Paper accepted at Fifth International Conference on Industrial Automation, June 11-13, 2007, Montréal, Canada

A PROPOSED AI-PLANNING APPROACH FOR STAFF MANAGEMENT IN A SERVICE CENTER

Minh-Quang Nguyen[1], Philip H.P. Nguyen[2], Xu Fang Zhao[3], Tho-Hau Nguyen[4], Jean-Guy Meunier[5], Douglas O'Shaughnessy[6]

[1,3,6]*Institut national de recherche scientifique, EMT*
800, De la Gauchetière Ouest, #6900, Montréal, H5A 1K6, Canada. E-mail:{ nguyenmq, zhaoxf, dougo}@emt.inrs.ca

[2]*Justice Technology Services, Department of Justice, Government of South Australia,*
30, Wakefield St., Adelaide, SA 5000, Australia. E-mail: nguyen.philip@saugov.sa.gov.au

[4]*Université de Québec à Montréal, Dép. d'informatique*
201, Avenue du Président-Kennedy, PK-4150, Montréal, H2X 3Y7, Canada. E-mail: nguyen.tho-hau@uqam.ca

[5]*Université de Québec à Montréal, Dép. de philosophie*
455, Boulevard René-Lévesque Est, W-5440, Montréal, H2L 4Y2, Canada. E-mail: meunier.jean-guy@uqam.ca

Abstract: This paper presents a new approach for task allocation in a Service Center, based on artificial intelligence (AI) planning techniques. Our objective is to build an expert system that automatically allocates tasks to technicians in a Service Center. We describe the process of building an AI planner, from the design phase up to the final implementation phase, through the example of an IT service Help Desk. We examine in particular the process of determining the most suitable technician to assign to a new service request. Our approach is based on first-order logic, situation calculus and partial-order planning, to demonstrate the use of AI planning to solve complex problems by starting with a simple high-level model, then progressively decomposing it into finer components, until the whole solution is computationally implementable. This approach enables faster determination of a solution while meeting all the goals of the Service Center management.

Keywords: artificial intelligence planning, expert systems, staff management application, algorithm, solving problem.

1. INTRODUCTION

The approach for problem solving through planning is usually based on one of its two main methodologies called state-space and plan-space. The former is conceptually simple but does not always work when the domain is complex and presents some uncertain knowledge [9]. The latter usually relies on Artificial Intelligence (AI) and is known to offer better solutions in those situations, as demonstrated by the autonomous controller in NASA's Deep Space One spacecraft [10] and the Blackbox system of Kautz et al. [5][10]. The Graphplan methods based on systematic or stochastic SAT (Satisfiability Solving Technology) for planning problems [10][11][12] and the algorithm of backtracking search for Quantified Boolean Formula (QBF) solvers [6] are also state-of-the-art techniques based on AI planning.
Within AI Planning, partial-order planning is a method that has demonstrated its superiority in a number of cases [1][2][7][9]. In this paper, we present an application of AI-based partial-order planning to assist a service center manage its workload and staff. As an example, we examine the operations of an IT Help Desk in its daily activities of providing technical support to its customers. Our application is inspired by a system currently in use at the University of Quebec at Montreal (UQAM). We focus on the process of managing user requests, in particular the assignment of a technician to a new request. We describe how to formulate problem solving knowledge into an AI plan and how to implement that plan. This paper is organized as follows. First, we present the main principles of AI planning (Section 2). Then, we describe the activities of a typical Help Desk, especially the user support tasks that are the focus of our paper (Section 3). Section 4 details the design of an AI planning system for Help Desk, in which situation calculus, first-order logic and the STRIPS (STanford Research Institute Problem Solver) knowledge representational language are used to formulate the plan. Section 5 shows how the system could be implemented through the JRules software package from ILOG Inc. [4].

