From: AAAI Technical Report SS-98-04. Compilation copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Synthesizing Solutions from Memory Chunks

S. Shankara Raman and **Deepak Khemani**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai 600 036 India
khemani@iitm.ernet.in

Abstract

An intelligent agent or problem solver is expected to solve a problem correctly and efficiently, communicate the solution and also explain the reasoning behind the solution. Several problem solving paradigms are available, each having its own advantages. Model based planners are very robust but at increased computational cost while case based methods are often more efficient. In this paper we explore a problem solving methodology integrating these two methods in a hybrid architecture, exploiting their advantages and avoiding the drawbacks. Route planning has been chosen as the candidate problem. It has two major subactivities, one in route finding and the other in route communication. Both these problems have been earlier studied from the point of view of path planning and natural language generation. We describe a system, named CaSyn (Case Synthesizer), which adopts a hierarchical problem solving approach with the higher level reasoning providing control and a means of explaining the solution.

Introduction

In this paper we, explore a problem solving methodology where the agent uses a memory based approach to find solutions but can fall back upon first principles when the memory does not contain a solution. We describe a system that synthesizes solutions from partial solutions retrieved from a case base. The synthesizer is essentially a weak problem solver that treats the retrieved parts as high level operators. CaSyn operates in a two level hierarchy, with problem solving at the higher level controlling the flow between the two modes at the lower level.

CaSyn plans routes in a city, which is organized as a network of roads and nodes at the lower level; as a graph of adjacent zones at a higher level; and a database of names and locations for communicating solutions and explanations. The user request is a source destination pair, both of which are nodes in the city network, and the output is a set of roads and intermediate nodes which lead from the source to the destination.

Related work

Route planning is a classical problem which was earlier studied from the point of view of a robot navigating in space. (Goel & Chandrasekaran 1991) describe their work spanning four generations of *Routers*. Each successive generation router was an improvement over its predecessor. The system *Router4* (Goel *et al.* 1992) has a hybrid architecture integrating a case based approach with the traditional model based approach. This system combines multiple cases without partial synthesis.

The work reported in (Liu *et al.* 1994) describes a prototype system integrating Dijkstra's algorithm with knowledge based and case based components. The case based component tries to obtain a complete or partial solution from any single case. Dijkstra's algorithm is guided by the knowledge based component.

The problem of route planning was also studied from the point of view of describing routes to a user. (Maab, Wazinski, & Herzog 1993) describe VITRA GUIDE which provides multimodal route descriptions, fully specified or incremental, depending upon the situation specific constraints. A plan based approach achieves the coordinated use of the different presentation modes.

The next section describes CaSyn, in which the results obtained from both these points of view are adapted.

CaSyn

CaSyn is based on an earlier system, Sightseer (Khemani *et al.* 1995), which planned routes for different modes of travel like public transport or private vehicle. Sightseer used a search based approach for planning routes using public transport. Road route planning used mainly a search approach combined with a simple case based module. The plan presentation was both

textual and graphical. The present system focuses primarily on route planning for private vehicles. The primary problem solver is the case based module. The major emphasis is on synthesizing solutions by combining partial solutions which are extracted from cases retrieved from the memory. The model based planner, the secondary module, is retained from Sightseer and is used mainly for case adaptation and when the case based planner fails to deliver a solution. The topographical knowledge of the city is used to plan routes.

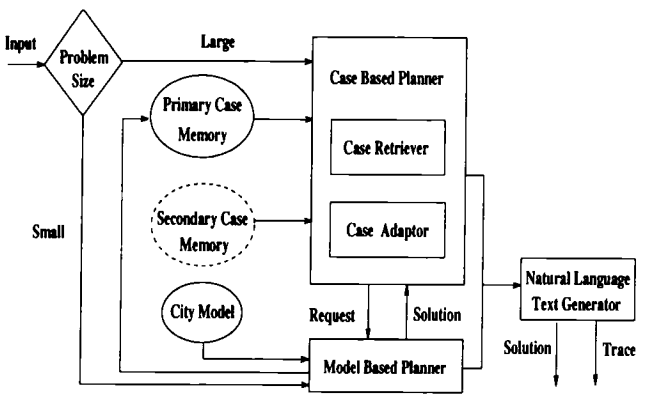

Figure 1: Architecture of CaSyn

Figure 1 illustrates the system architecture showing the various modules. The input module is a simple one, input being currently done through menus. The output module uses a simple text planning strategy to generate natural language sentences (Somayajulu & Khemani 1996). The system operates with a two level hierarchy; the processing decisions taken at the higher level being used to generate an explanation of how the solution is being arrived at. The system currently plans route in South Madras. The implementation is carried out in C under the UNIX environment.

