

S-MEP: A Planner for Numeric Goals

Javier Sanchez(*) & Amol Dattatraya Mali

Dept. of Elect. Engg. & Computer Science,

University of Wisconsin, Milwaukee, WI 53211, USA

Phone: 1-414-229-6762, Fax: 1-414-229-2769, mali@uwm.edu

(*) EDESA S.A, Cra 18 86A-14, Bogota, Colombia, javier.sanchez@edesa.net

Abstract

Planning for numeric goals is an important problem. Only one of the many participants from the 2002 international planning competition (Metric-FF) can effectively handle problems containing numeric goal expressions. We report on planner S-MEP (Sequential More Expressive Planner) in this paper. S-MEP handles non-linear as well as linear expressions in preconditions, effects and goal. Metric-FF handles only linear expressions. Metric-FF ignores decrease effects of actions while computing the heuristic information. S-MEP considers decrease effects of actions while computing heuristic information. We report on empirical evaluation of S-MEP and its comparison with two versions of Metric-FF on 400 problems from linear Jugs domain, 120 problems from linear Short-Move-Karel domain and 120 problems from linear Long-Move-Karel domain. Karel domains are robotic transportation domains and Jugs domain is a fluid transfer domain. S-MEP solves many problems that neither version of Metric-FF can solve.

1 Introduction

Planning in more expressive domains has received an increased attention in the last four years. These domains contain actions with fixed/variable durations and/or quantified preconditions and/or quantified effects and/or numerical variables in the preconditions and/or effects and/or conditional effects. Several planners solved problems from such domains at the international planning competition in 2002. They include Sapa [1], Metric-FF [5], TP4 [2], and LPG [3]. LPG, Sapa, and TP4 solve problems in domains containing resource quantities and durative actions. Metric-FF does not handle durative actions. Planning in more expressive domains is an important problem since most real-world domains contain durative actions and numerical resource quantities.

Though there are several planners which handle actions with numerical preconditions and numerical effects, there is only one participant from the international planning competition in 2002 which can efficiently handle numerical goals. This planner is Metric-FF. Numerical goals need to be achieved in many domains. Desired locations of robots and various objects they move in a workspace can be specified in the form of x and y coordinates. The goal of taking passengers to various floors through an elevator can be specified using equalities. The required quantities of various fluids in various tanks or jugs can be expressed as numerical equalities. The goal of finding a plan whose execution does not consume more resources than a given bound can be specified using numerical inequalities. In STRIPS planning, actions have only two types of effects (add and delete). In domains containing predicates as well as numerical variables, actions have five types of effects (add, delete, increase the value of a numeric variable, assign a value to a numeric variable, and decrease the value of a numeric variable).

In this paper we report on S-MEP. It finds plans in domains that do not contain durative actions. S-MEP finds sequential plans. We have built another planner P-MEP (Parallel MEP) which handles domains containing durative actions, quantifiers and numerical variables. P-MEP finds concurrent plans. S-MEP solves problems containing numerical goals. S-MEP handles numerical preconditions as well as numerical effects. S-MEP differs from Metric-FF in two important ways. Metric-FF handles only linear expressions in goal, preconditions and effects. S-MEP handles both linear as well as non-linear expressions in the preconditions, effects and goal. Actions can have three types of effects on numerical variables (increase, decrease and assign). Metric-FF ignores decrease effects of actions while computing heuristic information. S-MEP does not ignore decrease effects of actions while computing heuristic information. Metric-FF and S-MEP are both forward state-space planners. There are two versions of Metric-FF. One version does local search with enforced hill climbing. The other

version does best-first search. Both versions of Metric-FF find relaxed plans to compute the estimates of costs of cheapest paths to goal from various nodes. Both versions of Metric-FF thus use the same heuristic. Relaxed plans are plans obtained by ignoring certain constraints like delete effects and decrease effects. S-MEP allows a user to choose a heuristic from three heuristics: Cost, Sum of absolute differences and number of actions in the relaxed plan. Sum of absolute differences heuristic does not require construction of relaxed plans.

An empirical evaluation of S-MEP and two versions of Metric-FF on 640 problems from three linear numeric domains shows that considering decrease effects of actions while computing heuristic information significantly helps S-MEP. S-MEP solves many problems that no version of Metric-FF solves. S-MEP solves many problems faster than both versions of Metric-FF.

This work makes the following contributions.

