A PROPOSAL FOR A ROBOTIC PLANNING/REPLANNING FRAMEWORK

Şule Yıldırım* and Turhan Tunalı*

ABSTRACT

In this study, a new robotic task replanning architecture for a class of problems is proposed. The architecture supports replanning in deliberation layer in addition to the unexpected event handling of low level behaviours. The replanning mechanism is supported by vision feedback to update the internal state of the system at certain intervals for exegoneous event detection. The expected state is known after each plan step execution. High level plans are compiled into low-level behaviours for execution and some unexpected events that do not need deliberation layer interference are handled with low-level behaviours supported by sensory feedback.

Keywords: robotics, unstructured robotic environments, operation safety and efficiency, planning, replanning, robotic architectures, vision feedback.

1. INTRODUCTION

Robotics research is being done on building up robust robot systems for several decades by now. However, the state of art shows us that the deliberation side of these architectures needs more attention for increasing its contibution in exegoneous event handling. Although using low level behaviours as in Subsumption Achitecture [2] for unexpected event handling solves many problems, the opportunities that might arise with using high level behaviours need to be considered. The architecture in this study is an integrated deliberative planning and replanning system that also enables learning from past experience. The learning side of the system comes from the fact that it stores previous plans by indexing into the plan database by using initial and goal state pairs. A function defined takes initial and goal states as inputs and produces an index into the plan database for a previously stored plan. The other side of learning is to use a vision system that updates the world model after each execution step. This is currently possible by taking robot arm away until the camera grabs an image of the environment and proccesses the image for an unexpected event detection and goes to the pose of the most recent move to continue with the rest of the plan if there's not something unexpected.

Today, most of the studies concentrate on robots getting the best understanding of the world by forming world maps at the begining of the system execution. But it's always possible that the environment is dynamic and changing frequently. So the word "replanning" in our study corresponds to figuring out the next sequence of actions in case of a world change. Hence, in our study the robot gets the knowledge of the environment after each execution step to figure out changes. This aspect can be made continuous depending on how long it takes to process the

* International Computer Institute, Ege University, Bornova Izmir Turkey, yildirim@ube.ege.edu.tr, Research Assistant, tunali@ube.ege.edu.tr, Professor and Director.

image grabbed to get the current world model which will also be called observed world model.

The architecture to be used in robotics tasks is mostly affected by the type of tasks the robotic planner is going to produce plans for. However there are studies on producing domain-independent planners. As a result, the architecture formed in this study is somewhat dependent on the problems addressed although it has the fexibility to replace the embedded modules with the new ones that solve other problems. There are two sources of complexity in planning:

Satisfiability: the difficulty of finding any solution to the planning problem (regardless of the quality of the solution).
Optimization: the difficulty of finding the optimal solution under a given cost metric.

In particular, there are many domains in which t

he satisfiability problem is relatively easy and their complexity is dominated by the optimization problem. For example, there may be many plans that would solve the problem, so that finding a plan is easy in practice, but the cost of each solution varies greatly, thus finding the optimal one is computationally hard. We will refer to these domains as optimization domains. The planner currently used is a means-ends analysis planner which does heuristic local search**.** The efficiency of some other methods in literature for optimal planning will also be tested for the class of problems in hand. The methods that can be considered are iterative improvement, variable-depth search, simulated annealing [9] and tabu search [7]. The replanning mechanism we use also helps us to improve our plans in time as far as cost optimization is concerned.

The emphasis in the plans generated in our study is not on the selection among a variety of operators but on the binding of the operators with the correct values so that chosen values will result in an optimal travelled distance and minimal plan generation time. A recent study concentrates on rewriting planning rules in terms of replacing operators, links in a plan rule with the new ones if they will cause a more cost effective plan [1]. Veloso shows how to do plan transformations when the state changes by user intervention [14].

The robots can be working on their owns or they might be cooperating with each other using messaging or other distributed information sharing mechanisms. Generally a task scheduler at the highest level organizes the sequence of tasks including arranging message receive/send times. However, the scheduler might have less burden in interrupt driven systems depending on the real time importance of the tasks to be achieved since message handling wouldn't need to be sequenced.