2. AI PLANNING

AI Planning is a branch of AI that deals with the design, formulation, validation and maintenance of planning domain models. The aim of AI planning is to find a sequence of actions (or operators) that transform an initial state into a final state in which the goal is satisfied. It is generally recommended to adopt the following principles when building an AI planner [9][5]:

. Representation of facts in terms of states, goals and actions with first-order logic or theorem proving. This allows a direct connection between states and actions. A new action should only be added to the planner if it does not disturb other past or future actions. This approach is also in line with partial-order planning, in which not all possible actions are examined and only truly necessary actions are taken into account in the planner. Whenever possible, a problem should be divided into independent sub-problems that could be solved separately. In other words, a plan could be constituted by independent sub-plans. This method is called "divide-and-conquer" [9], which could be summarized through the following recursive first-order logic formula (expressed in Disjunctive Normal Form - DNF) $I(S_0)$ --> $G(a|sg(b,S) , S)$ Where $I(S_0)$ represents the initial state in situation S_0, and G represents the goal state in situation S. G can be satisfied either by result a or by a subgoal $sg(b,S)$, in which result b is the outcome of the subgoal in situation S.

3. TASK ALLOCATION IN A SERVICE CENTER

In this section, we will model the operation of a service center, such as a Help Desk in a company. A service center is a call center, to which users can submit any work-related problem or request for assistance. The solution for the Help Desk world described in this paper could be easily generalized to suit most other types of service center.
The University of Quebec at Montreal (UQAM) includes a Department of Computer Science and Telecommunications, called SITEL (Service de l'Informatique et des TELécommunications). One of other SITEL's responsibilities is to manage an IT (Information Technology) service Help Desk whose main goals are to assist its users (i.e., professors, students, employees and other staff of UQAM) in the use of their PCs (assistance management) and to repair problems associated with their computer hardware and software (anomaly management). In a typical case, a user calls the Help Desk to submit a service request. The request is entered into a database by a Help Desk staff. The Help Desk then analyzes, prioritizes and assigns the request to a technician for resolution. The main goal of the Help Desk is to satisfactorily complete all requests as quickly as possible, according to their priorities and other dynamic factors, such as technicians' availability and relative priorities between requests.
Service requests can be classified in two categories: requests for assistance (called "Help-requests") and requests for anomaly repair (called "Ano-request"). Preferably, the technician selected to work on the request should be available and have skills matching the type of the request. The key objective of management is to be able to immediately deal with the most urgent requests and to be able to eventually assign a technician to every request. So, in practice, dealing with priority also means dealing with the time duration to accomplish a task. The detailed rule for assignment of a technician to a new service request is as follows:

. A user calls the Help Desk to submit a request for service. A new request form is created in the request database with the appropriate type (Help-request or Ano-request) determined by the Help Desk staff, according to the description of the request by the user. If there is a technician available with skills matching the request type then he/she is selected for the job.

. If there is no technician available with the appropriate skills, but there is a technician available albeit with different skills, then he/she is selected.

. If there is no technician available at all, but there is a technician meeting three conditions: skilled in handling the type of the new request, currently being assigned to a lower priority request and having not started work on that lower priority request, then he/she is selected. He/she is then unassigned from his/her current job and re-assigned to the new request. The lower priority request goes back to the request database, waiting to be re-assigned.

. If there is no technician available at all and there is also no technician meeting all three conditions mentioned above, but there is a technician meeting the last two conditions, then he/she is selected. He/she is then unassigned from his/her current job and re-assigned to the new request. The lower priority request goes back to the request database, waiting to be re-assigned, but with its priority increased.

. If all technicians are assigned to requests of a higher priority (regardless of whether those requests have been started or not), then the new request stays in the request database, waiting to be assigned in the next iteration of this process.

. If there is no technician available at all, but there are some technicians assigned to requests of a lower or same priority and having started work on them, then the system calculates the remaining times for those technicians to complete their current jobs and compares them with the estimated maximum waiting time that the new request can permit. The system then selects the technicians who need more time to complete his/her current task than to take on the new request within its permitted maximum waiting time. If no such technician is found then the new request remains in the request waiting queue and will be re-processed at the next iteration of this process. The selected technician is then un-assigned from his/her current request and re-assigned to the new request. Once assigned, the technician is responsible for all aspects of resolution of the request, including liaison with the user. When the job is finished, the technician reports back to the request database by updating the request status to "completed" and providing details of his/her problem resolution. The technician then becomes available for assignment to another request. The above cycle continues until there are no more outstanding un-assigned service requests in the database.

We propose to formally translate the above process so that it could be computationally implemented. To assist with understanding, we will simplify the description of the solution by only taking into account the main actions.

