dialogue by a task coordinator. In Fig. 1, both functional modules and data types are shown.

The situation assessment subsystem roughly implements a blackboard architecture. Different knowledge sources (fire spreading model, spatial reasoner, etc.) collaborate to produce the global description of the problem domain contained in the dynamic data space. Typically, knowledge sources are activated according to the work-flow determined by the task model included in the MMI.

The intervention planning subsystem integrates constraint reasoning and case-based reasoning techniques. The case-based reasoner plays the role of assumption maker: given a new emergency situation it suggests old solutions, adopted to similar situations in the past; the constraint reasoner filters those assumptions into a feasible solution.

The bottom layer depicts the static data space that contains the database of georeferenced cartographic data, the database of means and firemen squads assigned to the bases of a given region, the intervention plan memory and an action library that models the firefighting activities that are typically used by a firefighting organization for emergency management.

The user is always called to an active role, for example refining an hypothesis proposed by the case-based reasoner or browsing the constraint network. The following sections will give a more detailed view of the intervention planning subsystem.

The hardware configuration includes two monitors to support at the same time the interaction with alphanumerical and geographical information, as shown in Fig. 2.

3. Interactive Case-Based Planning

In this section we concentrate on the interactive planning functions and discuss how they match the problem solving process peculiar to emergency management.

Usually, in an AI-based planning system, 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 the forest fire domain.

A joint, human and machine, decision-making process while planning, needs to be considered. New interesting problems arise as that of making data and knowledge representation appropriate for both the user and the system.

In CHARADE the whole planning process was modeled along six main tasks that alternatively can be
performed by the user or the planner: plan retrieval, plan selection, plan reuse, plan refine, plan execution, and plan recording. Let us describe them in detail:

Plan retrieval. The first task is performed by the planner that by looking into past intervention memory provides the user with a set of candidate solutions that were applied in similar circumstances. A good similarity metric is crucial to provide an effective association between fire scenarios and firefighting plans.

Plan selection. Among the proposed past solutions, the user selects the candidate that better fits the current problem. Here many unstructured factors, too difficult to be modeled, contribute to the human decision.

Plan reuse. Once a past solution is selected, the planner sets up a new plan reusing that past plan. Requirements derived by actions inside the plan need to be combined with the constraints posted by the current situation. The result is a computer oriented representation of a firefighting plan, as described in the next Section.

Plan revision. Past solutions do not always fit the current problem and further adaptation is required to develop an effective intervention plan. The revision task is accomplished by interleaving user and planner activities: the user provides new action specification or qualitative temporal constraints, while the planner checks the consistency of the whole intervention plan. The plan applicability is checked up on problems that can arise in the resource allocation phase.

Plan execution. The complexity derived from the highly dynamic and unpredictable domain of forest fires makes plan verification very difficult by simulating plan execution. This task is done in the real world or is submitted to the evaluation of an expert.

Plan recording. At the end of the whole planning cycle the user can decide whether to store the current case, both the fire scenario and the firefighting plan, in the case memory. Moreover, the planner itself filters out some cases. That is done for efficiency and accuracy.

The next sections focus on plan retrieval, plan reuse, and plan refinement. For each step, it is shown how case-based and temporal constraint reasoning techniques were extended to efficiently support forest firefighting.

4. Plan Representation

The system uses two different plan representations: one for the plan that is developed in response to a given crisis, that is, the active plan, another for the plan stored in the plan's memory. Here, we refer to the active plan.
and describe how we dealt with the critical requirement of making the plan representation sharable between the two main actors of the problem-solving process: the system’s user and the planner.

The solution proposed in CHARADE rests basically on the following points. First, we used a rich representation for an intervention plan that points to data dynamically produced by the situation assessment component and to static, domain dependent information, represented in the action library. This information can be browsed by the user by specific MMI components. Second, we adopted two different representations for the plan temporal constraints, one user-oriented, the other at use of the system, and provided a mechanism for the automatic translation from one representation to the other. Moreover, we casted in a single plan representation the possibility of reasoning following different firefighting schema that range from planning the intervention for the global fire to planning the intervention for a single fire front, and gave the possibility to switch from one scheme to the other.

An active plan in CHARADE is represented as a hierarchy of plan components. The root represents the global intervention plan that can be composed of different plans, one for each of the fire front sectors on which the user intends to act. The leaves correspond to single actions. This representation gives a uniform description of the different abstraction levels at which planning is performed, such as planning the intervention on one or more fire front sectors or reasoning on a global plan that includes one or more (up to four) sectors or checking the applicability conditions and dimensioning actions.

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 Fig. 3.

A plan component contains a set of domain dependent information, such as data resulting from the assessment of the fire alarm, that are coded into the scenario component of the action.net structure. Different data are relevant at different abstraction levels. So, for instance, the scenario at the action level contains information specific to a fire fighting technique such as the type and the number of resources to be used, the types of actions that can or must be performed together to be effective (for instance spraying retardant in order to lower the fire intensity where squads will operate), constraints that model the domain expertise on action dimensioning (i.e., relations between the amount of a specific resource and the amount of work that can be accomplished in a given time period). The scenario at the sector plan level contains spatial information (sector length and orientation, accessibility, availability of water reservoirs near the sector, type of vegetation), fire’s physical parameters (fire intensity, flame height, spreading direction and speed), information on the availability of resources located in the bases closest to a sector.