- We report on a sound and efficient planner S-MEP which handles linear as well as non-linear expressions in numerical preconditions and effects and goal.
- Since predicates are boolean numeric variables, S-MEP also handles domains containing only predicates and domains containing both predicates and non-boolean numeric variables.
- S-MEP uses closed intervals to represent the values of numerical variables that can be achieved in a relaxed fashion. The intervals allow an easy detection of whether a numerical goal or numerical subgoal is achievable in a relaxed fashion.
- Closed intervals for numeric variables in S-MEP make it easy to use decrease effects of actions in computing heuristic information. The closed intervals also make it easy to check if linear/non-linear expressions are achievable (satisfiable) in a relaxed fashion.
- We introduce a novel heuristic for numerical goals, based on sum of absolute differences. This heuristic is highly suitable for certain numeric domains. This heuristic does not require construction of relaxed plans at all.
- We report on empirical evaluation and comparison of S-MEP with both versions of Metric-FF on 640 problems from three linear domains. Each problem has numerical goals. All three domains contain linear numerical preconditions and linear numerical conditional effects. The problem set includes 400 problems from Jugs domain, 120 problems from Short-Move-Karel domain and 120 problems from Long-Move-

Karel domain. There are many problems which S-MEP solves which no version of Metric-FF solves. S-MEP solves many problems many times faster than both versions of Metric-FF.

- We show how non-linear versions of the three domains can be created. Our evaluation confirms that S-MEP is the only planner which solves problems from domains containing non-linear expressions and numerical goals.

2 Background

In this section, we explain the notions of relaxed plans in STRIPS planning (FF) and in numeric planning (Metric-FF). We also explain enforced hill climbing, conditional effects, updated and referenced variables and equivalence of states in S-MEP in this section.

FF, a STRIPS planner [4] received exceptional performance award at the international planning competition in 2000. Metric-FF is a variant of FF which handles linear numerical expressions. FF finds relaxed plans to find the estimates of costs of cheapest paths to goal from various nodes. FF [4] is a forward state-space planner doing local search using enforced hill climbing. For a node n , FF finds relaxed plan from the relaxed planning graph at n . Relaxed planning graph (RPG) at n is found by applying actions in forward direction until the goal is true, assuming that actions lack delete effects. RPG consists of action levels and proposition levels which alternate. FF finds relaxed plan for node n by backward search on the RPG at node n . In FF, estimated cost of cheapest path to goal from node n (denoted by $h(n)$) is the number of actions in the relaxed plan at n . An example of relaxed planning graph and relaxed plan is shown in Fig. 1. A relaxed plan is a sequence of sets of actions. Relaxed plans are found in polynomial time. In STRIPS planning, if a relaxed plan does not exist at node n , then no plan can be obtained by expanding n . n is then a dead end. If no child of n has a lower $h()$ value than $h(n)$, FF does enforced hill climbing (EHC). EHC carries out breadth-first search until a descendent of n with lower $h()$ value than $h(n)$ is found. Once such a descendent n' is found, FF includes the path from n to n' in its partial plan, removing all other nodes generated by breadth-first search. FF maintains only one partial plan. If n has a child n'' such that $h(n'') < h(n)$, path from n to n'' is included in the partial plan and breadth-first search is not carried out. Metric-FF was inspired by the success of FF. Metric-FF constructs relaxed plans assuming that actions do not have delete effects and also do not have decrease effects. Metric-FF extracts relaxed plans by backward search on relaxed planning graphs (RPG). RPGs for numeric domains have action levels and variable levels which alternate. In the variable levels

in the relaxed planning graphs, Metric-FF stores maximum value of each variable. Maximum value of each variable v in variable level i is found by updating its value in variable level $(i - 1)$ by adding all increases to v by actions in action level $(i - 1)$. Construction of RPG terminates when maximum value of each variable is greater than or equal to the value required in goal. Let $max(v_1), max(v_2)$ be the maximum values of variables v_1, v_2 in the last variable level of the RPG at a node n . If the goal of the planning problem is $(v_1 = c) \wedge (v_2 = c')$ and $max(v_1) \geq c, max(v_2) \geq c'$, then Metric-FF terminates the construction of RPG at n . Metric-FF extracts relaxed plan at n by backward search on the RPG at n . Existence of a relaxed plan at a node does not guarantee existence of a plan from the node. This applies to FF, both versions of Metric-FF and also to S-MEP.

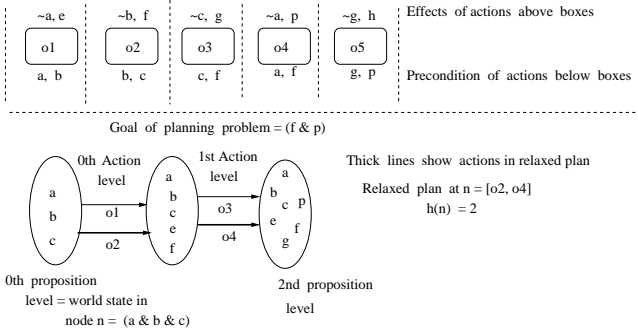


Figure 1. Relaxed planning graph and Relaxed plan

Conditional effects are context-dependent effects. The context-dependent effects are true only if certain conditions are satisfied. For example, the action of flying a plane P from Phoenix to LA has unconditional effects $at(P, LA), \neg at(P, Phoenix)$. When we mention effects, we mean unconditional effects. The flying action has conditional effects specified by the expression $\forall x (In(x, P) \rightarrow (at(x, LA) \wedge \neg at(x, Phoenix)))$. This expression states that all objects in the plane (if any) are also moved to LA. $at(x, LA)$ and $\neg at(x, Phoenix)$ are conditional effects because they are true only if $in(x, P)$ is true.