Architecture

Model based planners work from first principles and can solve new problems. They generally yield the best solutions, but are computationally expensive. Case based planners, on the other hand, are more efficient at solving problems which are similar to ones previously encountered. However, when a good solution does not already exist in the memory, a case based system may perform poorly or even fail. In order to reap the benefits of both methods, we adopt a hybrid approach. The case based planner generates the solution at lesser computational costs and the model based planner provides an analytical solution when solution to no similar problems are available in the memory. Thus the guarantee of generating a solution is retained.

The lower level problem solver is a model based one. The results of problem solving achieved by this module are stored in the case memory. The model based problem solver models the city as a network of roads. The city map is divided into different zones and the roads are represented separately for each zone in which they occur. At the lower level, intersecting roads are broken into segments with nodes at the end points. Every node is associated with a unique number, name and a set of ¡x,y¿ co-ordinates denoting its location in the city map. Every road has a unique name and number; all road segments carrying the same name and number along with a cost which is the actual length of the segment. At the upper level, each zone has a number, name and information regarding the road segments present in the zone. Neighbouring zones are permitted some overlap with entry points providing connection between the zones. Each zone is also represented in a space hierarchy to facilitate upper level search. This is the two level explicit domain representation.

The A^* (Nilsson 1982) algorithm is used as the search algorithm. A heuristic evaluation function f is defined as f(n)= g(n) + h(n) at node n, where g(n) is the estimate of the minimal cost path from the start node to node n and h(n) is the estimate of the minimal cots path from node n to a goal node. Thus f(n) is an estimate of a minimal cost path constrained to go through node n. An obvious choice for function g is the sum of the costs of the individual road segments connecting the start node to node n. A good heuristic for function h is the Euclidean distance between node n and the goal node.

At the top level, the search charts out the zones to be traversed for the required route using the source and destination nodes along with the all the zone and entry point information. The actual route planning is done by the lower level search finding the detailed route within the zones marked out by the upper level search. This strategy hierarchically narrows down the search space, but it is possible that a "nice" solution is missed out, like a long curving highway from an adjacent zone.

The case base planner is the primary problem solver. The organization of the case memory for efficient retrieval is an important issue. We adopt a simple method wherein entire solutions are stored as cases. This memory is hierarchical, based on the city network hierarchy described earlier. The upper level contains the source, destination (both node and zone information) and the zones that the solution traverses. The lower level details contain the actual route, containing a set of road segments (defined between nodes) which lead from the source to the destination. The choice of an index is another important issue, it must be easily

computable and be able to differentiate between a set of cases. The source and destination details (¡x,y,¿ coordinates and zone numbers) are effective indices. A serial search procedure is considered as the retrieval algorithm in order to keep it simple.

The case based planner first tries to solve the problem. If an old case matches completely the problem is solved. When there is no exact match, the case based planner looks for a case which is similar to the required solution. The source and destination zones are use as indices for case retrieval. The solution may also be embedded in a larger case. In this scenario, the source and destination zones are contained in the zones stored in the upper level case and serves as an index for case retrieval. The adaptation of this retrieved case is treated as a fresh problem which is assumed to be small enough to warrant a search based solution. Every node in the case beginning from the source node is examined to determine which is nearest the required source. A similar procedure is adopted at the destination end of the case also. The portion between the nodes so determined is extracted out. The missing parts of the solution are then computed by the model based planner. This strategy assumes that the solutions are linearly decomposable.

A key feature of the system is that it combines partial solutions retrieved from the memory to synthesize the complete solution. Such a necessity arises when the solution to the current problem is distributed across two or more cases in the memory. Case retrieval and adaptation are slightly different and the following paragraphs discuss the important issues that arise.

In the scenario wherein one is looking for useful parts to extract from a solution, all cases are potentially useful. No good indices are readily available and a higher level reasoning is employed. The model based planner generates the set of zones within which the solution is likely to be contained. This information is used as the index and all the cases which lie along these zones are alone retrieved from memory.

These matching cases are first sorted on the size of the contributing segment. A Means End Analysis procedure then picks the most useful cases and passes them on to the adapter. The useful portions of these cases are first extracted by search. This is achieved by identifying the nodes in each case which are nearest to the required source and destination nodes respectively. The nearness criteria is based on the distance, direction and the connecting roads between the node under consideration and the source/destination node. This search procedure is further explained in the example illustrated by Figure 2. The extracted solutions are then arranged in the correct order leading from source to destination. The model based planner fills in the missing portions to complete the solution. In this way, the partial solutions are effectively treated as (higher level) operators by the model based planner. In case of failure by the case base planner to generate even partial solutions, the entire problem is solved by the model based planner.