The task level scheduler in this study, is given as an algorithm in Figure 1. It calls the image capturing and processing routines, evaluates the images to construct world models, calculates costs in scope of the replanning mechanism and hence schedules replanner and calls executive layer functions incoporating low level behaviours to accomplish low level tasks such as basic robot movements. The sequence of the tasks can be different depending on how important real time

processing is and hence reactivity needs to be stressed rather than allowing replanning. It has also been the concern of the robotic AI since 1980s to figure out whether planning or reactivity is most effective in achieving robust robots and one side gains partial or full stress over the other depending on the type of the problem domain. Currently there are pure AI planning researchers as well as there are robotic planning researchers. The robotic planning researchers tend to apply methods of AI planning in robotic planning. However, there are other researchers who use the final architecture, 3T, as robotic architectures or its modifications [6]. These ones tend to concentrate on the reactive and sequencing layer of the architecture rather than the top level planning layer. In this study, we show that there might be robotic tasks that might need both reactive level and task planning level unexpected event handling. That's unexpected events that need to be solved by reactivity will use reactive layer for event handling but there will be tasks that will need to be solved with the replanning module closely integrated with the task level planner. The examples of reactive execution is to drop an object the robot is carrying or the arm is trying to put an object on another one while it's not supposed to do so. On the other hand, when the places of the objects in the robotic environment changes, a new plan will be necessary which replanning will be the task of the replanner.

2. PROPOSED ARCHITECTURE

The idea in planning is to integrate a vision system to detect timely changes in the domain by the vision system. This limits us to examine problems where the environment image can be grabbed from a top view of a single camera and hence we can't deal with problems such as putting blocks one on the other for example. However the effort in finding the best solution for computationaly hard problems compensates for not having three dimensional vision capabilities.

The general execution within the frame work is as follows: The task scheduler does Neural Network training with sample images of the environment and waits for user input to receive initial and goal world states. Then it calls the planner with the initial and goal states. The planner uses means-ends analysis to produce a plan. The produced plan steps are kept in a global plan file after which the scheduler schedules each of the plan steps until all of them are executed successfully. Execution of each step by the robot arm is followed by grabbing an image of the environment, segmenting the image into square cells and recognition of the object labels in each segment. The objects are labeled with letters that represents each object. Segmenting the image of the environment leads to the formation of the Observed_World_Model. If the Observed_World_Model is the same as the Expected_World_Model, then the execution of the last plan step was successful and there was nothing unexpected in the environment. However, if the two models are different (partially or completely), then the replanning module is scheduled for handling the results of the unexpected happening.
The replanning mechanism stores intermediate states for an initial and a goal state pair which are known to be optimal or suboptimal from the intermediate state to the goal state. For that reason, the replanner considers intermediate states as one of the alternatives in addition to the two other alternatives in a previous work.

Refer to [16] for these alternatives and a replanning decision mechanism implemented on the "Shuffled Pieces" problem.

Task Scheduler:

```
main() {
NN-training ;
user input();
Planner (Initial-State, Goal-State); // Produces steps like MOVE A (1,1) (1,8)

while(there is a plan command to execute) {

        RobotController (MOVE MISPLACED-PIECE (a,b) (c,d));
        Copy Expected-World-Model to Prev-World-Model;
        Update Expected-World-Model;
        Grab an image;
        Segment the grabbed image;
        While(!end of image) {
                NN-recognition( a cell from the image);
        }//form Observed-World-Model
        Replanner();
} // while
}

Replanner(){
intermediate-State = get-suboptimal-intermediate-state-on-the current-path();
dm1 = compare(Intermediate-State, Observed-World-Model);
dm = compare(Expected-World-Model, Observed-World-Model);
if(dm1 < thr && dm1 < dm) {
        new-plan = get-the-plan-from-intermediate-state-to-goal-state;
        execute new-plan;
}
else {
        if(dm!=0 && dm < thr) {
                alter1-cost();
                Planner(Observed-World-Model, Previous-World-Model);
                while(there is a plan command to execute) {
                        robot-controller(); // executive layer & behaviours
                        update Expected-World-Model;
                } //while
        } //if
        else {
                alter2-cost();
                Planner(Observed-World-Model, Goal-State);
                while(there is a plan command to execute) {
                        robot-controller(); // executive layer & behaviours
                        update Expected-World-Model;
                } //while  will not continue with old plan

        } // else
  } //else
} // Replanner
```

Figure 1: Task Scheduler

In scope of the explanations above, the general framework for the architecture is as in Figure 2.