S-MEP classifies variables in each action's description into referenced variables and updated variables. Referenced variables are variables which appear in the description of an action but are not updated by the action. Referenced variables of an action a are included in set $R(a)$. Variables updated by an action a are included in set $U(a)$. R is the union of $R()$ sets of individual actions. U is the union of $U()$ sets of all actions. So R is the set of variables referenced by one or more actions and U is the set of variables updated by one or more actions. A world state in S-MEP is a set of variable-value pairs. Two states s and s' are equivalent if

P #	S-MEP	MFF (EHC)	MFF (Befs)
1	6.48	-	-
5	15.25	-	-
12	93.63	-	-
15	3.89	-	-
19	0.4	1.78	0.16
20	0.45	1.93	0.13
21	0.74	0.36	0.57
23	-	1.76	763.32
38	19.61	84.94	25
45	46.19	-	41.86
107	60.9	-	290.24
123	7.67	52.15	1.18
141	36.96	-	0.15
148	-	0.35	0.15
156	4.37	61.3	11.72
189	7.66	17.83	0.13
200	4.36	0.86	4.59

Table 1. Solving times on 17 of the first 200 problems from Jugs domain.

P #	S-MEP	MFF (EHC)	MFF (Befs)
201	57.87	-	347.22
205	109.64	-	-
210	108.64	-	830.39
215	-	-	1.29
230	63.73	-	0.15
250	23.35	-	3.21
261	5.3	-	7.71
263	21.69	-	3.17
278	22.02	-	0.16
297	-	19.51	24.36
313	12.64	-	-
339	26.59	-	-
347	5.14	-	-
364	6.42	-	-
368	1.91	20.82	-
376	85.63	-	-
392	37.15	-	-
400	6	-	-

Table 2. Solving times on 18 of the last 200 problems from Jugs domain.

the values of all variables in $R \cap U$ are same in s and s' . This definition of equivalent states allows S-MEP to control search. Consider the variable total-fuel-consumed denoting the total fuel consumed by a partial plan, in transportation logistics domain. This variable is not relevant to achieving any precondition of any action. This variable belongs to U but not to R . This variable can have infinite non-negative values since infinite flights are possible. By not considering such variables in state equivalence test, S-MEP controls the size of its search tree.

3 S-MEP

In this section, we first describe the search algorithm in S-MEP, followed by its three heuristics. Then we explain the interval representation in S-MEP and the relaxed plan extraction algorithm in S-MEP.

S-MEP conducts weighted A* style search in the space of world states. It uses relevance analysis before beginning search, to identify relevant actions. Only relevant actions are used in its search. S-MEP avoids expansion of a node containing state s if s is equivalent to a state in an already expanded node. This allows S-MEP to avoid useless looping. S-MEP expands node with lowest value of $f()$ first, where $f(n) = (1 - w).g(n) + w.h(n)$, where $0 \leq w \leq 1$. $g(n)$ is the cost of path from root node to node n . If the heuristic h is admissible and $w \leq 0.5$, S-MEP is guaranteed to find the optimal solution. The use of weight w allows a faster search with a bounded loss of optimality.

S-MEP allows a user to choose one of the following three heuristics: (i) Cost, (ii) Number of actions in relaxed plan and (iii) sum of absolute differences. The first two heuristics require S-MEP to find relaxed plans. If the objective function in the problem is minimization of total fuel consumption and Cost heuristic is used, then $h(n)$ is same as the total fuel consumption by actions in the relaxed plan at n . If the second heuristic is chosen, $h(n)$ is equal to the number of actions in the relaxed plan at n . If the third heuristic is used, $h(n)$ is equal to sum of absolute values of the differences between the values of variables in the state in n and their values in the goal. For example, let the goal be $(v_1 = c_1) \wedge (v_2 = c_2) \wedge (v_3 = c_3) \wedge (v_4 = c_4)$, where c_1, c_2, c_3 and c_4 are constants. Let the values of v_1, v_2, v_3, v_4 in the state in node n be v_{1n}, v_{2n}, v_{3n} and v_{4n} respectively. If the third heuristic is used, $h(n)$ is equal to $\sum_{i=1}^4 |v_{in} - c_i|$.