The temporal information relative to a plan component is coded in a temporal constraint network associated to its action.net. The variables of this temporal network are the start and the end time points of the actions, of the sector and the global plans. The constraints are those representing the minimum duration of actions and those representing the interval relations between actions (for instance, the precedence relation). Moreover, the part-of relation between the sector plan and each one of its actions induces a pair of
temporal constraints between the sector’s start and the action’s start and respectively between the action’s end and the sector plan’s end (sector\textsubscript{start} – action\textsubscript{start} $\leq$ 0, action\textsubscript{end} – sector\textsubscript{end} $\leq$ 0). Analogous constraints exist between the start and the end of the global plan and the start and the end of the sector plans it is composed of.

These constraints were modeled in terms of bounded difference between time points. This defines a constraint satisfaction problem (CSP) on variables with continuous domains [3], the so called Simple Temporal constraint satisfaction Problems (STP).

Constraints on time points allow representing and reasoning about quantitative information, as discussed in Section 6, 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 is represented at two levels.

1. The user level: Here, time intervals are considered and constraints between time intervals in the form of single Allen’s interval relations [5] are handled. 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.

2. The planner level: Here, time points are defined and time constraints representing bounded differences between time points are implemented. The interval relations are mapped to a set of metric constraints representing bounded differences between the interval endpoints. An example is given in Fig. 4.

Figure 5 shows a partial snapshot of the MMI intervention plan editor. This snapshot highlights how the temporal structure of the plan is presented to the user. An action is represented by a rectangular box with its left side corresponding to the action’s earliest start time and the right side to the latest start time. The inner area filled in gray shows the action’s minimum duration. Lines between boxes correspond to temporal relations between actions.

The plan’s hierarchical structure is exploited for decomposing the temporal reasoning problem underlying

![Figure 4](image1.png)

**Figure 4.** Examples of correspondence between interval relations and bounded difference constraints between the points representing the start time and the end time of actions.

![Figure 5](image2.png)

**Figure 5.** A snapshot of the plan editor. The plan’s temporal information is depicted.
the interactive planning process. That allows an 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.

Plan management functions supporting reasoning tasks specific to each different abstraction level are associated with the action.net structure.

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. Plan Retrieval

The goal of the retrieval step is to find a past firefighting plan to be used as a starting point for a new planning phase. This provides the user with an example of a possibly applicable strategy.

This goal can be achieved by considering the retrieval step as a classification problem. Given a set of examples, each one belonging to one class in a given finite set of classes, a classification problem consists of assigning the right class to a new example whose class is unknown. To solve this problem, a set of features (predictive) is used to derive the correct value of an unknown target feature, usually called class.

In the forest firefighting domain, the classes are the main strategies: ground attack, mixed attack and aerial attack. The first strategy, ground attack, is applied when aerial means can’t be deployed; in aerial attack, the intervention can be managed only by aerial means; in mixed attack, usually a firefighting plan involves both kinds of means. The features describing the fire scenario play a predictive role in finding a viable strategy for a new case. The basic idea we use to solve the classification problem and the initial plan generation is to search in the memory for cases that are similar to the one we want to solve and reuse one of those plans that was used in similar situations. Similarity is computed comparing the (predictive) features of the current emergency scenario with those of the stored one.

In other words, the retrieval step is based on the nearest neighbor algorithm (NN), a well-known technique to solve a classification problem [6].

Given a new probe emergency case, for each case in the memory, a similarity-based distance function is computed to obtain the relative distance from the probe. A probe has a partial description of the fire case where only the fire scenario is given while the definition of the plan strategy is still missing. The nearest case in the memory provides a candidate strategy to be used as a viable solution for the probe.

It is clear that the similarity metric is a key element in the design of case-based system based on NN. Usually, global metrics, which are generalizations of the Euclidean metric, are exploited. In this case, if cases $x$ and $y$ are described with $n$-dimensional vectors of numerical features $(x_1, \ldots, x_n)$, $(y_1, \ldots, y_n)$ another vector of weights $w = (w_1, \ldots, w_n)$ is used to balance the contributions of the features to the total distance: $d(x, y) = (\sum_{i=1}^{n} w_i |x_i - y_i|^2)^{1/2}$. Global metrics have many limitations but essentially don’t capture the correlation among features and the variability of feature relevance in different parts of the input space. For that reason, local metrics have been introduced [7].

A local metric is a metric that depends on the point in the input space from which the distance is taken $d(x, y) = (\sum_{i=1}^{n} w_i(x) |x_i - y_i|^2)^{1/2}$. Local metrics are context-sensitive, that is, similarity between cases depends also on the absolute value of feature values. Using a local metric, it is possible to express conditional expressions like, “if the accessibility feature is greater than 1000 meters, don’t use the orography feature to compute the similarity”.

