CHAPTER 22

Emergency Evacuation Planning

22.1 Introduction

Generative planners traditionally require a complete description of the planning domain. However, in practical planning applications, developing a complete description is not always feasible.

One example is the task of planning how to evacuate groups of people who may be in danger. In general, there will be an incomplete domain description, in the form of standard requirements and operating procedures. However, these cannot be used to derive detailed plans, which often require knowledge about previous experiences.

Formulating an evacuation plan can be quite complex: typically there will be hundreds of tasks to be carried out. These tasks will depend on a wide range of factors: sources of danger, available resources, geography, weather predictions, political issues, and so forth. Complete information about the current state will never be available; the planning must include dynamic information gathering, and plans must be formulated with an incomplete world state.

For such a problem, the planning must be done by a human expert or under the supervision of a human expert. It is unrealistic to expect that a planning system could produce good plans by itself, and flawed evacuation plans could yield dire consequences.

This chapter describes a plan formulation tool, Hierarchical Interactive Case-Based Architecture for Planning (HICAP), that was designed to assist human experts in planning emergency evacuations. Because the plans are strongly hierarchical in nature, HICAP represents plans using HTNs.

As shown in Figure 22.1, HICAP integrates a task decomposition editor, Hierarchical Task Editor (HTE), with a mixed-initiative planning system, SHOP integrated with NaCoDAE (SiN). HTE allows users to edit tasks, and SiN allows users to interactively refine HTN plans. Their integration in HICAP ensures that operational plans are framed within the standard requirements and operating procedures or within the changes made by human planners through interactive

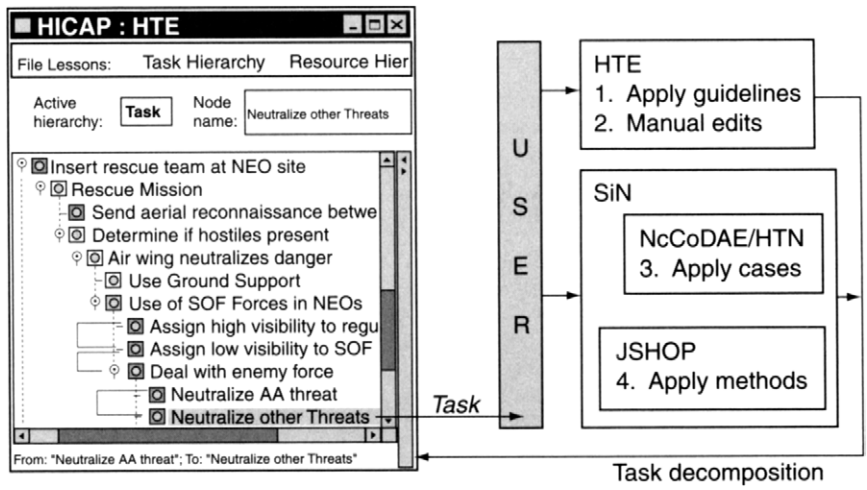


Figure 22.1 The HICAP plan-authoring system.

task editing and interactions with HICAP's case-based and generative planning modules.

Sections 22.4 and 22.5 describe HTE and SiN, and Section 22.6 gives an example of HICAP in operation. Section 22.7 gives a summary, and Section 22.8 discusses related work.

22.2 Evacuation Operations

The number of people involved in carrying out an evacuation operation can range into the hundreds, and they can be geographically distributed and often from several different countries. Depending on the situation, the number of evacuees can number into the thousands.

The official in charge of planning an evacuation operation will do so in the context of a set of standard procedures describing general aspects that must be considered. Figure 22.2 gives an example of the top-level tasks that may need to be performed. ISB denotes the *intermediate staging base*, the location where the evacuation team will be based prior to the evacuation. Arrows between tasks denote their execution order.