Instead of storing just the maximum value of each variable in the variable levels in RPGs, S-MEP stores an interval for each variable in each variable level. The interval contains maximum and minimum values. The size of an interval is monotonically increasing. For example, let the value of variable v_1 in initial state be 4. The initial interval for v_1 is [4,4]. If an action increasing v_1 by 10 is included

in the RPG, the interval for v_1 is changed to [4,14]. If an action decreasing v_1 by 20 is then included in the next action level of RPG, the interval for v_1 is changed to [-16,14]. If an action assigning 5 to v_1 is then included in the RPG, the interval for v_1 does not change, since 5 is in the existing interval. If two actions respectively assigning 40 and 80 to v_1 are then included in the next action level of RPG, interval of v_1 is changed to [-16, 80]. The intervals for variables make it easy to compute intervals for expressions. For example, let the intervals for v_1 and v_2 be $[min(v_1), max(v_1)]$ and $[min(v_2), max(v_2)]$ respectively. The interval for $v_1 + v_2$ is then $[min(min(v_1), min(v_2), (min(v_1) + min(v_2))), max(max(v_1), max(v_2), (max(v_1) + max(v_2)))]$. So if intervals of v_1, v_2 are [-3,5] and [-4,8], the interval of $(v_1 + v_2)$ is [-7, 13]. The intervals for $v_1.v_2, \frac{v_1}{v_2}, v_1 - v_2$ and polynomials with higher degree are found in a similar fashion. The intervals for variables and expressions can be considered as the intervals of relaxed values. This is because though S-MEP considers delete effects and decrease effects in computing intervals, it ignores the interactions between actions. So some of the values in the intervals may be impossible to achieve. Intervals of relaxed values for expressions containing division and/or multiplication, are not guaranteed to have minimum and maximum possible values. Computing minimum and maximum possible values of such expressions is time consuming and such a computation is not necessary to preserve completeness. The intervals of relaxed values affect the number of actions in relaxed planning graph and the existence of relaxed plans. If a relaxed plan exists at a node n , but no relaxed plan is found by S-MEP, $h(n)$ is assigned ∞ . n will be expanded after all states with finite $h()$ values are expanded, if solution is not found. Intervals of expressions are not stored in the RPGs. These intervals are found only to test if preconditions or goal are achieved in a relaxed fashion. Intervals for expressions make it easy to check if actions are applicable in RPG and if goal is true in RPG. For example, the precondition or subgoal $(v_1 + v_2) = 50$ is true in RPG if 50 lies in the interval of $v_1 + v_2$. Similarly, the expression/subgoal $v_1 < v_2$ is satisfied in a variable level in the RPG if $min(v_1) < max(v_2)$ is satisfied in the variable level.

S-MEP constructs RPG for a node by applying actions in forward direction and by computing action and variable levels, until the intervals of variables satisfy all expressions in the goal or no new action can be applied in the RPG, whichever occurs earlier. S-MEP includes an action in an RPG at the most once. Because of this it does not find relaxed plans for some nodes even if they exist. S-MEP does not treat these nodes as dead ends. It sets $h()$ values of these nodes to infinity and keeps them in the priority queue. Since S-MEP uses equivalent states-based search control, nodes with infinite $h()$ value are guaranteed to be expanded

if the state space is finite.

S-MEP finds relevant actions using relaxed planning graph for the root node. S-MEP constructs relaxed planning graph for root node by applying actions in forward direction. S-MEP generally finds far fewer relevant actions than Metric-FF. The fewer the relevant actions, the lower is the size of the search tree. Empirical results on reduction in the number of actions considered by S-MEP and Metric-FF in building the search tree are reported in the longer version of this paper [6]. We also explain the computation of intervals of relaxed values of non-linear expressions in [6]. Note that S-MEP can find plans requiring multiple occurrences of an action.

Consider a problem from Jugs domain where all jugs are full in the initial state and need to be all empty in the goal. On this problem, Metric-FF will infer that the goal is true in a relaxed fashion in the initial state itself. This is not the case in S-MEP.

S-MEP's algorithm for extracting relaxed plan from an RPG at node n is explained below. k is the number of action levels in the RPG. VG denotes the set of variables in the expressions in the goal G . $RP(i)$ is the set of actions at i th step in the relaxed plan. $RPG(i)$ is the set of actions in i th action level in the RPG. $RV(S)$ is the set of variables referenced by one or more actions from the action set S . $UV(S)$ is the set of variables updated by one or more actions from action set S .

Extract_Relaxed_Plan(RPG, G)

1. $i = k, M = VG, RP(j) = \phi, 1 \leq j \leq k$.
2. For each $a \in RPG(i)$ { If $(U(a) \cap M) \neq \phi$, do $RP(i) = (RP(i) \cup a)$ }
3. $i = i - 1$. If $i < 1$, return the relaxed plan.
4. $M = (M - UV(RP(i + 1))) \cup RV(RP(i + 1))$. Go to 2.

