# Heracles: Hierarchical Dynamic Constraint Networks for Interactive Planning

**José Luis Ambite, Craig A. Knoblock & Maria Muslea**
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{ambite,knoblock,mariam}@isi.edu

**Steven Minton**
Fetch Technologies
4676 Admiralty Way, Marina del Rey, CA 90292
minton@fetch.com

For any human activity there is a wealth of information available in the Internet and in corporate intranets. Unfortunately, such information is distributed among many sites, with different data formats, schemas, and semantics. Moreover, information access per se is of limited value. What is needed is a system that integrates and structures diverse information in support of the user tasks and goals. The system must focus on the relevant information, evaluate tradeoffs, and suggest courses of action to the user. In this demo, we we present Heracles, a constraint-based framework to easily develop such systems.

As an example consider travel planning. There are numerous sites with relevant travel information: flight schedules and fares (e.g., www.orbitz.com), hotel locations and rates (e.g., www.itn.com), car rental sites (e.g., www.hertz.com), weather information (e.g., weather.yahoo.com), maps and route planning (e.g., www.mapquest.com), airport parking rates (e.g., www.airwise.com), etc. This public information needs to be integrated with user preferences, such as preferred airlines or flying times (e.g. avoid red-eye flights), cost constraints, and company policies, such as allowable airlines, expenses caps, per-diem or mileage reimbursement rates. Although the user could visit these sites and take into accounts all the constraints and preferences, it is extremely tedious, error-prone, and time-consuming. A system that queries these the remote sites, access local information, and enforces the constraints and preferences is much more desirable.

Heracles models each piece of information as a variable in a constraint network. The different pieces of information are integrated using constraints. The resulting constraint network provides a coherent view of the user activities and captures the relevant information and user preferences.

For any non-trivial user activity the number of variables and constraints is very large. So we do not propose a monolythic constraint network, but we partition the network hierarchically. This hierarchy corresponds to the task structure of the application domain, in a manner similar to Hierarchical Task Network planning (Erol, Hendler, & Nau 1994). The application designer groups variables and constraint related to a distinct task into a package that we call a *template*. For example, in travel planning we would have a top-level template that contains the most important information, such as who is traveling, the dates of travel, and the origin and destination. The next layer of decisions include the alternative means of transportation, such as flying, taking a train, renting a car, driving the user's own car, or taking a taxi; and choices of accommodation at the destination. The variables and constraint related to flying would constitute a template. Task/Templates are further decomposed into smaller units. For example, once that the user decides to fly, the system suggests how to get to the airport: by taxi, driving one's car and leaving it parked at the airport, etc, so there are subtemplates for each of these alternatives.

Figure 1 shows the Heracles user interface with top-level template of the travel planning application. There is a set of boxes showing values, which we call *slots*, which show the value of a variable in the constraint network. For example, the street, city, and state of the departing address is `2700 University Park, Los Angeles, CA`. The hierarchical task structure of the domain appears on the left pane in the user interface. The system has suggested to fly and taking a taxi from and to the airports. The fly and taxi subtemplates are also shown.

This hierarchical decomposition allows to manage the complexity of the application in three ways: (1) it enables a more efficient evaluation of the constraint network, (2) it gives the user a clear understanding of the domain tasks, and (3) it helps to manage the complexity of application development.

Heracles is implemented as a dynamic constraint network (Mittal & Falkenhainer 1990). From the planning perspective, a template is equivalent to a task. Once the system selects a given course of action, only one of the alternative templates should be active. That is, the system only needs to evaluate constraints from that template. The active constraint network corresponds to the currently selected plan, which is comprised of the set of selected templates/tasks. Our use of dynamic constraints networks is analogous to their use in configuration. In a sense, Heracles "configures" a plan that satisfies the user goals.

In many domains, such as travel planning, evaluating a constraint can be expensive. The focused evaluation imposed by the task structure of the dynamic constraint network yields significant savings. For example, in travel planning the system may evaluate a constraint that retrieves maps and directions of travel from a web site such as mapquest.com. Such retrieval takes on the order of seconds,