The standard procedures are limited: they are idealized and cannot account for characteristics of specific evacuation operations. Thus, whoever is in charge of planning the operation must always adapt these procedures to the needs of the specific operation by eliminating irrelevant planning tasks and adding others. This adaptation process depends partly on the operation's needs and resource availabilities. For example, the standard procedures may state that a small initial team should go into

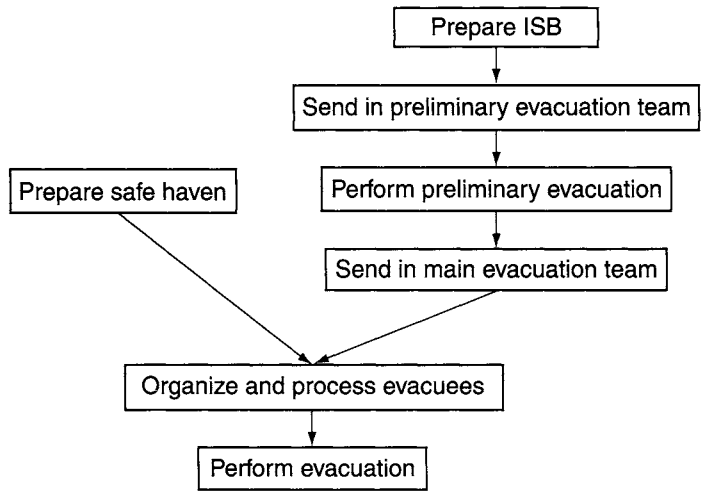


Figure 22.2 Top-level tasks.

the evacuation area prior to the main evacuation team, but in a specific evacuation operation, the time constraints may prevent this. It also depends on the planner's past experience, which may complement the standard procedures by suggesting refinements suitable for the current environment. For example, the planner could draw upon his or her own experience or the experience of others to identify whether it is appropriate to concentrate the evacuees in a single location or to plan multiple evacuation sites.

The following sections describe how HICAP can assist human planners by interactively developing evacuation plans.

22.3 Knowledge Representation

HICAP uses task networks similar to the ones discussed in Chapter 11. It also uses *cases*, portions of plans that were formulated during previous planning episodes (see Section 24.1). Both of these are described in the following subsections.

22.3.1 HTNs

In HICAP, an HTN is a set of tasks and their ordering relations, denoted as $N = (\{t_1, \ldots, t_m\}, \prec)$, where $m \geq 0$ and $\prec$ is a binary relation expressing temporal constraints between tasks. Decomposable tasks are called *compound*, while nondecomposable tasks are called *primitive*.

A domain description consists of methods and operators for generating plans. A method is an expression of the form $M = (h, P, ST)$, where h (the method's head) is a compound task, P is a set of preconditions, and ST is the set of M's subtasks. M is applicable to a task t, relative to a state S (a set of ground atoms), iff *matches*(h, t, S) holds (i.e., h and t have the same predicate and arity, and a consistent set of bindings B exists that maps variables to values such that all terms in h match their corresponding ground terms in t) and the preconditions P are satisfied in S.

An operator is an expression of the form $O = (h, aL, dL)$, where h (the operator's head) is a primitive task, and aL and dL are the add and delete lists, respectively. These are equivalent to the positive and negative effects defined in Chapter 2: every element in the add list is added to S, and every element in the delete list is removed from S. An operator O is applicable to a task t, relative to a state S, iff *matches*(h, t, S).

A planning problem is a triple (T, S, D), where T is a set of tasks, S is a state, and D is a domain description. A plan is the collection of primitive tasks obtained by decomposing all compound tasks in a planning problem (T, S, D).

22.3.2 Cases

In many domains it is impossible to assume that a complete domain description of the world is known. A partial domain description may exist, in the form of standard requirements and operating procedures—and these can be encoded into methods and operators.

For those parts of the domain for which no domain description is available, reasoning is done through cases. In HICAP, a case is a task decomposition that was created by the user while solving a previous planning problem. A case looks similar to an instance of a method, but usually it is not an instance of any method in the domain description.

Syntactically, a case is denoted by $C = (h, P, ST, Q)$, where h, P, and ST are defined as for methods and Q is a set of (*question, answer*) pairs. Q defines preferences for matching a case to the current state. Preferences are useful for ranking cases in the context of incomplete world states and/or domain theories because they focus users on providing relevant additional state information.