In brief, this algorithm includes all actions that update relevant variables at each step of the relaxed plan, starting from the goal.

4 Empirical Evaluation

We used three linear numerical domains in our empirical evaluation. Solving problems from Jugs domain involves achieving required quantities of a fluid in various jugs by filling, emptying the jugs or transferring fluid into another jug. This domain contains three operators. Short-Move-Karel and Long-Move-Karel are robotic transportation domains. Solving problems in these domains involves picking beepers, depositing beepers, moving the robot and turning the robot. Short-Move-Karel contains five operators. Long-Move-Karel contains six operators including all operators from Short-Move-Karel. Very few problems containing numerical goals are publicly available for testing planners. None of the problems used in the planning compe-

tion in 2002 have numerical goals. We do not consider numerical objective functions in problems as challenging numerical goals. Jugs is a pre-competition domain made available by the organizers. Short-Move-Karel and Long-Move-Karel are domains created by us.

S-MEP and both versions of Metric-FF stop search immediately after a plan is found. These planners were run on a Sun Ultra 10 machine running Solaris 9. We found solving times (cpu seconds), number of actions in the plans found and the total number of nodes expanded by the planners. A planner did not continue to attempt a problem after 188 MB RAM was consumed or 20 cpu minutes were used, whichever happened earlier. When S-MEP did not solve problems, it was generally because of the consumption of RAM. When any version of Metric-FF did not solve a problem, it was generally because the cpu time allowed had passed. S-MEP used $w = 0.85$. The version of Metric-FF which does best-first search (Befs) expands node with lowest value of $w_1.g(n) + w_2.h(n)$ first, where w_1 and w_2 are non-negative integers. The cost of solution found by weighted A* using $f(n) = (1 - w).g(n) + w.h(n)$ is same as the cost of solution found by weighted A* using $f(n) = g(n) + \frac{w}{1-w}.h(n)$. So we used $w_1 = 3$ and $w_2 = 17$ when running Metric-FF with best-first search. Since $\frac{w}{1-w}$ and $\frac{w_2}{w_1}$ are approximately equal for the values of w, w_1 , and w_2 we used, the comparison between S-MEP and the Befs version of Metric-FF was fair.

The empirical results have been reported in tables 1,2,3,4,5,6,7,8,9, and 10. Solving times have been reported in tables 1,2,3 and 5. Speedup obtained with S-MEP over the two versions of Metric-FF has been reported in tables 4 and 6.

P# in tables 1,2,3,4,5 and 6 denotes problem number. MFF in tables 1,2,3,4,5,6,7, and 8 denotes Metric-FF. EHC in these tables denotes Enforced Hill Climbing. Befs in these 8 tables denotes best-first search. - in tables 1,2,3 and 5 denotes "not solved in the time limit". - in tables 4 and 6 denotes that no speedup was obtained with S-MEP. Entries in the form x,y in table 7 show the minimum and maximum number of actions in the plans found by a planner in a domain. For example, the plans found by S-MEP in jugs domain had at least 5 actions and at the most 24 actions. Also, the plans found by Metric-FF (EHC) in Short-Move-Karel domain had at least 1 action and at the most 23 actions. Entries of the form "x / y" in table 8 show the number of problems solved and attempted by a planner in a domain. For example, S-MEP solved 142 out of the 400 problems it attempted in jugs domain. Metric-FF (EHC) solved 62 out of the 120 problems it attempted in Short-Move-Karel domain.

In table 9, SNE denotes the number of problems solved by S-MEP which Metric-FF (EHC) did not solve, in a specific domain. In table 9, SNB denotes the number of prob-

lems solved by S-MEP which Metric-FF (Befs) did not solve, in a specific domain. In table 9, SNEB denotes the number of problems solved by S-MEP which neither version of Metric-FF solved, in a specific domain. For example, 49 of the problems that S-MEP solved in Short-Move-Karel domain were not solved by Metric-FF (EHC). 43 of the problems solved by S-MEP in jugs domain were not solved by either version of Metric-FF.

FE in table 10 denotes the number of problems that S-MEP solved faster than Metric-FF (EHC) in a specific domain. FB in table 10 denotes the number of problems that S-MEP solved faster than Metric-FF (Befs) in a specific domain. FEB in table 10 denotes the number of problems that S-MEP solved faster than both versions of Metric-FF in a specific domain. For example, S-MEP solved 92 problems faster than Metric-FF (EHC) in jugs domain. S-MEP solved 78 problems faster than both versions of Metric-FF in Long-Move-Karel domain.

In the remainder of this section, we report on empirical results on Jugs domain in section 4.1 and the empirical results on the Karel domains in section 4.2.