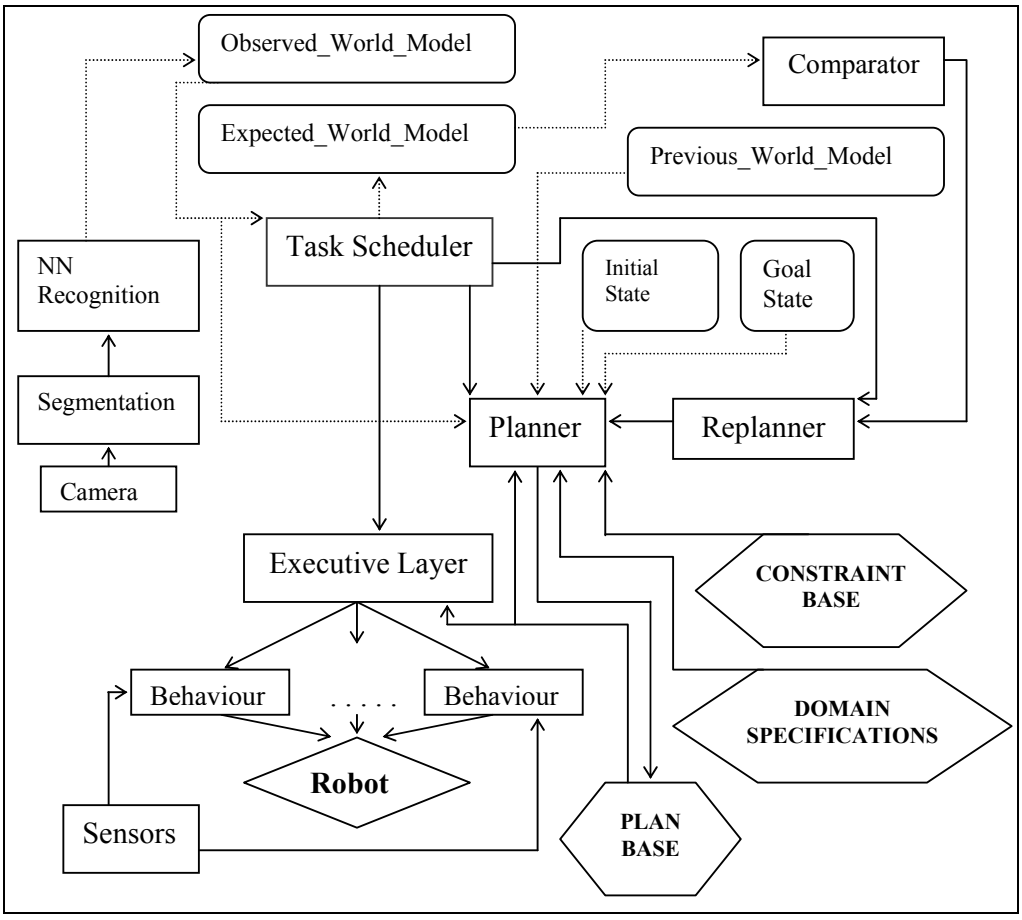

Figure 2: Proposed Framework.

The behaviours in Figure 2 are suppressed when there's a replanning decision. The behaviours have ("MOVE PIECE_NAME (X1 Y1) (X2 Y2) ", supress-value) as the parameters. If the supress-value = 1 then that means an unexpected event was detected from the scanned image.

3. THE DESCRIPTION OF THE VISION GUIDED PLANNER (VGP)

The field of AI planning seeks to build control algorithms that enable an agent to synthesize a course of action that will achieve its goals [15].