22.4 Hierarchical Task Editor

Because evacuation operations can be complex, it can be difficult to keep track of the completion status for each task to be performed and each element of the evacuation team. HTE was conceived to facilitate the planning process. A portion of the user interface to HTE is shown in the left-hand part of Figure 22.1. Given a domain-specific knowledge base for tactical planning, HTE can be used to browse and edit the knowledge base's components, select tasks for further decomposition,

and investigate the status of tasks. HTE serves HICAP as a bookkeeping tool; it maintains the task agenda and helps planners formulate plans for decomposable tasks.

HTE's knowledge base consists of an HTN, a command hierarchy, and an assignment of tasks to commands. For applying it to plan formulation, an HTN (i.e., a task network decomposition hierarchy) was developed that captured critical planning knowledge corresponding to some standard requirements. This required a substantial manual knowledge acquisition effort; the HTN consists of more than 200 tasks and their ordering relations. The HTN also includes a tree structure to represent the command hierarchy among the team that will carry out the operation. In addition, HTE's knowledge base includes relations between tasks and the team elements responsible for them. This is represented by an assignment function from the team elements to the tasks because the mapping of tasks to team elements is many-to-one.

In addition to providing users with a visual description of the standard requirements and procedures, HTE can be used to edit the HTN, its ordering relations, the command hierarchy, and the mapping between tasks and command assignments. Thus, users can use HTE to tailor its knowledge base according to the particular circumstances of the current operation. Furthermore, users can modify the command hierarchy as needed to represent the resources available for the current planning scenario. Finally, they can reassign tasks and/or team elements.

22.5 SiN

HICAP incorporates a mixed-initiative planner, SiN. SiN is a synthesis of JSHOP, a generative planner, with NaCoDAE, a conversational case retriever [94]. SiN is a provably correct algorithm that does not require a complete domain description nor complete information about initial or intermediate world states.

Users can interact with HTE by selecting a task T to be decomposed. This invokes SiN to start decomposing the task under the supervision of the user. This decomposition can be recursive; subtasks of N can themselves be decomposed further. Eventually, nondecomposable tasks corresponding to operational actions will be reached. Task decompositions are immediately displayed by HTE.

The following subsections describe SiN, including theoretical results on its correctness with respect to incomplete domain descriptions, and describe an empirical analysis that demonstrates the impact of the preferences on plan quality.

22.5.1 How SiN Works

As mentioned, the SiN planning algorithm integrates the task decomposition algorithms of two planning systems: the JSHOP generative planner and the NaCoDAE case-based planner. A single (current) state S is maintained in SiN that is accessible

to and updateable by both JSHOP and NaCoDAE. Answers given by the user during an interaction with NaCoDAE are added to S (i.e., each question has a translation into a ground atom). Changes to the state that occur by applying JSHOP's operators are also reflected in S.

JSHOP is a Java implementation of SHOP [414], a planning algorithm similar to the TFD procedure described in Chapter 11.[1]

NaCoDAE is a mixed-initiative case retriever. Users interact with NaCoDAE in *conversations*, which begin when the user selects a task t. NaCoDAE responds by displaying the top-ranked cases whose preconditions are satisfied and whose heads match t. Cases are ranked according to their similarity to the current state S, which is the state that exists at that time during the conversation. Similarity is computed for each case C by comparing the contents of S with Q, C's (q, a) preference pairs. (That is, each pair is represented as a monadic atom in S, and similarity for a given (q, a) preference pair becomes a membership test in S.) NaCoDAE also displays questions, whose answers are not known in S, ranked according to their frequency among the top-ranked cases. The user can select and answer (with a) any displayed question q, which inserts (q, a) into S. This state change subsequently modifies the case and question rankings. A conversation ends when the user selects a case C, at which time the task t is decomposed into ST (i.e., C's subtasks).

SiN receives as input a set of tasks T, a state S, and a knowledge base $I \cup B$ consisting of an incomplete domain description I and a collection of cases B. The output is a solution plan π consisting of a sequence of operators in I. Both JSHOP and NaCoDAE assist SiN with refining T into a plan. As does JSHOP, SiN maintains the set of tasks in T' that have not been decomposed and the partial solution plan π. At any point in time, either JSHOP or NaCoDAE is in control and is focusing on a compound task $t \in T'$ to decompose. SiN proceeds as follows.