4.1 Jugs Domain

A partial description of this domain is given below. This description contains everything except the definition of the operator that empties a jug. The domain description is followed by the description of a sample problem from this domain. The sample problem is followed by empirical results. The *pour* operator has numerical conditional effects. As the number of jugs and the difference between the maximum capacity and minimum capacity increase, the size of the state space increases fast. For example, if there are 5 jugs with capacities 1,10,20,21 and 22 respectively, the number of possible states is (2).(11).(21).(22).(23).

```
(define (domain Jugs)
  (:requirements :typing :fluents :conditional-effects)
  (:types jug nonjug)
  (:functions (capacity ?j - jug)
              (contents ?j - jug))
  (:action fill
   :parameters (?j - jug)
   :precondition (< (contents ?j) (capacity ?j))
   :effect (and (assign (contents ?j) (capacity ?j))))
  (:action pour
   :parameters (?j1 ?j2 - jug)
   :precondition (> (contents ?j1) 0)
   :effect (and (when (<= (+ (contents ?j1) (contents ?j2)) (capacity ?j2))
```

```
(and (assign (contents ?j1) 0)
      (increase (contents ?j2) (contents ?j1))))
      (when (> (+ (contents ?j1) (contents ?j2)) (capacity ?j2))
            (and (decrease (contents ?j1) (- (capacity ?j2) (contents ?j2)))
                  (assign (contents ?j2) (capacity ?j2))))))
  (define (problem jugs2)
    (:domain Jugs)
    (:objects jug1 jug2 jug3 jug4 jug5 - jug)
    (:init (= (capacity jug1) 1)
           (= (capacity jug2) 5)
           (= (capacity jug3) 10)
           (= (capacity jug4) 25)
           (= (capacity jug5) 67)
           (= (contents jug1) 0)
           (= (contents jug2) 0)
           (= (contents jug3) 0)
           (= (contents jug4) 0)
           (= (contents jug5) 0))
    (:goal (and (= (contents jug3) 8)
                (= (contents jug5) 41)
                )))
```

S-MEP used second heuristic for this domain, as per which the $h(n)$ value equals the number of actions in the relaxed plan at n . The empirical results are shown in tables 1,2,7,8,9 and 10. The problems were generated by varying the following parameters: (i) number of jugs in initial state (from 5 to 13), (ii) number of jugs in goal (from 3 to 7), (iii) capacities of jugs (from 1 to 1200), (iv) contents of jugs in initial state (from 0 to their capacities), and (v) contents of the jugs in the goal (from 0 to their capacities). Tables 1 and 2 together show solving times of the planners on 35 of the 400 problems. These results show that S-MEP solved several problems that no version of Metric-FF solved. The results also show that S-MEP solved several problems faster than both versions of Metric-FF. There are no dead ends in the Jugs domain, since every possible state is achievable from every other state. Despite this, there are several problems solved by S-MEP that no version of Metric-FF solved. A summary based on the results in this domain is shown in tables 7,8,9 and 10. Since neither S-MEP nor any version of Metric-FF is optimal, they all may find plans containing many more actions than optimal plans. Still the number of actions in the plans found can serve as an indicator of the ability of a planner to solve problems requiring longer planning solutions. Table 7 and our manual inspection of some

optimal plans and plans found shows that S-MEP is capable of solving problems requiring longer plans.

