
IMPROVING THE PLANNING SOLUTION QUALITY BY REPLANNING

Flavio Tonidandel² and Marcio Rillo^{1,2}

¹ Universidade de São Paulo - Escola Politécnica
Av. Luciano Gualberto 158, trav3
05508-900 - São Paulo - SP - Brazil

² Centro Universitário da FEI - UniFEI
Av. Humberto de A. Castelo Branco, 3972
09850-901 - São Bernardo do Campo - SP - Brazil
flaviot@fei.edu.br, rillo@lsi.usp.br

Resumo: A qualidade de uma solução é crucial em muitas aplicações de Inteligência Artificial, inclusive para as soluções de um sistema de planejamento. Entretanto, não existe nenhum método independente do domínio que possa melhorar a qualidade de um plano de um sistema de planejamento. Este artigo apresenta um método independente do domínio que melhora a qualidade de uma solução determinando segmentos de planos que precisam ser replanejados. Este método, chamado SQUIRE, não necessita de nenhum conhecimento adicional e apresenta bons resultados em testes empíricos. A versão do método neste artigo é baseada no modelo STRIPS e, portanto, se foca na redução do número de ações de uma solução de planejamento.

Palavras-Chaves: *Qualidade de plano; Planejamento Inteligente de Ações.*

Abstract: The quality of a solution plan is crucial in many AI Planning applications. However, there is no effective domain-independent method, which improves a solution plan quality. This paper proposes a domain-independent and anytime behavior method that improves the quality of a solution plan by determining segments for replanning. This method, called SQUIRE, does not need any additional knowledge and it presents some useful results in the empirical tests. The version of the method in this paper is based on a STRIPS model and, therefore, it improves the quality of a plan by reducing the number of steps on it.

Keywords: *Plan Quality; Anytime Replanning; Domain-Independent Classical Planning*

1. INTRODUCTION

Solution quality is important in many AI applications. In planning research field, this quality refers to the solution plan length and the rational use of resources and time.

The concerning about quality is not an easy task. Finding a solution in domain-independent planning is already a hard task (Bylander, 1994), and this task becomes even harder if the quality of the planning solution is taken account.

A possible approach to find better quality plans is the adaptation of plans. Gerevini and Serina (2000) propose the ADJUST-PLAN algorithm that adapts a plan to find a solution of a planning problem. However, their algorithm was not designed to improve the quality of a solution plan.

On the other hand, the Planning by Rewriting paradigm (Ambite, Knoblock 2001) addresses the problem of adapting a solution plan through rewriting rules in order to find a better quality plan as a solution. However, the rewriting rules are domain-dependent rules that can be determined by hand or by a learning process (Ambite, Knoblock, 2000).

Differently of the previous approaches, this paper presents an effective domain-independent method with anytime behavior to improve the quality of a solution plan. This method, called SQUIRE (*Solution Quality Improvement by Replanning*), uses the *FF-heuristic* (Hoffmann, Nebel, 2001) to detect sub-plans in the solution plan that can be replanned.

This paper focuses in a STRIPS-version of the SQUIRE method, where just the number of steps can state the quality of a plan in a solution plan. Its natural evolution to metrical and temporal domains is discussed in the discussion section.

2. SOLUTION QUALITY AND PLAN ADAPTATION

In the planning competition, besides the planning time and the number of problems solved in an specific domain, the solution quality are one of the most important parameters that is used to confront the planning systems performance. In addition, the quality of a solution plan is crucial for some real applications

Following (Bylander, 1994), finding an optimal planning problem solution is *NP-hard*. However, it is possible to use plan adaptation in order to find sub-optimal solutions efficiently.

Adapting a plan, in theoretical point of view, is also *NP-hard* (Nebel, Koehler, 1995). However, as shown by Gerevini and Serina (2000), the adaptation of an existing plan can be more efficient than generate a plan from scratch.

An adaptation method that refines a plan is the ADJUST-PLAN Method (Gerevini, Serina, 2000). It tries to find a complete solution from a given plan by refining sub-plans on the graph created by the Graphplan system (Blum, Furst, 1995). The given plan can be obtained by a retrieval phase in a case-based planning for instance.