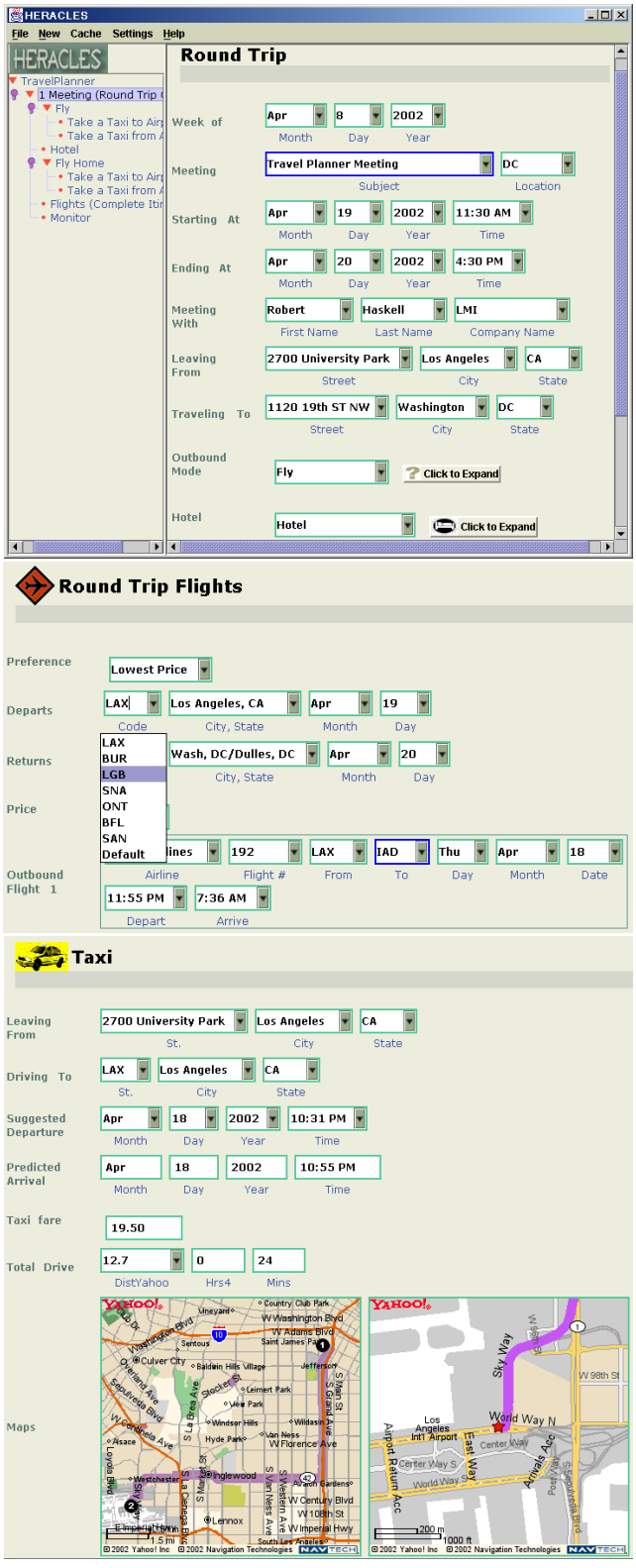


Figure 1: Travel planning: top-level, Fly, and Taxi templates

so the number of evaluations of the constraint must be minimized. Such request will not be evaluated unless the driving template is active.

Since we cannot assure that all information, constraints, and preferences are ever captured by the system, the planning process should be conducted in a mixed-initiative fashion where the user can explore different alternatives and override the system suggestions if needed. Allowing the user such control over the system presents significant challenges. First and foremost, the system must behave in a way comprehensible to the user. The usability of the system would be seriously diminished if when the user selected a new value, suddenly all values shown in the interface changed. The user would get lost. For this reason, Heracles does not perform full constraint satisfaction, but only *directed constraint propagation*. When the user inputs a value, downstream variables acquire consistent values if possible, but they do not affect the values of variables upstream in the network. This is analogous to doing a beam search of the constraint network, where the user guides the exploration of certain paths in the space of solutions. If a dead end is reached the user is informed by showing no values for downstream variables in the interface. The user can then backtrack on a previous choice, but always maintains control and understanding of the system behavior. Second, since Heracles allows the user to input values for some of the variables in the constraint network and override the system suggestions, some constraints may not be satisfied (precisely those that compute the suggested value). Heracles allows such local inconsistencies when they cause no confusion in task planning. For example, the system may suggest to drive one's car to the airport because it's cheaper, but the user can always choose to take a taxi. Heracles will disregard the constraint that selects driving and propagates values such a taxi cost from the selected task/template. Finally, since the directed constraint network can be cyclic, the system ensures (1) that the user interaction does not produce infinite cycles in the constraint propagation, (2) that the latest user inputs are fully propagated, and (3) the results from obsolete constraint evaluations are disregarded (since calls to external sources return the result asynchronously).

In the demo session we will demonstrate mixed-initiative planning in Heracles. We will show the declarative specification of the template hierarchy, including constraints that access local sources as well as local computation.

## References


Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, 1123–1128. Seattle, Washington, USA: AAAI Press/MIT Press.

Mittal, S., and Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 25–32.