Methods for compiling planning problems into propositional formulae for solution using the latest, speedy systematic and stochastic SAT algorithms in particular have attracted much attention. These methods are impacted by recent progress in constraint satisfaction and search technology and they have quite impressive

performance level. For example, BLACKBOX planner [8] requires only six minutes to find a 105-action logistics plan in a world with 10^{16} possible states.
In the STRIPS representation, each action is described with a conjunctive precondition and conjunctive effect that defines a transition function from worlds to worlds. The action can be executed in any world satisfying the precondition formula. To execute an action in a world is described by taking the state description of the world and adding each literal from the action's effect conjunction to the state to eliminate the contradictory literals along the way.

A simple formulation of the planning problem defines three inputs:
1. A description of initial state or a description of initial state of the world in some formal language, i.e., predicate calculus.
2. A description of the agent's goal(s) i.e. what behavior is desired, in some formal language.
3.The operator to match preconditions into actions.

The planner in this study uses STRIPS representation of actions, goals and intial state and uses means-ends analysis to extract actions. We are not aiming at generating a planner that has better performance than the ones in the literature but to realize an integrated system architecture to solve planning problems and do vision based replanning at high level of abstraction and present a replanning approach that extends the idea of replanning from replacing operators with the new ones in a newly generated state after an unexpected event(s) occur to making decisions among alternatives for best actions in terms of cost optimization. Since we consider cost optimization for replanning and since the replanning idea is highly dependent on the planner itself, an improvement in the planner's performance will also affect the replanning performance. As future work, we plan to compare the cost optimization performance of replanning with a problem specific algorithm to the performances obtained by using means–ends analysis.

Some recent work uses "dynamic CSP (Constraint Satisfaction Problems)**"** to optimize planning problems as a result of observation of similarities between CSP to the planning solutions [5]. Dynamic CSP is a constraint satisfaction problem in which the set of variables and associated constraints change based on the selection of values to earlier variables. This is also the point that causes the computational intractability in the shuffled pieces problem [16]. Eventually all variables must have values assigned but the order in which they are selected can have a huge impact on efficiency. In general, a good heuristic is to select the variable with the fewest remaining and non-conflicting values. However, in the shuffled pieces problem, the variables don't conflict with each other but the problem is to get a sequence of moves that will be optimal.

A planner can be made as specific as possible using a specific algorithm for the optimization considerations of a problem or can use a common inferencing mechanism for many domains. These two alternatives may result in a single planner when the problem is not computationally tractable and hence possible to define the solution in form of constraints and control rules. In this study a means-end analysis planner is used where it doesn't seem to be possible to define an optimal or a suboptimal solution in form of defining constraints and control rules.

It doesn't seem possible to embed the algorithm in one of the methods in literature to approach an optimal solution.

Researchers have begun investigating the possibility of relaxing the perfect knowledge assumptions while staying close to the framework of classical planning in 1990s [11,4,12,3,13].

Cassandra [12], a contingency planner whose plans have the following features:
1-The plans include specific decision steps to determine which of the possible courses of action to pursue.
2-Information gathering steps are distinct from decision steps.
3-The circumstances in which it is possible to perform an action are distinguished from those in which it is necessary to perform it.

If there are many decision steps whose preconditions are satisfied in current state and hence each one is applicable, select the one that will minimize overall cost. In addition, in case of unexpected events, not only decide to backtrack to remove the effects of unexpected happening but also "make a decision between backtracking or going directly to a final state". This mechanism is actually enriched by applying a previously stored plan if there's one already. There are previously designed intermediate states that are known to have the best path from where they are to the final state. If the diffence between one of these states and the unexpected state is below some threshold, then it will then directly go to that state to execute the plan from that state to the goal.

Contingency planning is only one approach to the problem of planning under uncertainty [13]. The aim of contingency planning is to construct a single plan that will succeed in all circumstances, it is essentially an extension of classical planning. There are other approaches to planning under uncertainty that do not share this aim. Probabilistic planners aim to construct plans that have a high probability of success. Systems that interleave planning and execution do not attempt to plan fully in advance. In both of these approaches, it is possible to address the problem of determining which contingencies should be planned for, which is not currently possible in Cassandra. A third approach is that of reactive planning, in which behavior is controlled by a set of reaction rules.