However, the ADJUST-PLAN can not be applied to a solution plan, since the method in Gerevini and Serina (2000) is designed to work only until a solution is found.

* This work was supported by FAPESP under contract no. 98/15835-9.

Similar to a plan adaptation process is a replanning process called SHERPA Replanner (Koenig et al, 2002). The SHERPA algorithm, although it is not designed to improve the solution quality, it finds replanned solutions which quality are as good as those achieved from scratch (Koenig et al, 2002).

The adaptation of a plan can be also useful for generative planning systems. The new planning paradigm, called Planning by Rewriting (PbR), uses an adaptation phase, called rewriting, that turns a low quality plan into a high quality plan by using some domain dependent rules.

The PbR is a planning process that finds an easy-to-generate solution plan and then adapts it to a better solution by some domain-dependent rules. These rules can be designed by hand or through an automated learning process (Ambite, KnoBlock, 2000). In both cases, the PbR needs an additional domain specific knowledge, either a complete specification of the rules or some training instances for the learning process.

In fact, there is no effective domain-independent method, which improves the solution quality without any additional knowledge besides the planning standard knowledge - operators' specification, initial and goal states.

This paper focuses on a domain-independent and anytime behavior method of replanning, called SQUIRE (*Solution Quality Improvement by Replanning*), that adapts a low quality solution plan to a better quality plan. This method does not require any additional domain knowledge than operators specification, initial state and a goal state, and it is very useful for case-based planners whose solutions are usually longer than necessary (Tonidandel, Rillo, 2002).

3. THE ADAPTATION OF PLANS

The challenge of the most plan adaptation processes is to determine which part of the plan must be adapted.

A solution plan is usually a network of actions. This network is a part of a complete network composed of all states and all actions in a specific domain. For total order planners, a solution plan is necessarily a sequence of actions, and it can be represented as a path in a directed graph where each node is a consistent state and each edge is an action.

Considering a distance heuristic function, h^d , like the FF-heuristic (Hoffmann, Nebel, 2001) or HSP-heuristic (Bonet, Geffner, 2001), it is possible to estimate the number of actions between two planning states. If this heuristic is applied to estimate the distance from an initial state to all other states in a directed graph, it is possible to determine the costs of reaching each state from the initial state in number of actions.

However, there is no accurate heuristic function to determine optimal costs for each state without domain specific rules. An approximation of these optimal costs can be obtained by using a heuristic function used by the Heuristic Search Planners.

Considering the recent results of the FF system in the last two planning competitions, the *FF-heuristic* is the best choice to be used in this cost estimation. It is also successfully used in other methods, like the ADG similarity (Tonidandel, Rillo, 2001) and the FAR-OFF system (Tonidandel, Rillo, 2002).

The method SQUIRE, in the following, uses the *FF-heuristic* to calculate the cost of each state and determining the segments of replanning.

4. THE SQUIRE METHOD

The process of the SQUIRE method is quite simple. It is a recursive and anytime behavior method that determines the segments of a solution plan that must be replanned. The process of replanning is made by a generative planning system, which is, in this STRIPS version, a modified version of the original FF planning system (Hoffmann, Nebel, 2001).

4.1. A modified version of the FF planner

The original FF planning system, as presented in (Hoffmann, Nebel, 2001), is designed to plan to a goal in a fast way, pruning states by using some methods and heuristics, like the *added-goal deletion* heuristic.

The *added-goal deletion* heuristic does not consider a state on which a goal that has been just reached can be deleted by the next steps of the plan. It is done by analyzing the delete lists of all actions in the relaxed solution provided by the *FF-heuristic*. If a reached goal is in the delete list of any action in the relaxed solution, the FF planner does not consider this state in the search tree.

It works fine when the goal contains no predicates that are easy to change, like *handempty* in the Blocks World domain or the *at(airplane,airport)* in the logistic domain. If one of these easy-to-change predicates is in the goal, the FF planner can not find a solution.

For example, consider that the initial state of a planning problem in Blocks World domain is *on(A,B)*, *clear(A)*, *ontable(B)* and *holding(C)*; and the goal is *on(A,C)* and *handempty*. The FF planning system will not find a solution because any other state from the initial state will have a reached goal – *handempty* – and any other action from this state will delete the *handempty* predicate. It will force the *added-goal deletion* heuristic to prune all states from the initial state and no solution is found.