- Rule 1: If JSHOP is in control and can decompose t, it does so and retains control. If JSHOP cannot decompose t but NaCoDAE has cases for decomposing t, then JSHOP will cede control to NaCoDAE.
- Rule 2: If NaCoDAE is in control, it has cases for decomposing t whose preconditions are satisfied. If the user applies one of them to decompose t, then NaCoDAE retains control. If NaCoDAE has no cases to decompose t or if the user decides not to apply any applicable case, then if t can be decomposed by JSHOP, NaCoDAE will cede control to JSHOP.

If neither of these rules applies, then SiN backtracks, if possible. If backtracking is impossible (e.g., because t is a task in T), this planning process is interrupted and a failure is returned.

By continuing in this way, assuming that the process is not interrupted with a failure, SiN will eventually yield a plan π.

---

1. JSHOP is available for downloading at http://www.cs.umd.edu/projects/shop.

22.5.2 Correctness of SiN

In this section we will assume that SiN performs ordered task decomposition. That is, we assume that all tasks are totally ordered and at each iteration, when refining a set of tasks T', SiN will start by decomposing the first task in T'.

If I is an incomplete domain description and B is a case base (i.e., a set of cases), then a domain description D is consistent with $I \cup B$ iff (1) every method and operator in I is an instance of a method or operator in D and (2) for every case $C = (h, P, ST, Q)$ in B, there is a method $M = (h', P', ST')$ in D such that h, P, and ST are instances of h', P' and ST', respectively. Although many different domain theories might be consistent with $I \cup B$, in general we will not know which of these is the one that produced I and B. However, SiN is correct in the sense that, if it succeeds in outputting a plan, then that plan could have been generated by JSHOP using any domain description consistent with $I \cup B$.

Proposition 22.1 *Let T be a collection of tasks, S be an initial state, I be an incomplete domain description, and B be a case base, and let SiN(T, S, I, B) represent the invocation of SiN with those items as inputs. Suppose that SiN performs ordered task decomposition. Then:*

- *If SiN(T, S, I, B) returns a plan π, then for every domain description D consistent with $I \cup B$, π is a solution plan for the planning problem (T, S, D).*
- *If SiN(T, S, I, B) cannot find a plan, then there is a domain description D consistent with $I \cup B$ such that no solution plan exists for (T, S, D).*

The proof is done by induction on the number of iterations of the SiN algorithm. The proof shows that each SiN task decomposition in $(T, S, I \cup B)$ corresponds to a JSHOP task decomposition in (T, S, D). This is sufficient to prove correctness because of the correctness of JSHOP's planning algorithm [414].

This proposition suggests that cases in SiN supply two kinds of knowledge. First, they provide control knowledge, similar to the knowledge encoded in cases using derivational replay when a complete domain description is available [275, 523]. Because cases are instances of methods, applying a case is comparable to a replay step in which the method selected to decompose a task is the one in the case's derivational trace. The main difference is that, while cases in replay systems correspond to a complete derivational trace, cases in SiN correspond to a single step in the derivational trace. Second, cases in SiN augment the domain description and thus provide domain knowledge as do cases in many case-based planners (e.g., [255]).

22.5.3 Imperfect World Information

SiN uses NaCoDAE to dynamically elicit the world state, which involves obtaining the user's preferences. Depending on the user's answers, cases will get reranked. When solving a task, the user can choose any of the cases, independent of their

ranking, provided that all their preconditions are met. The preferences play a pivotal role in determining plan quality due to the absence of a complete domain description.

Consider the following two simplified cases.

Case 1:

Head: selectTransport(ISB,evacuation-site)
Preconditions: HelosAvailable(ISB)
Question-Answer pairs: Weather conditions? Fine
Subtasks: Transport(ISB,evacuation-site,HELOS)

Case 2:

Head: selectTransport(ISB,evacuation-site)
Preconditions: groundTransportAvailable(ISB)
Question-Answer pairs:
  Weather conditions? Rainy
  Imminent danger to evacuees? No
Subtasks: Transport(ISB,evacuation-site,GroundTransport)

These cases both concern the selection of transportation means between an ISB and the site to be evacuated. The first case suggests using helicopters provided that they are available at the ISB. The second one suggests using ground transportation provided that the corresponding transportation means are available at the ISB. If the two cases are applicable because both preconditions are met, the answers given by the user will determine a preference between them. For example, if the weather is rainy and there is no immediate danger for the evacuees, NaCoDAE would suggest the second case. The rationale behind this is that flying in rainy conditions is risky. Thus, selecting ground transportation would be a better choice.