Figure 6(a) illustrates the notion of a local similarity metric. A local metric is depicted as a box centered on the feature value and with width equal to the unit distance on the accessibility axis.

![Figure 6. Local weights and asymmetric metrics.](image-url)
Case x#12 describes a case with accessibility value 800. Accessibility is the distance between the fire front and the nearest road. The local weight value of x#12 is 0.01; this implies that, along that feature, a unit distance separates x#12 from cases with accessibility value 700 or 900.

Let us suppose, as depicted in Fig. 6, that the accessibility value is used to discriminate among three classes of intervention: ground attack, mixed attack and aerial attack. For each strategy, a different set of actions can be combined to obtain a specific tactic. Usually, when it is easy to reach the fire front, (accessibility <500 meters), a ground attack is chosen while when the distance is greater than 1000 meters, only an aerial attack can be effective. Our case memory and the associated local metric must incorporate this knowledge. If a new probe case has accessibility value 700, the nearest neighbor found using the local similarity metric shown in Fig. 6(a) is x#12. Therefore, the fire fighting plan associated with x#12 is proposed, a plan based on a mixed attack strategy.

Using weights in distance computation poses an additional problem. Weights must be precisely chosen to optimize system performance. We have proposed in [8] a learning procedure for computing features’ relevance in a local metric. This procedure uses a reinforcement learning scheme and improves the retrieval accuracy of the initial local metric with all the weights equally set to the same value.

Normally, both local and global metrics on real features obey to the following property \( d(x, x + \Delta) = d(x, x - \Delta) \). That means that the metric is invariant with respect to the inversion of the direction of the displacement. Adapting weights in a metric with that property may lead to some difficulties. For instance, given three cases \( x, y, \) and \( z \), if a case \( x \) should be made closer to \( y \) and more distant from \( z \), there are cases in which one cannot attain that goal only changing the local metric attached to \( x \).

A greater flexibility is obtained if, for each feature a pair of weights is used, one for the “left” direction and one for the “right” direction [8]. In the example quoted above, if the metric attached to \( x \) is modified in such a way that the space in the direction \( y - z \) is compressed and in the direction \( z - y \) expanded, the effect is that \( y \) comes closer to \( x \) and \( z \) goes farther from \( x \). This cannot be achieved with a normal weighted metric.

In the following section, we give a more formal account of local metrics, and we describe the learning procedure used for learning weights.

### 5.1. Local Weights and Asymmetric Metrics

The features used to describe a fire scenario can be categorical or real. In the first case, feature value varies in a set of symbols; in the second case, it is normalized to be a number in the closed unit interval \([0, 1]\).

Let \( F_j \) be a generic feature space, i.e., a finite set or the closed unit interval, then \( \prod_{j=1}^{n} F_j \) denotes the input space and \( x \in \prod_{j=1}^{n} F_j \) a generic example. Each space \( F_j \) is endowed with a feature metric \( d_j : F_j \cup \{?\} \times F_j \cup \{?\} \to \mathbb{R}_{\geq 0} \), where ? is a special symbol denoting an unknown value in \( F_j \):

\[
\begin{align*}
d_j(x_j, y_j) = \\
\begin{cases}
|x_j - y_j| & \text{if } x_j, y_j \in [0, 1] \\
0 & \text{if } F_j \text{ is categorical and } x_j = y_j \\
1 & \text{if } F_j \text{ is categorical and } x_j \neq y_j \\
0.5 & \text{if } x_j \text{ or } y_j \text{ is unknown}
\end{cases}
\end{align*}
\]

A set of asymmetric weights \( w(x) \) for \( x \) is a \( 2 \times n \) matrix with values in \([0, 1]\). Let \( w_j^f(x), \ x = 0, 1 \) and \( j = 1, \ldots, n, \) be a generic element of \( w(x) \). We assume that \( w_j^f(x) = w_j^f(x) \) if \( F_j \) is a set of symbols. Let \( y \) be another point in the input space, if \( F_j = [0, 1] \) then the following notation will be adopted:

\[
w_j(x) \cdot d_j(x_j, y_j) = \begin{cases} w_j^f(x)d_j(x_j, y_j) & \text{if } y_j \leq x_j \\ w_j^f(x)d_j(x_j, y_j) & \text{otherwise} \end{cases}
\]

Conversely, when \( F_j \) is a set of symbols \( w_j(x) \cdot d_j(x_j, y_j) = w_j^f(x)d_j(x_j, y_j) = w_j^f(x)\delta_j(x_j, y_j) \).