4.2 Karel Domains

In this section, we first describe the Short-Move-Karel domain. Then we describe the Long-Move-Karel domain, followed by the empirical results. Karel is name of a robot. The description of Short-Move-Karel domain contains everything except the description of the turn-left operator. The grid has square shape in both Karel domains. In both Karel domains, the move operator has numerical preconditions in disjunctive normal form and it has numerical conditional effects. If the size of the grid is n , there are $(n + 1)^2$ points on the grid. So in the presence of a robot and m beepers, the size of the state space in both Karel domains is at least $(n + 1)^{2 \cdot (m+1)}$. The comments in the domain and problem description start with ;;.

```
(define (domain Short-Move-Karel)
  (:requirements :typing :fluents)
  (:types beeper robot dir - object)
  (:constants north south east west -
  dir)
  (:predicates (in-bag ?r - robot ?b -
  beeper)) ;; Is b in r's bag ?
  (:functions
  (at-x ?o - object) ;; X Coordi-
  nate for either beeper or robot
  (at-y ?o - object) ;; Y Coordi-
  nate for either beeper or robot
  (direction ?d -
  dir) ;; 0=East,1=South,2=West,3=North
  (facing ?r - robot) ;; a direction
  (size)) ;; size of the grid
  (:action turn-
  right ;; Turn a robot to the right
  :parameters (?r - robot)
  :effect (and (when (= (facing ?r) (di-
  rection north)) (assign (fac-
  ing ?r) (direction east)))
  (when (= (facing ?r) (direc-
  tion west)) (assign (facing ?r) (di-
  rection north)))
  (when (= (facing ?r) (direc-
  tion south)) (assign (facing ?r) (di-
  rection west)))
  (when (= (facing ?r) (direc-
  tion east)) (assign (facing ?r) (di-
  rection south))))))
  (:action pick-
  beeper ;; Pick up a beeper from robot's
```

```
;; current posi-
tion and store it in robot's bag
:parameters (?r - robot ?b - beeper)
:precondition (and
  (= (at-x ?r) (at-x ?b))
  (= (at-y ?r) (at-y ?b))
  (forall (?t - robot) (not (in-
  bag ?t ?b))))
:effect (and (in-bag ?r ?b)
  (assign (at-x ?b) 0)
  (assign (at-y ?b) 0)))
(:action put-
  beeper ;; Put a beeper on the grid.
:parameters (?r - robot ?b - beeper)
:precondition (in-bag ?r ?b)
:effect (and (assign (at-x ?b) (at-
  x ?r))
  (assign (at-y ?b) (at-y ?r))
  (not (in-bag ?r ?b))))
(:action move)
:parameters (?r - robot)
:precondition (or (and (= (fac-
  ing ?r) (direction east)) (< (at-
  x ?r) (size))) ;; Maximum x
  (and (= (fac-
  ing ?r) (direction south)) (> (at-
  y ?r) 0)) ;; Minimum y
  (and (= (fac-
  ing ?r) (direction west)) (> (at-
  x ?r) 0)) ;; Minimum x
  (and (= (facing ?r) (direc-
  tion north)) (< (at-
  y ?r) (size)))) ;; Maximum y
:effect (and (when (= (facing ?r) (di-
  rection east)) (increase (at-x ?r) 1))
  (when (= (facing ?r) (direc-
  tion south)) (decrease (at-y ?r) 1))
  (when (= (facing ?r) (direc-
  tion west)) (decrease (at-x ?r) 1))
  (when (= (facing ?r) (direc-
  tion north)) (increase (at-y ?r) 1))))))
```

```
(define (problem Short-Move-Karel10) (:domain Karel)
  (:objects robot1 - robot b1 - beeper) (:init (= (direction east)
  0) (= (direction south) 1) (= (direction west) 2) (= (direction
  north) 3) (= (size) 500) ;; Grid is size x size, (= (at-x robot1)
  5) ;; (5,5) is the robot's initial position, (= (at-y robot1) 5)
  (in-bag robot1 b1) ;; All beepers in the robot's bag, (= (fac-
  ing robot1) 0) ;; Start looking at East, (= (at-x b1) 0) (= (at-y
  b1) 0) (:goal (and (not (in-bag robot1 b1)) (= (at-x b1) 3) (=
  (at-y b1) 3) (= (at-x robot1) 5) (= (at-y robot1) 5))))))
```

Long-Move-Karel is Short-Move-Karel augmented with *long_move* operator. This moves the robot in the direction it is facing by 10 units, if the move does not cross/touch

the boundaries of the grid. Because of longer moves, this operator can allow the planners to find shorter plans as well as shorter relaxed plans. The definition of this operator is very similar to that of move operator.

```
(:action long_move
:parameters (?r - robot)
:precondition (or (and (= (facing ?r) (direction east)) (< (+ (at-x ?r) 10) (size)))) ;; Maximum x
                (and (= (facing ?r) (direction south)) (> (- (at-y ?r) 10) 0))
                (and (= (facing ?r) (direction west)) (> (- (at-x ?r) 10) 0)) ;; Minimum x
                (and (= (facing ?r) (direction north)) (< (+ (at-y ?r) 10) (size)))) ;; Maximum y
:effect (and (when (= (facing ?r) (direction east)) (increase (at-x ?r) 10))
              (when (= (facing ?r) (direction south)) (decrease (at-y ?r) 10))
              (when (= (facing ?r) (direction west)) (decrease (at-x ?r) 10))
              (when (= (facing ?r) (direction north)) (increase (at-y ?r) 10))))))
```

The problems in both Karel domains were generated by varying the following parameters: (i) number of beepers in initial state, (ii) number of beepers in goal, (iii) coordinates of beepers in initial state, (iii) coordinates of beepers in goal, (iv) number of beepers in robot’s bag in initial state, (v) size of grid, (vi) number of beepers in robot’s bag in goal, (vii) coordinates of the robot in initial state, (viii) coordinates of the robot in goal, (ix) presence of robot in goal, (x) direction faced by the robot in initial state, and (xi) direction faced by the robot in goal.

S-MEP used third heuristic for Karel domains (sum of absolute differences heuristic). Empirical results on Short-Move-Karel are shown in tables 3,4,7,8,9 and 10. Table 3 shows solving times on 17 out of 120 problems.