Since the SQUIRE method will consider a complete and consistent state as a goal, it would experience some problems by using the original FF planning system to find an alternative plan. Therefore, in order to avoid this problem, a modified version of the FF planning system is implemented. In this version, a relaxed *added-goal deletion* heuristic is implemented.

The relaxed *added-goal deletion* heuristic does not prune a state if the predicate goal reached has one of the following conditions:

1. It is presented in the insert list of more than one action;
2. It is the unique predicate in a precondition of at least one action.

The first condition excludes predicates like *holding(_)* and *handempty* in Blocks World domain; and the second condition avoids that the *added-goal deletion* disables the application of an action because of a pruned state.

Besides the relaxed *added-goal deletion* heuristic, the modified version of the FF planner has also the following features:

- It does not switch to the complete Best First Search if a solution was not found by using Enforced Hill-Climbing search.
- It allows to avoid an specific action as the first action in the solution plan

```

procedure det_costs(plan , initst)
create_relaxed_graph(initst, ∞)
S0 ← initst;
H0 ← 0;
i ← 0;
while i < number of actions ∈ plan do
    i ← i+1;
    Si ← execute_action(Ai ∈ plan, Si-1)
    Hi ← determine_relaxed_solution(S0);
endwhile
end;

```

Fig. 1 – Algorithm of a function that determines the distances of each intermediary state from the initial state *initst* of a solution plan *plan*.

- It allows to specify the maximal length of a solution
- It allows to specify an upper time limit time to find a solution

The first feature described above is stated because it is not important to find a solution at any time cost; the other three features are stated to let the SQUIRE method to focus the FF planning to its purpose.

4.2. The Recursive Replanning Process

As its first phase, the SQUIRE method determines the cost of each intermediate state of a solution plan. The algorithm in the figure 1 calculates, given an initial state and a solution plan, the estimated distance of each intermediate state by using the *FF-heuristic* from the initial state.

In order to determine the final and the intermediate state, the algorithm executes each action of the plan from the initial state.

The functions *create_relaxed_graph* and *determine_relaxed_solution* in the figure 1 are algorithms extracted from the *FF-heuristic* (Hoffmann, Nebel, 2001).

As defined by Hoffmann and Nebel (2001), the relaxed graph is created by ignoring the delete list of the actions. It is constituted by layers that comprise alternative facts and actions. The first fact layer is the initial state (*initst*). The first action layer contains all actions whose preconditions are satisfied in *initst*. Then, the add lists of these actions are inserted in the next fact layer together with all facts from the previous fact layer, which leads to the next action layer, and so on. The symbol infinite (∞) in the *create_relaxed_graph* indicates that the relaxed graph must be created until a fixpoint is found, i.e., when there are no more fact layers that are different from previous ones.

With the relaxed graph created, the *determine_relaxed_solution* algorithm can be applied for finding a relaxed solution for any state that can be reached from *initst*. The process of determining the relaxed solution, following Hoffmann and Nebel (2001) is performed from the last layer to the first layer, finding and selecting actions in layer *i-1* which their add-list contains one or more of goals initialized in layer *i*. Then, the preconditions of the selected actions are initialized as new goals in their correspondent layer. The process stops when all unsatisfied goals are in the first layer, which is exactly the initial state. The relaxed solution is the selected actions in the graph and the estimate distance is the number of actions in this relaxed solution.

```

function det_Sr (<v0, v1, v2, v3... , vn>, from)
retv ← 0;
i ← from;
while (retv <> 0) and (i < n) do
    i ← i+1;
    if vi < vi-1 then retv ← i;
endwhile
return ← retv;
end;

```

Fig. 2. -- Algorithm to detect the first occurrence of *Sr* and *vr* in a plan

```

function det_Sb (<v0, v1, v2, v3... , vn>, r)
backv ← 0;
i ← r;
while (backv <> 0) and (i < n) do
    i ← i+1;
    if vi >= vi-1 then backv ← i;
endwhile
return ← backv;
end;

```

Fig. 3. -- Algorithm to detect the first occurrence of *Sb* and *vb* in a plan from the position *r*.