Given a case base \( CB \) and a set of asymmetric weights for each example in \( CB \), a local asymmetrically weighted metric (LASM) is defined as follows:

\[
d : CB \times \prod_{j=1}^{n} F_j \to \mathbb{R}_{\geq 0}
\]

\[
d(x, y) = \left( \sum_{j=1}^{n} w_j(x) \cdot d_j(x_j, y_j)^2 \right)^{1/2}
\]

An example of such a metric is illustrated in Fig. 6(c). For sake of simplicity here, only a feature dimension is shown, that associated with the accessibility. The metric defined in case x#12 is defined by two weights \( w^0 \) and \( w^1 \). \( w^0(w^1) \) is used when computing the distance between x#12 and another case with accessibility less (greater) than 800.
5.2. The Learning Procedure

Let us now focus on the learning procedure we use to compute weight values. It is based on a reinforcement learning scheme and iteratively transforms the weights attached to a set of cases in the case memory using feedback given by the evaluation of the classification response of the NN classifier.

The target problem is to find an approximation of the target function $Plan: \prod_{j=1}^{n} F_j \rightarrow S$, where $S$ is the set of firefighting strategies (classes). The $Plan$ function is known only on a sample $C \subset \prod_{j=1}^{n} F_j$ and we want to learn its behavior on the whole input space.

The objective of the learning task is twofold: acquiring the weight values and at the same time reducing the size of the case base. Since naive NN algorithms have a linear time complexity with respect to the size of the case base, using a subset of the case base (prototypes) yields an improvement in time performance.

Given a selection of prototypes $CB \subset C$, where $C$ is the complete set of cases for which a correct plan is known, our goal is to find a system of weights for $CB$ that maximizes the accuracy of the nearest-neighbor algorithm, that is, \[ E[Plan(nn(x)) = Plan(x)] \] is maximum, where $nn(x) \in CB$ is the nearest neighbor of $x \in \prod_{j=1}^{n} F_j$.

To attain this goal, we proposed in [8] a training procedure that, given an example $p$ in $C$ but not in $CB$, computes the value of $Plan$ on $p$, which is known, with the value of $Plan$ on the nearest neighbor of $p$ in $CB$. If the two values are equal, then the prediction is correct, and the distance between the nearest neighbor and $p$ is decreased; whereas, if the two values are not equal, the distance between the nearest neighbor and $p$ is increased (see also [9–11] for other applications of this learning strategy). Our approach is similar to the competitive learning technique [11] or learning vector quantization [9]. But, while in these approaches the cases stored are moved in the input space, we instead change the metrics attached to the stored cases, and therefore, directional information has to be taken into account.

For instance, consider the example in Fig. 6(d). It is simple to show that if the 18th and the 18th cases are given, to obtain the maximal accuracy storing only an additional case, an abstract case with accessibility value equal to 750 would be needed. But that fire scenario could not belong to the real past experience. In a real domain, it is not advisable to create these abstract cases; they could even be impossible due to other domain constraints.

To describe this process in a more formal way, let $CB = \{x_1, \ldots, x_m\}$ be a subset of cases where $(|CB|)$ is normally taken as 10% of $C$. The prototypes $x_i$ are selected at random but the percentage of prototypes in a given class is equal to the percentage of cases in that class that were found in the original sample $C$. A learning step is a pair of functions $R$ and $P$ that map a set of weights $w(x_i)$ for $x_i$, the example $x_i$ and a testing example $y$ in a new system of weights $R(w, x_i, y)$ and $P(w, x_i, y)$ for $x_i$. $R$, the reinforcement step is chosen if the value of $Plan$ on $y$ is equal to the value of $Plan$ on $x_i$. If this is not the case, $P$, the punishment step is used. A learning procedure iterates that step adjusting an initially constant system of weights for $CB$, aiming to optimize classification accuracy. Let $x_i \in CB$ be the nearest neighbor of $y$ then the definition of $R$ and $P$ is shown in Fig. 7.

$\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are called the reinforcement and punishment rate respectively. Note that each learning step updates at most $n$ parameters and it can be shown that it maintains the weights in $[0, 1]$ [12].

The punishment function is chosen in such a way that the punishment, i.e., the value used to decrease $w_{ij}^k$, is equal to $\beta w_{ij}^k d_j(x_{ij}, y_j)$ for $0 \leq w_{ij}^k \leq 1/2$, and it is equal to $\beta (1 - w_{ij}^k) d_j(x_{ij}, y_j)$ for $1/2 \leq w_{ij}^k \leq 1$. This behavior is needed to maintain $0 \leq w_{ij}^k \leq 1$ when punishment is applied.

Finally, note that if $x_{ij}$ or $y_j$ are unknown values and $F_j$ is a numeric feature, it is not possible to determine whether $y_j < x_{ij}$ so the maps $R$ and $P$ choose randomly which weight, among $w_{ij}$ and $w_{ij}^k$, to update.

Figure 8 shows the pseudo code of the learning procedure that takes as input a learning step map, a discrete function $Plan$, a sample $C$, a case base $CB$, a system of weights for $CB$, and terminates, returning a new system of weights for the original case base.

In the Learn-Weights procedure, four auxiliary procedures are used.