4. THE PROBLEM DOMAIN

We define domains where replanning is an important concept to handle events. We want to show that while doing replanning we might need to consider different alternatives in order to main cost effectiveness instead of choosing a step to execute to get rid of the unexpected effects of a happening.

The defined domains are the shuffled pieces problem, chess game, component insertion in an electronic board and box transportation in factories. The domain base is possible to grow as far as the nature of the domain is consistent with the optimization replanning idea. That is, the cost of travelling is important and the users have optimization considerations. The vision capabilities such as three-dimensional vision has importance in increasing the domain types. Constraints are embeded in the control rules.

Search control rules are provided to reduce the number of choices at each decision point by pruning the search space or suggesting a course of action while expanding the plan. Control rules are *if-then* rules that indicate which choices should be made (or avoided) depending on the current state and other meta-level information. In particular, control rules can select, prefer or reject specific planning choices at every decision point [10]. Control rules can be used to focus planning on particular goals and towards desirable plans. **VGP** planner uses a control rule that either controls the movement of the arm to the closest misplaced piece or moves the misplaced piece to the closest goal using heuristic search. In Table 1, these two situations combine into a single rule using the *if-then-else* structure. The operators that will be enabled in the control rule are defined in Table 3. Table 2 defines the object structure that holds the characteristics of the objects in the environment and needs to be instantiated (object type (ARM/BLOCK, current object location, etc.). The actions (move_block, move_arm) are recorded as plan steps.

```
if (!strcmp(obj.type,"BLOCK") &&                          } Preconditions A
  find_closest_destination_from_block(&obj) &&
  !empty_destination(&obj)) {
    move_block(&obj);                                     } Actions A
    strcpy(status,"arm-is-moving");}
else if (!strcmp(obj.type,"ARM") &&
     find_closest_destination_from_arm(&obj)              } Preconditions B
     !empty_destination(&obj)) {
       move_arm(&obj);
       strcpy(status,"block-is-moving");}                 } Effects B
```

Table 1: Control Rule

Preconditions A and the Actions A will hold for the situation when a block has already been decided on for being carried to its destination. Preconditions A check whether the object to be moved is a block, whether its closest destination has been determined and whether the destination is empty. If all of these preconditions hold in the current state, Actions A are executed. Actions A include moving the block to its empty destination. If the destination is not empty, this situation is detected in function empty_destination(&obj) and the occupied destination is emptied by carrying the occupying block to the closest empty place. If the object to be moved is the arm but not the block, this means the arm seeks to find the closest block to itself to carry the block to its destination. Thus, Preconditions B check whether the object to be moved is the arm, whether the destination that the arm will be moving to is empty or full (supposed to be full because there will be a block there) and whether the arm has chosen the closest block to itself. If all these conditions hold, the arm moves to the closest block with actions B. The status is updated to be "arm-is-moving" or "block-is-moving" depending on the current status of the planner.

```
Object {
      char type[ ];                          // Arm/Piece
      char name;                             // P(iyon), V(ezir), K(ale), S(ah), A(t),F(il)
      char destinationOccupancy[ ];          // Full / Empty
      char distanceToDestination[ ] ;        // Closest / Unknown
       int distance, destx, desty, curlocx, curlocy;
      int processed;
 } ;
```

Table 2: Object structure

```
void move_block(object *obj){
   if(!strcmp(status,"block-is-moving") &&
     !strcmp(obj-estinationOccupancy,"EMPTY")){
       strcpy(obj->type,"ARM");
       Desired_World_Model[obj->destx][obj->desty] = *obj;
       Desired_World_Model[obj->curlocx][obj->curlocy].name = ' ';
       Goal[obj->destx][obj->desty].processed = 1;
       Initial[obj->curlocx][obj->curlocy].processed = 1;
       obj->name = ' ';
       obj->curlocx = obj->destx;
       obj->curlocy = obj->desty;
       obj->destx = -1;
       obj->desty = -1;
       strcpy(obj->destinationOccupancy, "FULL");
      obj->distance = -1;

}
```