The algorithm of the figure 1 returns an array of costs that contains each intermediate state distance estimation values from the initial state. In an optimal or optimized plan, these values must increase constantly from the beginning to the end. Any value that is less than the anterior can indicate a hot point of replanning, because it indicates a possible return in the directed network of search. This value and its respective state are called *returned*.

Definition 1: (Returned State) *For a plan with <S₀, S₁, S₂, S₃, ..., S_n> intermediary states, a Sr state is a state S_i where its value, obtained by a heuristic function, h^d, is less than the value of S_{i-1}.*

This *returned* value indicates that its respective intermediate state is nearer to the initial state than the intermediate state immediately before it. The *returned* state and its *returned* value are indicated as *S_r* and *v_r*, respectively. The SQUIRE method detects the first occurrence of a *returned* state in the solution plan. The algorithm to determine the *Sr* and *vr* is presented in figure 2.

After detect a *Sr*, the SQUIRE method tries to detect the next intermediary state that comes back to pursue the goal. This intermediary state, called *Sb*, and its respectively value *vb*, indicates for the SQUIRE method the point where the plan returns to converge directly to the goal after the *Sr* state. The back State (*Sb*) and its value (*vb*) is determine as follows:

Definition 2: (Back State) *Given a plan with <S₀, S₁, S₂, S₃, ..., S_n> intermediary states, and a state Sr, a Sb state is a state S_i, with i > r, where its value, obtained by a heuristic function, h^d, is more than the S_{i-1} value.*

The main idea of the SQUIRE method is to create an alternative plan that bridges the correct segments of the solution plan with the segment of the plan that returned to pursue the goal after the state *Sb*. This alternative plan will substitute a misplaced sub-plan, called *Rp*, which the last part is defined by the *Sb* State. The figure 3 presents the algorithm that detects the back state and its value.

In order to determine the beginning of the misplaced sub-plan *Rp*, the SQUIRE method determines the first state, from the

```

function det_Sm(<v0,v1,v2,v3...,vn>, r, b)
  misv ← 0;
  i ← 0;
  for i ← 0 to r-1 do
    if (vi>misv) and (vi<vb) then misv ← vi;
  endfor;
  return ← misv;
end;

```

Fig. 4. -- Algorithm to detect the first occurrence of Sm and vm in a plan from the first state to the position r .

initial state, whose value is the highest value among all states before Sr but less than vb value.

This state, called Sm , becomes the initial state of a possible misplaced sub-plan, and its value is indicated as vm . Formally, Sm can be defined as follows:

Definition 3: (Sm) Given a plan with $\langle S_0, S_1, S_2, S_3, \dots, S_n \rangle$ intermediary states, and a state Sr and the vb value, a Sm state is a state S_i , with $i < r$, where its value vm is the highest value among all other intermediary states $\langle S_0, S_1, S_2, S_3, \dots, S_{r-1} \rangle$ and less than vb . The Rp sub-plan is formed by the actions between Sm and Sb . The Sm can be considered the initial state and the state Sb as the final state of the Rp sub-plan.

The algorithm that determines the position of Sm and vm in the solution plan is given in figure 4.

In fact, the SQUIRE method considers the Sm as the state immediately before a possible sub-plan with misplaced actions and that must be replanned.

However, the actions of the sub-plan Rp can not be misplaced by themselves, and the returned value can be caused by any other action before Sm . To determine if the Rp sub-plan really contains misplaced actions, the algorithm det_costs , presented in figure 1, is applied for the Rp sub-plan. If there is any returned value, the Rp becomes a potential misplaced sub-plan that must be replaced.

If the Rp does not contains any returned value by itself, then the initial state of the Rp sub-plan is *backtracked*. The new initial state becomes the state immediately before the state Sm . The algorithm det_costs is applied to this new Rp sub-plan in order to determine if it contains a returned state or not. The state backtracking continues until the initial state is reached or a misplaced sub-plan is detected.