The example given below describes the working of the system. In figure 2, (S_R,D_R) is the user query. The source and destination zones are 1 and 10 respectively and the case memory contains no case which is directly useful. The model based planner generates the upper level route as zones 1, 8, 9 and 10 in that order. Case 1 (S_1,D_1) which lies along zones 1 & 8 and case 2 (S_2,D_2) along 9 & 10 are two useful cases. Both cases 1 & 2 have equal match length of two zones. A search along case 1 starting from S_1 identifies S_R as a node nearest the required source node (S_R itself) and S_M as the node nearest D_R. The portion from S_R to S_M is extracted from case 1. A similar search identifies case 2 as being fully useful. The portion (S_M,D_M) is missing and is computed by the model based planner.

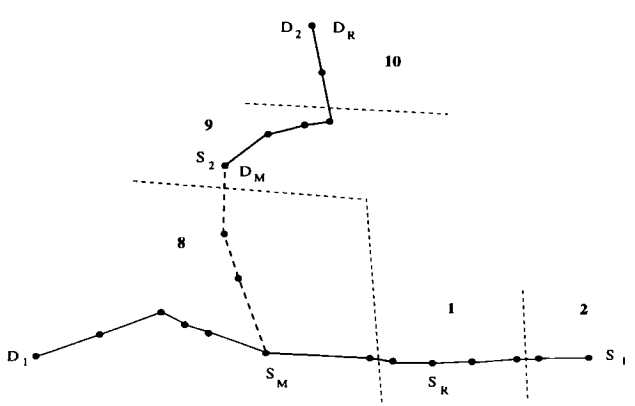

Figure 2: An Illustrative example

Plan Output

Source IIT Gate: From here go straight on sardar patel road(1) for a distance of about 0.43 Km. You can see the location 120. Take a turn there to your right. Go straight in that direction for a distance of about 0.94 Km. There you can find the location 121. Go along the road 81 for a distance of about 0.45 Km. You can see the location 122. Take a turn to the road 82. Take the road 82 and go straight for a distance of about 0.28 Km. There you can find saidapet bus stn. Go by Mount road(93) for a distance of about 1.37 Km. You can see saidapet veterinary hospital. Take a turn there to your left. Go straight in that direction for a distance of about 0.85 Km. There you can find t.nagar bus stn.

The plan output shown above is generated by the text generator module puts the set of road segments into simple sentences. The source node S_R is IIT Gate. The destination node D_R is T.Nagar Bus Station. Location 120 is S_M and D_M is Saidapet bus station. The output describes only the route, no information about the problem solving mode adopted is revealed.

Program Trace

You want to go from the IIT Gate to the T.Nagar Bus Station. I have not made a similar journey. You know the route passing from IIT Gate to Guindy Railway Station. You know the route from Saidapet Bus Station to T.Nagar Bus Station.

The program trace show above gives an explanation to the solution generated. The first sentence gives the problem definition, i.e., S_R and D_R. The second sentence articulates the fact that there is no single case in the memory which fits the current requirements. The last two sentences convey the fact that two cases exist in the memory, parts of which are useful.

Conclusions

In this paper we describe a system that demonstrates that a memory based problem solver is strengthened considerably when it has recourse to fall back on solving problems using first principles. Hierarchical problem solving provides control between the case based and model based modules. The problem solving at a higher level is used to generate an explanation of the problem solving process. We believe that hierarchical problem solving may be crucial to self explanatory systems.

References

Goel, A., and Chandrasekaran, B. 1991. Representation, organization, and use of topographic models of physical spaces for route planning. In *Proc. of the Seventh Conference on AI Applications.*

Goel, A.; Donnolan, M.; Vasquez, N.; and Todd, C. 1992. An integrated experience-based approach to navigational path planning for autonomous mobile robots. In *Proc. of the AAAI Fall Symposium on Applications of AI to Real World Autonomous Robotics.*

Khemani, D.; Somayajulu, S.; Gopal, S.; and Sitaraman, K. 1995. Sight seeing plans in natural language. In *Proc. of the IEEE International Conference on Systems, Man and Cybernatics.*

Liu, B.; Choo, S.-H.; Lok, S.-L.; Leong, S.-M.; Lee, S.-C.; Poon, F.-P.; and Tan, H.-H. 1994. Finding the shortest route using cases, knowledge and dijkstra's algorithm. *IEEE Expert* 7–11.

Maab, W.; Wazinski, P.; and Herzog, G. 1993. Vitra guide: Multimodal route descriptions for computer assisted vehicle navigation. In *Proc of the Sixth International Conference on Industrial and Engineering Applications on AI and Expert Systems.*

Nilsson, N. J. 1982. *Principles of Artificial Intelligence.* Springer-Verlag.

Somayajulu, S., and Khemani, D. 1996. Planning for task instruction. In *Proc. of International Conference on Knowledge Based Computing Systems, KBCS-96,* 291–302.