- Initialize-Weights($w$): initializes all weights equally (uniform deformation of the Euclidean metric).
- Generate-Probe($C \setminus CB$): cyclically extracts all the elements in $C \setminus CB$.
- Retrieve($y$, $CB$): finds the nearest neighbor of $y$ in $CB$ measuring the distances between $y$ and points in $CB$ with the current local metric definition.
- Exit-Condition: stops the loop when the accuracy on $C \setminus CB$ decreases two times consecutively in each pass through $C \setminus CB$. Accuracy is computed as $nr/(nr + np)$, where $nr$ and $np$ are the number of
If $\text{Plan}(x_i) = \text{Plan}(y)$ then:

$$R_{ij}^0(w_{ij}^0, x_i, y_j) = \begin{cases} w_{ij}^0 - \alpha w_{ij}^0 d_j(x_{ij}, y_j) & \text{if } y_j < x_{ij} \\ w_{ij}^0 & \text{if } x_{ij} \leq y_j \end{cases}$$

$$R_{ij}^1(w_{ij}^1, x_i, y_j) = \begin{cases} w_{ij}^1 & \text{if } y_j < x_{ij} \\ w_{ij}^1 - \alpha w_{ij}^1 d_j(x_{ij}, y_j) & \text{if } x_{ij} \leq y_j \end{cases}$$

if $F_j = [0, 1]$. Otherwise if $F_j$ is a set of symbols:

$$R_{ij}(w_{ij}, x_i, y_j) = w_{ij} - \alpha w_{ij} d_j(x_{ij}, y_j)$$

If $\text{Plan}(x_i) \neq \text{Plan}(y)$ then:

$$P_{ij}^0(w_{ij}^0, x_i, y_j) = \begin{cases} w_{ij}^0 + \frac{\beta}{2} (1 - |2w_{ij}^0 - 1|) d_j(x_{ij}, y_j) & \text{if } y_j < x_{ij} \\ w_{ij}^0 & \text{if } x_{ij} \leq y_j \end{cases}$$

$$P_{ij}^1(w_{ij}^1, x_i, y_j) = \begin{cases} w_{ij}^1 & \text{if } y_j < x_{ij} \\ w_{ij}^1 + \frac{\beta}{2} (1 - |2w_{ij}^1 - 1|) d_j(x_{ij}, y_j) & \text{if } x_{ij} \leq y_j \end{cases}$$

if $F_j = [0, 1]$. Otherwise if $F_j$ is a set of symbols:

$$P_{ij}(w_{ij}, x_i, y_j) = w_{ij} + \frac{\beta}{2} (1 - |2w_{ij} - 1|) d_j(x_{ij}, y_j)$$

Figure 7. The definition of the reward and punishment functions.

**Algorithm 1:** Learn-Weights($w, R, P, C, f, CB$)

1. Initialize-Weights($w$)
2. while not Exit-Condition
   1. $y \leftarrow$ Generate-Probe($C \setminus CB$)
   2. $x \leftarrow$ Retrieve($y, CB$)
   3. if $\text{Plan}(x) = \text{Plan}(y)$
      1. then $w(x) \leftarrow R(w(x), x, y)$
      2. else $w(x) \leftarrow P(w(x), x, y)$
   4. endif
3. endwhile
4. end Learn-Weights

Figure 8. The Learn-Weights procedure.

reinforcements and punishments respectively, in a pass through $C \setminus CB$.

Regarding the compression rate in general, it depends on the specific case memory. But applying this method on a number of different case bases, we discovered that on average, less than 10% of the original sample cases $C$ still provides the same accuracy given by the $k$-NN algorithm ($k$ optimized with cross validation [13]). As a consequence, query time can be reduced by approximately the same rate. In other words, with our approach, part of the knowledge contained in the data is moved to the local metric.

In the next section we describe how, starting from an initial retrieved plan, the CHARADE system is used to derive a more suitable plan for the current situation.

6. Plan Reuse and Revision

In a case-based planning approach, a candidate plan retrieved from the past intervention memory needs to be adapted to the current scenario in order to yield a viable solution. Different approaches to case adaptation are discussed in [14, 15], and a basic consideration that can be drawn is that this task needs to be carefully designed for a given application, exploiting specific domain heuristics and available reasoning models.

So for instance, when a generative problem solver is available, the adaptation step can be performed by
using the retrieved solution to guide the generative problem solver while searching a solution rather than directly modifying the retrieved one.

This approach called generative adaptation [14] is not suitable for intervention planning in the firefighting domain that lacks a generative problem solver.

In CHARADE, the adaptation task is performed in two steps:

Plan reuse. As described in Section 4, the retrieved plan includes a set of constraints on time and resource variables that needs to be reinstantiated in the constraint network associated with the active plan. These constraints are automatically propagated and checked against other descriptive variables of the current situation. For instance, if the candidate plan contains the action of spraying water by employing two squads, and, in the current emergency, the situation assessment calls for a greater amount of water to be sprayed, the number of squads associated with the action will be recomputed.