When a misplaced sub-plan Rp is determined, the SQUIRE method starts the replanning process by calling the modified version of the FF planning system to create an alternative plan to the Rp sub-plan. This alternative plan must have fewer actions than the Rp sub-plan and it cannot have the same first action of the Rp sub-plan. The objective of changing the first action is to force the planning system to find an alternative solution. The state definitions and a Rp sub-plan is schematized in figure 5.

If no other alternative plan is found, the SQUIRE backtracks the initial state of the Rp sub-plan and tries again. The backtracking goes until the initial state of the solution plan is found or until an specific number, kb , of backtrackings are performed. If there is no backtracking and no alternative solution is found, the SQUIRE method continues to detect points of replanning after the Sb State.

On the other hand, if an alternative plan is found, it substitutes the Rp sub-plan and the SQUIRE method continues to detect

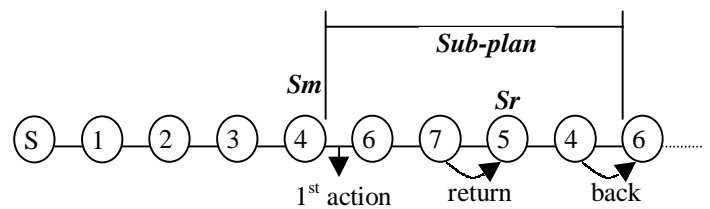


Fig. 5 – A solution plan with the intermediate state represented as circles with their respective values.

```

function SQUIRE(plan, initst, from)
  <v0, v1, . . . , vn>, ← det_costs(plan, initst, from);
  r ← det_Sr(<v0, v1, . . . , vn>, from);
  if r < > 0 then
    b ← det_Sb(<v0, v1, v2, . . . , vn>, r);
    m ← det_Sm(<v0, v1, v2, . . . , vn>, r, b);
    Rp ← plan between Sm and Sb;
    bktk ← 0;
    i ← m;
    isRp ← false;
    while (time < maxtime) and (i > 0) and
      (bktk < NumMaxbktk) and not(isRp) do
      <vp0, vp1, . . . , vpn> ← det_costs(Rp, Si, 0);
      if det_Sr(<vp0, vp1, vp2, . . . , vpn>, 0) = 0
        then isRp ← true;
      else
        i ← i-1;
        Rp ← plan between Si and Sb;
      endif;
    endwhile
    if isRp then Rp ← call modifiedFF with
      - prune the action = 1st action of Rp;
      - max profund ← b-i-1;
      - time < maxtime;
      if Rp = null then isRp ← false;
    endif;
    if not(isRp) then from ← b;
  else substitute Rp in plan between Si and Sb;
    if time < maxtime then
      SQUIRE(plan, initst, from);
    endif
  return ← plan;
end;

```

Fig. 6. – The SQUIRE algorithm. It needs a solution plan, an initial state and the first position of the plan that will be considered.

points of replanning after the Sb state until the final state is reached. The complete SQUIRE algorithm is presented in figure 6. The figure 7 shows the behavior of the backtracking for an example in the Blocks World domain.

Although the replanning process is fast because of the modified FF planning works with a complete state as goal and a limited length for the plan that must be find, the numbers of backtrackings and the complexity of a solution plan can turn the SQUIRE method into a time consuming process

Because of that, the SQUIRE method allows that the user defines the maximal time that the method can try to improve the solution quality and the maximal numbers of backtrackings that the process can do.

5. EMPIRICAL TESTS

The SQUIRE method was tested in some STRIPS-model domains, like Blocks World, Logistic and DriverLog domains.