4. AI PLANNER DESIGN

To formulate the Help Desk problem and solution, we use partial-order planning in order to avoid as much as possible unnecessary actions. The approach is to reason in a *top-down* manner, starting from a very simple plan, consisting of just an initial state, a final state and a goal, then recursively decomposing them into finer objects, until the whole plan is computationally programmable. Our initial planner thus simply looks as follows: The Help Desk world described in a simplistic way. The transition from Start towards Finish depends on a precondition: All the requests must be completed in the Finish state. The pseudo-code for the above in the STRIPS language is as follows:

```
Plan(STEPS:{S1:Op(ACTION:Start),
S2: Op(ACTION:Finish),
PRECOND: AllCompletedRequests()}
ORDERINGS: {S1 < S2}
BINDINGS:{}
LINKS:{} )
```

Note that in STRIPS, *steps* describes the above Start and Finish states and the precondition, *orderings* define the ordering constraints between states (in this case, "$S_1 < S_2$" means that State S_1 should always precede State S_2), *bindings* define the constraints between variables (there are no initial variable constraints in our case) and *links* define the causal linkages between states (there are no initial causal linkages between states in our case).

The solution of the problem lies in the construction of the A*llCompletedRequests()* action (or operator). Here, we encounter our first problem with AI planning using STRIPS: the universal quantifier does not exist in STRIPS. How then can we express the action: "*All* requests completed"? We can work around this difficulty by representing the action in clauses that resemble first-order logic as follows:

For all x, $Completed(x,S_i)) \rightarrow \neg Todo(x,S_i)$ *when* $Treated(x,y)$

This means that for any request x in state S_i, it is considered completed if its status in that state becomes "no more to do" ($\neg$) when treated by a technician y. The design of our planner now hinges on further recursive decomposition of the *Completed()* action until all the atomic actions are programmable. To assist with understanding of the next steps, we will now consider a simple example with only two requests in the initial state S_0: one request of type Ano and the other, of type Help, and two technicians available for work, Mike and Tom, with Mike skilled in Ano-requests and Tom, in Help-requests. We will also write all formulas in Conjunctive Normal Form (CNF) [9].

Initial state. The formula for the initial state is as follows (to simplify, we omit the formulation of the request priorities at this point):

$Todo(x_1,S_0)\wedge Todo\ (x_2,S_0)\wedge$
$Available(Mike,S_0)\wedge Special(Mike,'ANO') \wedge$
$Available(Tom,S_0)\wedge Special(Tom,'HELP')$

Final state or goal state. The desired outcome would be the completion of both requests in the final state S_2. The logical formula for a completed request x_1 and x_2 in state S_2 would be:

$Completed(x_1,S_2) \wedge Completed(x_2,S_2)$

Actions. We start with the *Completed()* action, which could be described as follows: a request x_1 is completed with result a in situation S if and only if a is returned by the precondition *Treated*() and the request x_1 has not stayed in *Todo* status (i.e., $\neg Todo()$):

$Completed(x_1,S) \Leftrightarrow [Treated(x_1,Tech) \wedge \neg Todo(x_1,S)]$

By backtracking search, we find the effects of the *Completed()* action: request x_1 is "not to do", the technician becomes available and x_1 is no longer assigned to him. Continuing to decompose further, we now have to deal with the *Treated()* action, which is the precondition to the execution of *Completed().* In the real world, the *Treated()* action represents the time spent by the technician Mike to resolve the client's request. He has to liaise with the client, completes the job, reports back to his supervisor, and finally updates the request database with details of the problem and of what he has done. *Treated()* in turn introduces the precondition *Assigned()*, which determines which technician to be assigned to a particular request. The *Assigned()* action is formulated as follows (with simplified technician selection criteria):

$$Assigned(req1, Result(Tech1 | Tech2,S)) \Leftrightarrow [(available(Tech1,S) \wedge special(Tech1)=special(req1)) \vee (FindTech(req1,Tech2)) \wedge \neg Treated(reqX,Tech2))]$$