Plan revision. A further adaptation phase is done through an interactive process of editing activities performed by the user, like inserting/deleting actions, modifying action durations or temporal relations between parts of the plan, and allocating resources. At this level, plan adaptation exploits constraint reasoning techniques for modeling the constraints on actions and checking their consistency. In particular, this paper focuses on the temporal constraints of a plan. For example, a plan for controlling the fire on a given sector can be impractical if the time required to perform it is greater than the deadline posed by the fire propagation toward that sector.

The exploitation of constraint satisfaction techniques to adapt the candidate solution in the plan reuse and in the plan revision steps is becoming more and more common [16, 17].

In the following subsections, we focus our attention on plan revision, a critical step in CHARADE since it rests on an interactive process.

First, we describe the system functions for interactive plan revision. Each function has been carefully designed in order to minimize computational cost. For instance, efficiency improvement has been obtained localizing constraint reasoning to components of the constraint network according to the plan hierarchical structure described in Section 4.

Then we will describe the temporal reasoning techniques exploited by these functions.

6.1. System Functions

This section gives a description of the main functions that can be called by the user during plan revision, pointing out the specific kind of temporal reasoning computations underlying each function call. An extended description of these functions can be found in [18, 19].

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 defined during the situation assessment phase, 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.

For instance, if the restored plan corresponds to the plan in Fig. 9(a), the current flame height at the head sector makes a direct ground attack hard to perform, and building a fire line would be advisable; the user decides to insert the action building a fire line before the action direct ground attack. This can cause a deadline violation, as shown in Fig. 9(b), that can be managed by either attempting to delay the plan deadline or shortening the action duration allocating more squads on the actions.

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 cannot produce inconsistency. The feasible time intervals of the network variables can increase. So the variables domain must be set to their default domains and the feasible times recomputed.
Figure 9. A plan, described as an action graph. The nodes, represented by gray boxes, correspond to actions; the edges, labeled by ovals, to the temporal relations between actions. (a) The plan restored in the time window between 14:30 and 16:30. Here each action is described by two overlapped rectangles: one, external, defined by the earliest start time and the latest end time of the action; the internal one, filled in gray, by the earliest start time and the earliest end time. (b) Trying to insert a new action in plan (a). (c) Removing the temporal relation before between two actions of plan (a).

For instance, deleting the temporal relation before between the actions Retardant by helicopter and Water by plane increases the feasible interval of the start time of Water by plane. This yields a greater flexibility of these actions’ execution with respect to the global plan deadline (Fig. 9(c)).

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 an action’s start and end are still good values. These intervals must be updated, computing new consistent values. When trying to increase the minimum duration, the consistency has to be checked against sector deadline violation.

Automatic allocation on the sector plan. 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 heuristically orders the actions to be allocated and schedules them, setting the action start time to the earliest possible time that is updated according to the already scheduled actions. This corresponds to computing a solution (or completing a partial one) for the sector temporal network.

6.2. Temporal Reasoning

As pointed out in the above discussion, the addition (or deletion) of constraints to (or from) the constraint network associated with the current plan during the
adaptation step prompts two basic reasoning tasks: first, determining the consistency of a given set of constraints (i.e., checking that at least one value assignment to the variables of this constraint network exists), and second, computing a "window" of feasible times for each involved temporal variable. Moreover, resource allocation calls for finding a consistent assignment for all the network variables (i.e., a solution of the constraint network).

In many AI planning and scheduling applications these are "on-line" tasks that require computational efficiency [20, 21]. Our approach to this problem is based on a careful analysis of the relationship between shortest-paths algorithms for directed weighted graphs and arc-consistency techniques for constraint networks coded as Simple Temporal Problems (STP). Using that relationship we have designed efficient algorithms for the incremental updating of a subclass of STP [22].

6.2.1. Simple Temporal Problems. Simple temporal constraint satisfaction problems, also called STP, were first introduced in [3] as constraint satisfaction problems with real-valued variables and constraints expressed in terms of temporal distances.

An STP network \( T = (V, C) \) represents a set of constraints \( C \) of the form \( y - x \in I \), where \( x \) and \( y \) are point-variables in a set \( V \), and \( I \) is an interval in the time domain [3].

Well-known properties of STP networks are that both consistency checking and computing the "minimal network" representation\(^2\) can be performed in \( O(n^3) \) time [3, 23]. Moreover, minimal networks are 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. In [3] it is shown that checking the consistency of a STP network leads to dealing with a shortest-paths problem for a directed graph.

The distance graph \( G = (V, E) \) associated with an STP network \( T \) is a directed weighted graph having the same vertices as \( T \) and a pair of edges \((x, y)\) and \((y, x)\) labeled \(-a\) and \(b\) respectively, for each constraint \( y - x \in [a, b] \) in \( C \) [3]. It can be proved that if \( s \) is a temporal variable constrained to be the time origin (i.e., \( s = 0 \)), then for each variable \( x \) the earliest time and the latest time are \(-d^s(x)\) and \(d^s(x)\) respectively, where \(d(x)\) and \(d'(x)\) are the shortest-path distances from \( s \) to \( x \) in \( G \) and in the transpose graph \( G^t = (V, E^t) \).