		Blocks World (IPC 2000)		4-0	5-0	6-0	7-0	8-0	9-0	10-0	11-0	12-0	
FF System	Original	#steps	6	12	16	20	18	30	34	32	36		
	SQUIRE	#steps	6	12	12	20	18	30	34	32	36		
	Bcktk<15	Time sec	0.05	0.07	0.08	0.18	0.85	0.88	9.40	3.01	0.98		
	Bcktk unlimited	Time sec	0.03	0.06	0.08	0.27	1.17	61.58	200.9	5.22	6.60		
HSP	Original	#steps	6	12	12	26	18	40	98	44	36		
	SQUIRE	#steps	6	12	12	22	18	36	44	44	34		
	Bcktk<15	Time sec	0.1	0.06	0.03	0.83	0.18	5.59	20.85	2.38	0.52		
	Bcktk unlimited	Time sec	0.03	0.06	0.03	0.99	0.32	88.82	266.5	35.77	3.69		
FAR-OFF Case-Based Planner	Original	#steps	6	12	26	40	20	34	34	44	42		
	SQUIRE	#steps	6	12	12	20	20	34	34	34	42		
	Bcktk<15	Time sec	0.03	0.05	0.21	1.37	1.69	8.21	9.38	3.32	5.72		
	Bcktk unlimited	Time sec	0.03	0.05	0.20	1.51	2.80	99.75	200.4	67.83	37.20		
		Logistic Domain (IPC 2000)		4-0	5-0	6-0	7-0	8-0	9-0	10-0			
FF System	Original	#steps	49	49	25	36	31	36	46				
	SQUIRE	#steps	49	49	25	36	31	36	46				
	Bcktk<15	Time sec	0.09	0.04	0.04	0.04	0.04	0.04	0.04				
	Bcktk unlimited	Time sec	0.04	0.04	0.03	0.04	0.04	0.04	0.04				
FAR-OFF Case-Based Planner	Original	#steps	26	39	33	63	52	56	86				
	SQUIRE	#steps	26	33	33	53	43	50	80				
	Bcktk<15	Time sec	0.03	0.04	0.04	0.12	0.09	0.12	0.20				
	Bcktk unlimited	Time sec	0.03	0.05	0.03	0.25	0.12	0.20	0.25				
		DriverLog (IPC 2002)		Pfile1	Pfile2	Pfile3	Pfile4	Pfile5	Pfile6	Pfile7	Pfile8	Pfile9	Pfile10
LPG Quality	Original	#steps	7	20	12	16	18	17	13	22	23	17	
	SQUIRE	#steps	7	20	12	16	18	14	13	22	23	17	
	Bcktk<15	Time sec	0.03	0.10	0.03	0.11	0.80	0.35	0.03	0.03	0.13	0.03	
	Bcktk unlimited	Time sec	0.03	0.14	0.03	0.11	0.80	0.34	0.04	0.04	0.21	0.03	
FAR-OFF Case-Based Planner	Original	#steps	11	19	33	31	22	22	30	32	31	28	
	SQUIRE	#steps	9	19	20	29	22	16	26	30	31	23	
	Bcktk<15	Time sec	0.04	0.03	0.11	0.06	0.10	1.81	0.07	0.71	0.08	0.06	
	Bcktk unlimited	Time sec	0.04	0.03	0.11	0.12	0.11	1.82	0.15	1.22	0.15	0.06	

Table 1 – The results of the application of the SQUIRE method in HSP, FF, LPG and FAR-OFF results in the Blocks World, Logistic and DriverLog domains.