In the above, the *Result()* action in situation *S* can return result *a* or result *b* (each of which is a technician's name). The *special()* action returns a constant variable: Help or Anomaly. In case all technicians are busy on other requests, the *FindTech()* action is a sub-goal that attempts to locate a technician who has not started work on his current request. Its logic formula is as follows:

$$FindTech(req1,Result(Tech,S)) \Leftrightarrow [\neg Available(Tech,S) \wedge Assigned(reqX,Tech) \wedge \neg Treated(Tech,reqX) \wedge (priority(req1)>priority(reqX))]$$

5. IMPLEMENTATION

We have implemented the above AI planner for Help Desk with the ILOG JRules software package. This package is selected because of its suitability for this Help Desk application and its easy integration with other Java code. JRules is a complete business rule management system (BRMS) for Java. Policy managers and developers capture business logic as business rules [4], which can then be encoded in JRules language and embedded in Web, legacy or back-office applications. JRules considers all objects and rules in a working memory space as first-order logic, expressed in the general "if …then…else…" format. Note that depending on how well the real-world problem is described in input to an AI planner, the resulting AI plan for that problem could be determinist or non-determinist. For example, if we don't declare any technician with the "repair" skills in the initial state, the resulting AI plan will not satisfy the goal when there are "Ano" requests present in the system as they will always remain indefinitely unassigned. This highlights the importance of business analysis, in addition to having suitable methodologies and adequate software tools, in building any system for business use.

6. CONCLUSION AND FUTURE WORK

This paper describes the use of AI planning to build an expert decision system to assist a service center in managing its user support requests and the workload of its staff. The system relies on partial-order planning, situation calculus, first-order logic, and other techniques such as STRIPS knowledge representational language, to formalize the different steps of the process of assigning a technician to a new service request. Our top-down reasoning approach demonstrates the use of AI-based partial-order planning to solve complex problems by starting from a simple model, then progressively decomposing it into finer components, until they are all computationally realizable. This approach enables production of concise but optimized plans by avoiding having to examine unnecessary situations. Our next step will focus on the refinement of the priority management algorithm described in Section 3 and on other real-life applications of our technique. Our approach could also be enhanced by other AI techniques such as belief network, to determine the initial priority of a new request, its maximum allowable waiting time and its estimated completion time, based on the assessments and beliefs of the Help Desk operators and technicians. Another area of future work could be the management of Help Desk staff. When multiple technicians are available for assignment to a new job, the system could take into account additional factors such as the relative skills of the different technicians in handling the same type of request and in meeting other request requirements (such as specific data security expertise), the current and past workloads of the technicians and their past performance in user support. This could result in a plan for staff management and staff performance appraisal.

REFERENCES

[1] Barrett, A., Weld, D., "Partial-order planning: Evaluating possible efficiency gains," *Artificial Intelligence*, Vol. 67, pp. 71-112, 1994.

[2] Breek, P., Chen, X., "CPlan: A constraint programming approach to planning," *Proc. Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 585-590, 1999.

[3] Fikes, R., Nilsson, N., "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, Vol. 2, pp. 189-208, 1971.

[4] ILOG JRules, http://www.ilog.com.

[5] Kautz, H., Selman, B., "BLACKBOX: A new approach to the application of theorem proving to problem solving," *Workshop Planning as Combinatorial Search*, *AIPS-98*, Pittsburgh, PA, 1998.

[6] Otwell, C., Remshagen, A., Truemper, K., "An effective QBF solver for planning problem," *Proc. International Conference on Algorithmic Mathematics and Computer Science*, pp. 311-316, 2004.

[7] Minton, S., Bresina, J., Drummond, M., "Total-order and partial-order planning: A comparative analysis," *Journal of Artificial Intelligence Research*, 1994.

[8] Rintanen, J. "Constructing conditional plans by a theorem-prover," *Journal of Artificial Intelligence Research*, Vol.10, pp.323-352, 1999.

[9] Russell, S., Norvig, P., "Artificial Intelligence: A modern approach," *Pearson Education*, 1995.

[10] Weld, D., "Recent advances in AI planning," *AI magazine*, 1998.

[11] Wilkins, D., "Can AI planners solve practical problems?" *Computational Intelligence*, Vol. 6, No. 4, pp. 232-246, 1990.

[12] Wilkins, D., "Practical planning: Extending the classical AI planning paradigm," *Morgan Kaufmann Pub.*, San Mateo, CA, 1988.