22.6 Example

During a typical planning episode, the user views the top-level tasks first, revising them or their assignments if necessary. He or she may choose to decompose any of the tasks and view their decomposition. Figure 22.3 shows an intermediate stage during this process. The user has selected the task "Select assembly areas evacuation & ECC [Evacuation Control Center] sites." Thus, the left-hand pane highlights this task. The right-hand pane shows the hierarchical organization of the evacuation team, so that the user can assign the task to a particular part of the team.

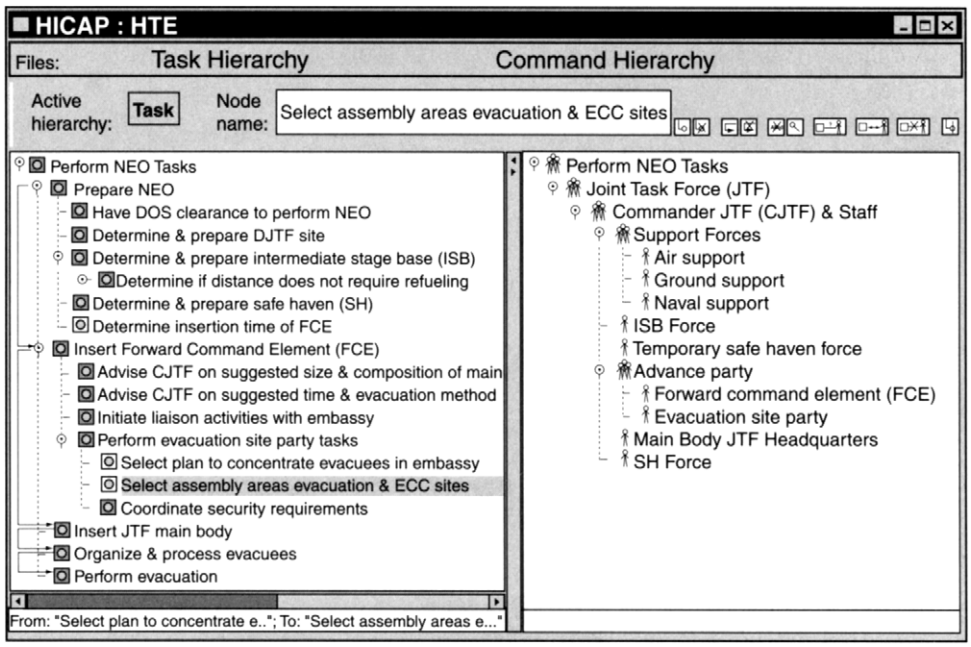


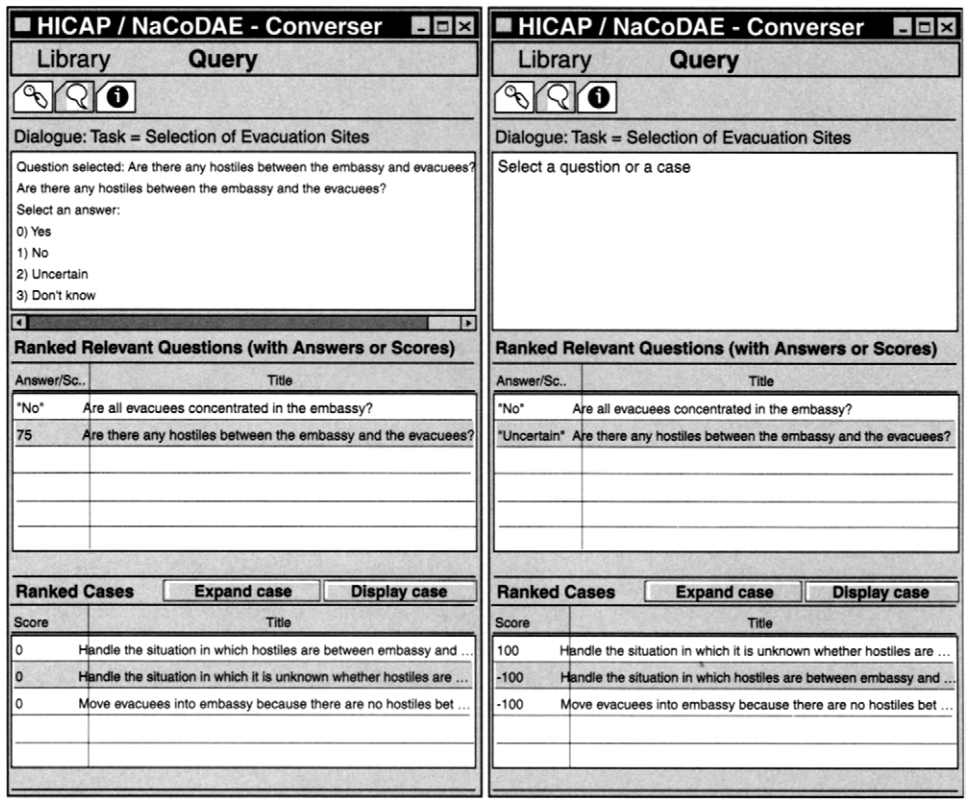
Figure 22.3 A snapshot of HTE's interface, displaying tasks (left) and the hierarchical organization of the evacuation team (right). Arrows denote ordering constraints.

Several alternative methods can be considered for decomposing the "Select assembly areas" task. When the planner selects this task, HICAP starts NaCoDAE, which displays the alternatives along with two questions to help distinguish which is the best match [see Figure 22.4 (a)].

In Figure 22.4 (b), the user has answered one of the questions, and this has yielded a perfect match to one of the cases for the "Select assembly areas" task. Suppose that the user selects this case to decompose this task. Figure 22.5 shows the result of this decomposition; two new subtasks are displayed that correspond to this case's decomposition network. Interaction can continue in a similar manner until all of the operational elements of the plan have been elaborated.

22.7 Summary

HICAP is an interactive planning tool that assists users in formulating an operational plan. As of the publication date of this book, HICAP had not been deployed but was still under development at the U.S. Naval Research Laboratory.



(a) Before a question has been answered

(b) After a question has been answered

Figure 22.4 Two snapshots of NaCoDAE/HTE's interface. In each, the top window displays advice on what to do next and, when the user selects a question, lists the possible answers. The lower windows display the questions and cases, respectively.

HICAP is interactive; it supports task editing and triggers conversations for tasks that can be decomposed in more than one way. The planning process consists of HTN task decomposition, some of which is done by the user using the HTE editor and some of which is done by HICAP using the SiN planning algorithm. The HTE plan editor allows the user to visually check that all tasks are assigned the necessary resources.

HICAP's SiN procedure is a provably correct procedure for combined case-based and generative planning with incomplete domain descriptions. It tightly integrates NaCoDAE/HTE's case-based task decomposition and JSHOP's generative planning ability. Experimental results with SiN show that a user can dynamically guide SiN