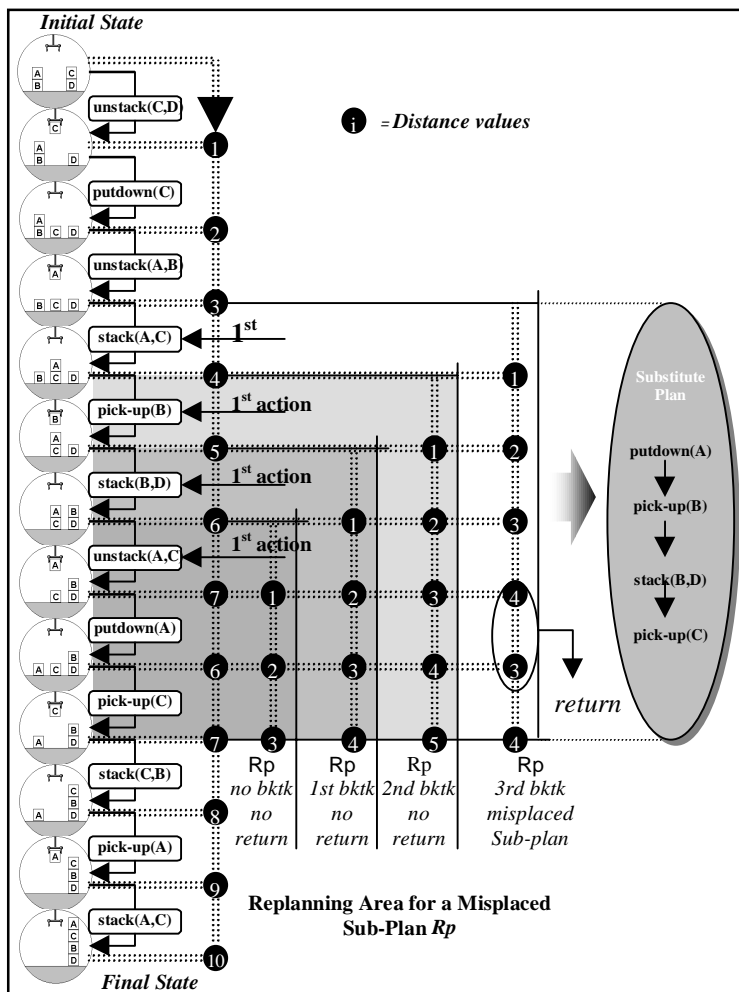


Fig. 7 – Example of the SQUIRE in the Blocks World domain

The solutions of the LPG (Gerevini, Serina, 2002)], FF (Hoffmann, Nebel, 2001) and HSP (Bonet, Geffner, 2001) planning systems in the last IPC'02 (Long, Fox, 2002) and IPC'00 (Bacchus, 2000) planning competitions and the solutions of the FAR-OFF case-based planning system (Tonidandel, Rillo, 2002) constitute the results set in the tests.

The tests considered two types of configurations: a limited number of backtrackings and an unlimited number of backtrackings. No tests imposed any time limit, although the SQUIRE method allows that the user defines a maximal time that can be used to improve the solution. Defining a limited time of replanning does not affect the solution of the planning problem, it only prunes the improvement of its quality.

The Table 1 shows the steps of the planning systems original results and the time and steps after the use of the SQUIRE method. The tests were performed in a Windows® Environment in a Pentium® III 450 MHz computer with 512 Mbytes of RAM memory.

It is possible to observe that the SQUIRE method is effective for the FAR-OFF system results in all domains considered in the tests. The FF system, also the LPG, present many optimal results, and the SQUIRE method improves only some of their results. The HSP has many bad quality results and the SQUIRE method can improve the quality in many of them.

The time spent by the SQUIRE method is proportional to the number of returned values and misplaced sub-plans. Since more actions a plan have, higher is the possible number of returned values and misplaced sub-plans, the SQUIRE method takes more time to adequate a solution plan with more actions.

In the Blocks World domain, the SQUIRE method is useful for all planning systems considered in the tests. It reduces, for

example, the solution of the problem 10-0 provided by the HSP system in more than 50%. It also reduces the solution of the problem 6-0 of the FF system to an optimal plan. For the FAR-OFF system, it reduces its solutions to near-optimal plans.

In the Logistic domain, the SQUIRE method is only effective for the FAR-OFF system, since the SQUIRE method does not change the FF results. In fact, the Logistic domain has many *equal-values* states for the *FF-heuristic* relaxed point of view, because the SQUIRE method does not detect any return value for these problems. On the other hand, the method gives an important quality improvement in the FAR-OFF results.

In the DriverLog domain, although the best quality results of the LPG system are considered in the tests, the SQUIRE method still improves some of them. However, the SQUIRE method can not reduce the results length to optimal solutions.

Some results can be reduced more with an unlimited number of backtrackings than with a limited number. It happens with the results in the Logistic Domain for the FAR-OFF system mostly. However, in a general frame, the use of unlimited backtracking is not relevant for the quality solutions, and it still spends much more time than when a limited of 15 backtracking is imposed. Therefore, the SQUIRE with limited number of backtracking is an effective tool to reduce the plan length.

6. DISCUSSION

Sometimes, the SQUIRE method tries to replace an optimal sub-plan. It is caused because the *FF-heuristic* is not appropriate to detect if a plan is optimal or not. Because of this, the time of some replanning applications is higher than necessary. Some other heuristics or methods of misplaced sub-plan verification must be designed in the future.