Therefore, consistency checking and computing the feasible times can be achieved by using a single-source shortest-paths algorithm such as the algorithm by Bellman and Ford (BF) [24]. Furthermore, if the graph is acyclic then a more efficient shortest-paths algorithms specialized for directed acyclic graphs can be used [25].

The BF algorithm can be considered a particular arc-consistency algorithm that does not suffer the termination problem which may aect known algorithms for enforcing arc-consistency in continuous domains CSP. This problem consists of the possibility of generating an arbitrarily long succession of edge revisions [26, 27]. We call this version of the algorithm the AC-BF algorithm and show it in Fig. 10.

The procedure "Refine(\(u, v\))" in AC-BF(T), used by traditional arc-consistency algorithms [26, 27], eliminates from the current domain of \( u \) those values that are not compatible with the current domain of \( v \) according to the constraints between \( u \) and \( v \) of \( T \). The following theorem for AC-BF has been proven in [24].

Theorem 1. Let \( T = (V, C) \) be a simple temporal constraint network, AC-BF(T) enforces arc-consistency on \( T \) in \( O(|V||C|) \) time. Moreover, \( T \) is consistent if AC-BF(T) returns TRUE, and if \( T \) is consistent, then AC-BF(T) computes the earliest time and the latest time for each variable in \( V \).

The interactive editing functions described in 6.1 were supported by temporal reasoning procedures based on arc-consistency and on the AC-BF(T) algorithm. Moreover, the resource allocation function is supported by a procedure that exploits decomposability property of the minimal network of an STP. This property builds 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.

6.2.2. Incremental Algorithms for STP⁻. We now introduce a subclass of STP, called STP⁻, consisting of the STP-constraints of the form $x_j - x_i \leq a$, where $a \leq 0$. STP⁻ has an appealing feature: the distance graph of a set of STP⁻-constraints can be incrementally managed through efficient algorithms.

Figure 11 shows an example of a plan whose temporal dimension can be represented as a simple temporal negative constraint problem (see also [18]). Actions are represented with temporal intervals that are constrained to be within a time span given by two temporal variables, the source $S$ and the end $F$. Constraints on the minimal duration can be expressed. Even if we cannot express explicitly a constraint on the maximal duration of an action, we can place deadlines. The expressiveness of STP⁻ is discussed more deeply in [24].

Here we point out the ideas exploited in the incremental updating algorithms. A detailed description is given in [24]. The dynamic management of STP⁻-constraints is based on a special data structure that we called the “distance metagraph.” The distance metagraph is derived from the distance graph of a given STP (or STP⁻) whose vertices are maintained topologically ordered upon addition/deletion of constraints. This data structure allows the applicability of known directed acyclic graph (DAG) shortest-path algorithms the extension of to the distance graph of a STP⁻.

The distance metagraph of a distance graph, $G^*$, is a directed acyclic graph in which sets of equivalent vertices are “collapsed” into a single vertex forming a metavertex. Two vertices $x$, $y$ of a distance graph $G$ which does not contain negative cycles are equivalent (written as $x \sim y$) if and only if there exists in $G$ a cycle connecting $x$ and $y$ whose length is zero.

Since $G^*$ is acyclic, its shortest paths can be computed by using a shortest-paths algorithm for DAGs such as DAG-Shortest-Paths given in [25]. DAG-Shortest-Paths takes $O(|E|)$ time and space, and thus it is more efficient than the Bellman-Ford algorithm requiring $O(|V||E|)$ time. Insertion (or deletion) of constraints is performed by checking possible modifications (inconsistency, expansion) of the $G^*$ cycles and eventually updating the topological order of $G^*$’s vertices before running DAG-Shortest-Paths algorithms that updates the graph’s shortest-paths.

An experimental comparison of the proposed algorithms with the (non-incremental) BP algorithm showed that drastic CPU-time reductions can be obtained [24].

7. Related Works in Crisis Management

Crisis management poses challenging issues to be investigated by AI researchers. One of the first attempts to approach the problem of the real-time management of forest fires is the Phoenix project [28]. Phoenix addresses two research objectives: the design of complete autonomous agents and the constraints that a complex dynamic environment places on the design of intelligent agents.

Several works in crisis management are reported in the review conducted in the Esprit Project NOW [29]. Moreover, the interest in these research topics was promoted also by Darpa projects [30, 31], where
specific issues such as distributed planning [32], planning in dynamic environments [33] and mixed-initiative planning [34] are still open problems. The last point is indeed a key aspect faced in the work here discussed.

In the rest of this section, we examine a few examples where CBR techniques have been applied to crisis management problems.

One of the first attempts to use case-based reasoning to approach the problem of strategical planning is Battle Planner [35]. The domain is the design of military operations. Battle planning shares with firefighting the feature that a complete causal model is not available. Furthermore, in both cases, the user usually designs new solutions starting from his experience. In Battle Planner, the author prefers an inductive discrimination analysis to nearest neighbor. He or she is mainly concerned with the problem of finding the right splits for numerical features. That problem is solved manually case by case.