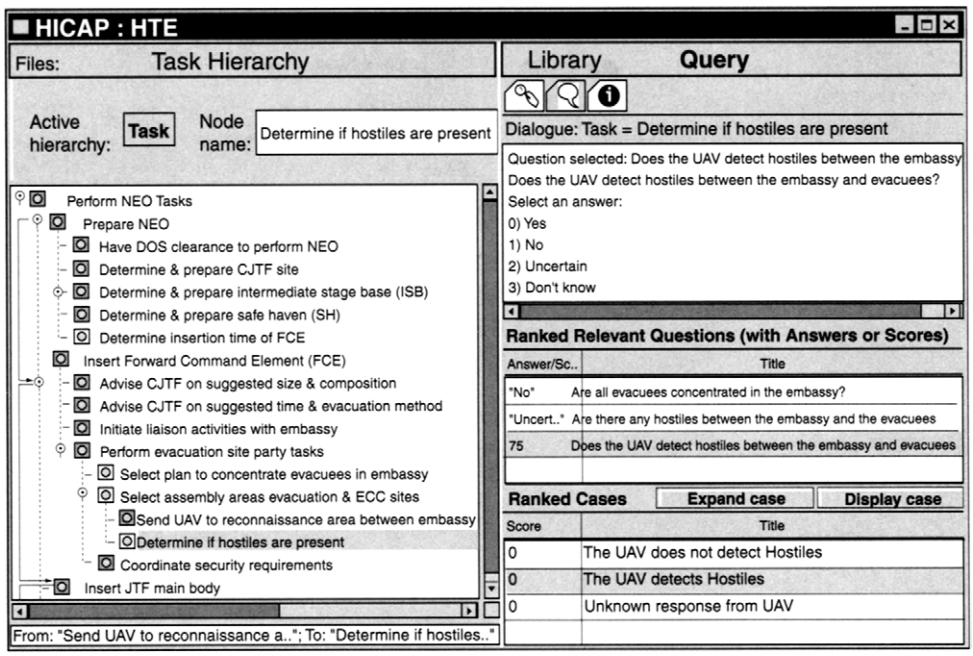


Figure 22.5 HICAP's interface after decomposing the "Select assembly areas" task.

by giving preferences to it as part of the user's normal interaction with SiN during the planning process.

SiN's ability to combine both experiential and generative knowledge sources can be beneficial in real-world domains where some processes are well known and others are obscure but recorded memories exist on how they were performed. Evacuation planning is an example of this type of domain.

22.8 Discussion and Historical Remarks

HICAP is being developed at the U.S. Naval Research Laboratory. Our descriptions of HICAP, SiN, and JSHOP are based on two sources [406, 414]. The following text summarizes other work related to HICAP.

CHEF [256] and DIAL [356] are case-based but do not have a generative component, and thus they need a large case base to perform well across a wide variety of problems. PRODIGY/Analogy [523], DerSNLP [275], and PARIS [61] integrate generative and case-based planning but require a complete domain theory and are not mixed-initiative.

At least three other integrated (case-based/generative), mixed-initiative planners exist. MI-CBP [526], which extends PRODIGY/Analogy, limits interaction to providing it with user feedback on completed plans. Thus, MI-CBP must input or learn through feedback a sufficiently complete domain description to solve problems. In contrast, SiN gathers information it requires from the user through NaCoDAE conversations but does not learn from user feedback. CAPlan/CbC [407] and Mitchell's system [397] use interaction for plan adaptation rather than to acquire state information.

Among integrated case-based/generative planners, SiN's interleaved control structure is unique in that it allows both subsystems to equally control the task decomposition process. In contrast, other approaches either use heuristics (PRODIGY/Analogy, MI-CBP) or order case-based prior to generative planning (DerSNLP and Mitchell's system), although PARIS does this iteratively through multiple abstraction levels. Distinguishing the relative advantages of these control strategies is an open research issue.

CaseAdvisor [107], like SiN, integrates conversational case retrieval with planning. While CaseAdvisor applies prestored hierarchical plans to gather information to solve diagnostic tasks, SiN instead uses its case retriever to gather information and applies cases to refine hierarchical plans.

Other researchers have described related systems for crisis response tasks. Tate *et al.* [505] describe the use of O-Plan as a mixed initiative multi-user planning and replanning aid generating multiple potential responses for a range of non-combatant evacuation and crisis response operations, Ferguson and Allen [186] describe an interactive planner for crisis response applications, and Wolverton and desJardins [557] describe a distributed planning tool, these systems are note case-based.

Gervasio *et al.* [221] describe an interactive hierarchical case-based scheduler for crisis response, but it does not perform interactive plan formulation. Avesani *et al.* [26] describe a case-based planning approach for fighting forest fires that supports interactive plan adaptation, but it does not use hierarchical guidelines to formulate plans as is done in HICAP. Finally, Leake *et al.* [357] describe a case-based planner applied to disaster response tasks that focuses on learning case adaptation knowledge, but it is not driven by standard requirements and operating procedures, and it focuses interactions on knowledge acquisition rather than problem elicitation.