(a) Move Block

```
void move_arm(object *obj){
   if(!strcmp(status,"arm-is-moving") &&
      !strcmp(obj->destinationOccupancy,"FULL"));
     strcpy(obj->type,"MISPLACED");
     obj->name = Desired_World_Model[obj->destx][obj->desty].name;
     obj->curlocx = obj->destx;
     obj->curlocy = obj->desty;
     obj->destx = -1;
     obj->desty = -1;
     obj->processed = 0;
     strcpy(obj->destinationOccupancy, "UNKNOWN");
     obj->distance = -1;
}
```

(b) Move Arm

Table 3: The primary operators in VGP's task planning domain

5. CONCLUSION

In this study, a vision guided architecture is proposed as a framework for the planning/replanning of tasks for a class of robotic problems. The main difference of the proposed architecture from the current architectures in literature is its functionally at the deliberation layer with vision supported replanning. Our architecture incorporates sequencing and reactive layers of 3-Tier architecture. However, it diverges the load on the lowest two layers (sequencing and reactive)

of the 3-Tier architecture partially to the deliberation layer. This feature is necessary as it's shown in this paper that some of robotic tasks might need modifications in their high level plans and the lowest two layers of the 3-Tier architecture will be insufficient for handling those needs.

REFERENCES

1. Ambite, J. L. and Knoblock, C. A., 2001, "Planning by Rewriting", Journal of Artificial Intelligence Research 15, pp. 207-261.

2. Arkin, R. C., "Behaviour-Based Robotics", The MIT Press, Cambridge, Massachusetts, 1999.

3. Draper, D., Hanks, S., & Weld, D.,1994, "A probabilistic model of action for least_commitment planning with information gathering", In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, pp. 178-186, Seattle, WA. Morgan Kaufmann.

4. Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N. & Williamson, M., 1992, "An approach to planning with incomplete information", In Proceedings of the Third International Conference on Knowledge Representation and Reasoning, pp. 115-125, Boston, MA, Morgan Kaufmann.

5. Falkenhainer, B, Forbus, K., 1998, "Setting up large scale qualitative models", In Proc. 7th National Conf. AI, pp. 301-306.

6. Gat, E., 1998, "Three Layer Architectures", in Artificial Intelligence and Mobile Robots, MIT Press.

7. Glover, F., 1989, "Tabu search-Part 1", ORSA Journal on Computing, 1 (3), pp.190-206.

8. Kautz, H. , Selman, B., 1998, "Blackbox: A new approach to the application of theorem proving to problem solving", In AIPS98 Workshop on Planning as Combinatorial Search, pp. 58-60.

9. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by simulated annealing", Science, 220, pp. 671-680.

10. Laird, E. L., Congdon, C. B., and Coulter, K. J., 1998, "The Soar User's Manual Version 8.2", University of Michigan.

11. Peot, M. A., Smith, D. E., 1992, "Conditional nonlinear planning", In Proceedings of the First International Conference on Artificial Intelligence Planning Systems, pp189-197, College Park, Maryland, Morgan Kaufmann.

12. Pryor, L. & Collins, G., 1993, "Cassandra: Planning with contingencies", Technical report 41, Institute for the Learning Sciences, Northwestern University.

13. Pryor, L., & Collins, G., 1996, "Planning for Contingencies: A Decision_based Approach", Journal of Artificial Intelligence Research, 4, pp. 287-339.

14. Veloso, M. M., Carbonell, J., P'erez, M. A, Borrajo, D., Fink, E., and Blythe, J., 1995, "Integrating planning and learning: The Prodigy architecture", Journal of Experimental and Theoretical Artificial Intelligence, 7(1), pp. 81-120.

15. Weld, D. S., 1999, “Recent Advances in AI planning”, AI Magazine.

16. Yıldırım, Ş. and Tunalı, T., 1999, “A new methodology for dealing with uncertainty in robotic tasks”, XIV. Int. Symp. on Comp.& Inf.Sci., Kuşadası, TURKİYE.