DIAL [36] is another example of case-based planning in the disaster response domain. The system objective is to generate plans to guide damage assessment, evacuation, etc. DIAL uses case-based reasoning both for similarity assessment during plan retrieval and for the adaptation of the retrieved plans. In DIAL, the adaptation knowledge isn't static and the adaptation experiences acquired from the plan refinement performed by the user is learned. A new case-based reasoning problem arises: how to adapt adaptation cases and how to learn similarity criteria based on adaptability.

A more recent planning architecture was proposed in INCA [37], an INteractive Crisis Assistant applied to hazardous materials domain. The retrieval step is not crucial in this domain, and the features don't partition the case memory into meaningful groups. Moreover, the accuracy level is not a strong requirement because providing a reasonably good initial solution rather than the best initial solution is sufficient to allow the user to perform an additional adaptation phase. The interactive adaptation in INCA shares with CHARADE the most typical operations: add an action, delete an action, shift the start time of an action, change the duration of an action, or replace one of the specific resources assigned to an action. But the adaptation is much less constrained than in CHARADE. In fact, INCA doesn't introduce temporal relations among actions.

8. Conclusions

This paper presents an interactive planning system embedded in a decision-support system devoted to emergency management. The hybrid architecture satisfies the requirement to interactively involve the user in the decision process. The planner relies on a case-based reasoning component to support initial candidate solutions drawn from previous cases. The collaboration between the user and the planner is supported by a temporal constraint reasoner. While the user retains control of the adaptation process, the constraint reasoner verifies the consistency of such a process.

From the retrieval point of view, this paper describes a novel approach to computing nearest neighbor based on a local asymmetric metric. In this way, it is possible to reduce the size of case memory while maintaining the same accuracy and increasing the time response. Moreover, this method is robust with respect to the selection of the prototypes used in the retrieval phase. A reinforcement learning procedure is also provided for adapting the weights of a local similarity metric to the training data. Among the advantages of this new kind of metric is that it doesn't require a problem-specific setting of parameters.

Plan reuse and plan revision were supplied with efficient temporal reasoning procedures that support the incremental updating of the plan. In particular, interactive editing functions, such as adding or deleting actions, shifting action start time, or modifying action duration, can be efficiently performed. The planner plays the role of consistency checker of such operations. The interactive management and adaptation is done with a novel algorithm that performs fast incremental constraint propagation.

CHARADE is currently used in the south of France at CIFSC, a firemen school. Here, CHARADE supports training by a role playing methodology.

Acknowledgments

This work has been partially supported by the EspritIII project#6095 CHARADE (Combining Human Assessment and Reasoning Aids for Decision Making in environmental Emergencies). The CHARADE project is managed by Alenia. The other partners are Alcatel (Software Platform), Thomson CSF/SDC (Man Machine Interface), Inisel (Situation Assessment),
Italsoft (Resource Allocator), Thomson CSF/LER (cartographic module).

Notes

1. In the context of training, where the system is currently used, the global evaluation of an intervention plan, as designed by the trainee according to the plan representation above, is done by the expert.
2. 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, y are directly available from the label of the edge that connects x, y and, considering the rest of the network, does not produce any further reduction.
3. $E' = \{(x, y) \in V \times V \mid (x, y) \in E\}$ and the weight of an edge $(x, y)$ in $G'$ is the same as the weight of $(x, y)$ in $G$.
4. $\sim$ is an equivalence relation and if two vertices are equivalent, then they have the same s.p. distance from the source vertex of the graph. The graph $G^\ast = (V', E')$ is called the canonical metagraph of $G = (V, E)$. If $G$ contains only loops whose length is zero, then the canonical metagraph is also known as the component graph. The vertices of the component graph are the strongly connected components of $G$ [10].

References


**Anna Perini** received her doctoral degree in physics from the University of Trento in 1981. She worked in the area of statistical mechanics and computer simulation applied to solid state physics problems, with a fellowship from the Istituto per la Ricerca Scientifica e Tecnologica, Trento (Ist-IRST), Italy. Since 1985 she has been a Research Scientist in the Artificial Intelligence Department at IRST, working on the application of AI techniques to environmental decision support systems devoted to agricultural and to forest fire management problems. Her research interests include constraint satisfaction problems and temporal reasoning, case-based reasoning and planning.

**Francesco Ricci** was born in Venice, Italy, in 1959. He received the doctoral degree in Mathematics from the University of Padova, in 1983. From 1984 to 1986, he was Ph.D. student in Mathematics at Scuola Normale Superiore, Pisa, Italy. From 1986 to 1987, he was with Enichem, S.P.A., Milan, Italy, where he engaged in developing expert systems and office automation applications. Since 1988 he has been a researcher at Istituto per la Ricerca Scientifica e Tecnologica, Trento. His current research interests include constraint satisfaction problems, machine learning and case-based reasoning. Dr. Ricci has been on the programme committee of international conferences and is co-chair of EWCBR-2000 in Trento. Dr. Ricci received the best degree award from University of Padova in 1984.