The SQUIRE method does not guarantee to find an optimal solution of an specific problem because it only analyses the intermediary states and the final state that were provided by the solution plan. The method is restricted to reduce the plan length between the initial and the final state of the original solution plan. For instance, if an optimal solution of an specific problem is to consider another final state than that provided by the given solution plan, the SQUIRE method is unable to find this optimal solution.

This limitation of the SQUIRE method is responsible for the difficulty that this method presented in DriverLog domain. This limitation will be analyzed in the future.

This version of SQUIRE method is designed to improve the solution quality by decreasing the number of steps of the solution plan. However, this method is not restricted to the solution length, but it can also be extended to deal with complex domains with resources and time.

To work in other domains with numerical variables, a heuristic function that estimates the cost of a state by taking in consideration these numerical values must be available. The metric-FF planner system (Hoffmann, 2002) provides a Metric-heuristic that can be used to extend the SQUIRE method to numerical domains. However, it will be left for future studies.

7. CONCLUSION

The SQUIRE method, presented in this paper, is the first effective domain-independent method of plan quality improvement with anytime behavior. It uses the *FF-heuristic*

and a modified version of the FF planning system to determine and find alternative plans for some misplaced sub-plan. These misplaced sub-plans are bad segments of the entire solution.

The SQUIRE method improves the quality of many solutions plans in a reasonable amount of time. For case-based planners, the SQUIRE method is even more effective, because the case-based planners' solutions are usually longer than those from the generative systems.

The SQUIRE method in this paper focuses on the plan length and does not consider time or resources. The method will be extended in the future to support domains with these features.

REFERENCES

- Ambite, J. L.; Knoblock C. A. 2001. Planning by Rewriting. In: *Journal of Artificial Intelligence Research*, 15, pp 207-261.
- Ambite, J. L.; Knoblock, C. A.; Minton S. 2000. Learning plan Rewriting Rules. In: *Proceedings of AIPS '00*. AAAI Press. p.3-12.
- Bacchus, F. AIPS-2000 Planning Competition Results. Available in: <http://www.cs.toronto.edu/aips2000/>.
- Blum, A.; Furst M. 1997. Fast Planning through Planning Graphs Analysis, *Artificial Intelligence*, 90, p. 281-300.
- Bonet, B; Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*. 129: 5-33.
- Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69 (1-2), 165-204.
- Gerevini A.; Serina, I. 2000. Fast Adaptation through Planning Graphs: Local and Systematic Search Techniques. In: *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling AIPS '00*. AAAI Press. p.112-121.
- Gerevini A.; Serina, I. 2002. LPG: A Planner Based on Local Search for Planning Graphs with Actions Costs. In: *Preprints of the 6th International Conference on Artificial Intelligence Planning and Scheduling AIPS '02*. AAAI Press. p.281-290.
- Hoffmann J. 2002. Extending FF to Numerical State Variables, in: *Proceedings of the 15th European Conference on Artificial Intelligence*, Lyon, France.
- Hoffmann, J.; Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*. 14: 253 – 302.
- Koenig,S. , Furcy, D., Bauer, C. 2002. Heuristic Search-Based Replanning. In: *Preprints of 6th International Conference on Artificial Intelligence on Planning and Scheduling (AIPS-2002)*. Toulouse. p.205-212.
- Long, D.; Fox, M. 2002. The 3rd International Planning Competition - IPC '2002. Available in <http://www.dur.ac.uk/d.p.long/competition.html>.
- Nebel,B. ; Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, n. 76, p.427-454.,. Special Issue on Planning and Scheduling.
- Tonidandel, F.; Rillo, M. 2001 . An Accurate Adaptation-Guided Similarity Metric for Case-Based Planning In: *Aha, D., Watson, I. (Eds.) Proceedings of 4th International Conference on Case-Based Reasoning (ICCBR-2001)*. Lecture Notes in Artificial Intelligence. vol 2080. 531-545. Springer-Verlag.
- Tonidandel, F.; Rillo, M. 2002. The FAR-OFF system: A Heuristic Search Case-Based Planning. In: *Proceedings of 6th International Conference on Artificial Intelligence on Planning and Scheduling (AIPS-2002)*. Toulouse. p.279-288.