Table 4 shows the speedup obtained with S-MEP over each version of Metric-FF on 17 out of the 120 problems. A summary based on the results on problems in Short-Move-Karel is shown in tables 7,8,9 and 10. S-MEP turned out to be best on this domain as well as its Long-Move version due to at least two reasons. The first reason is that sum of absolute differences heuristic is more suitable for Karel domains than the number of actions in relaxed plans. The second reason is that the difference-based heuristic can be computed much faster than the relaxed plan-based heuristic.

P #	S-MEP	MFF (EHC)	MFF (Befs)
1	0.9	1.9	-
5	1.03	-	-
11	0.79	-	-
17	0.03	0.25	-
30	6.85	0.15	-
43	0.7	-	-
50	-	140.52	-
64	0.71	-	-
74	0.78	-	-
80	0.02	0.02	0.02
82	0.91	87.41	-
85	0.63	100.58	-
88	4.59	0.02	-
90	1.04	17.58	-
98	-	0.04	-
105	1.42	44.78	-
109	0.71	61.44	-

Table 3. Solving times on 17 of the 120 problems from Short-Move-Karel domain.

P #	Over MFF (EHC)	Over MFF (Befs)
3	> 1690	> 1690
6	> 1153	> 1153
14	740.3	> 5217
15	818	> 6000
17	8.33	> 40,000
19	> 1714	> 1714
30	-	> 175
31	> 3243	> 3243
43	> 1714	> 1714
51	> 1538	> 1538
53	> 1463	> 1463
55	> 1714	> 1714
62	> 1690	> 1690
71	> 2448	> 2448
79	> 628	> 628
95	71.8	> 952
119	85.1	> 1690

Table 4. Speedup with S-MEP on 17 of the 120 problems from Short-Move-Karel domain.

Empirical results on Long-Move-Karel are shown in tables 5,6,8,9 and 10. Table 5 shows solving times on 12 out of 120 problems. Table 6 shows the speedup obtained with S-MEP over each version of Metric-FF on 15 out of the 120 problems. A summary based on the results on problems from Long-Move-Karel is shown in tables 8,9 and 10.

Minimum and maximum number of actions in plans found by S-MEP and the two versions of Metric-FF in jugs and Short-Move-Karel domains are shown in table 7. The number of problems solved and attempted by each planner in each domain are reported in table 8. S-MEP solved many more problems than both versions of Metric-FF in both Karel domains.

P #	S-MEP	MFF (EHC)	MFF (Befs)
1	1.03	-	-
11	0.82	-	-
20	0.8	-	-
29	3.22	62.29	-
41	0.22	-	-
49	-	5.68	-
56	0.78	-	-
64	0.8	-	-
75	1.68	0.03	0.09
86	0.24	66.18	3.31
97	5.72	-	-
118	0.87	-	-

Table 5. Solving times on 12 out of 120 problems from Long-Move-Karel domain.

Table 9 shows that S-MEP solved 160 problems which neither version of Metric-FF did. These include 43 problems from jugs domain, 49 problems from Short-Move-Karel domain and 68 problems from Long-Move-Karel domain. S-MEP solved 195 problems that Metric-FF (EHC) did not solve. S-MEP solved 196 problems that Metric-FF (Befs) did not solve. Table 10 shows that S-MEP solved 248 problems faster than Metric-FF (EHC). S-MEP solved 226 problems faster than Metric-FF (Befs). S-MEP solved 208 problems faster than both versions of Metric-FF.

To test the ability of S-MEP to handle non-linear expressions, we created two additional domains NL-Jugs and NL-Karel, where NL denotes non-linear. In NL-Jugs, each jug has a radius. Also, if fluid is poured from a jug J_2 with radius r_2 into jug J_1 with radius r_1 , such that $r_2 > r_1$, there is spilling. In this case, the maximum amount of fluid transferred to J_1 is just $j_2 \cdot (\frac{r_1}{r_2})^2$, where j_2 is the content of jug J_2 . In NL-Karel, the distance that a robot will cover using move operator is proportional to the product of its x and y coordinates. This is because there is wind in this domain which exerts force on the robot. Also, some of the

P #	Over MFF (EHC)	Over MFF (Befs)
8	> 1008	> 1008
15	> 5000	> 5000
17	28.66	28,601
20	> 1500	> 1500
27	> 3636	> 3636
34	> 1500	> 1500
51	> 1318	> 1318
56	> 1538	> 1538
62	> 1500	> 1500
63	> 1481	> 1481
71	> 2000	> 2000
75	-	-
80	199	120
93	3.99	-
118	> 1379	> 1379

Table 6. Speedup with S-MEP on 15 of the 120 problems from Long-Move-Karel domain.

Domain	S-MEP	MFF (EHC)	MFF (Befs)
Jugs	5, 24	4, 17	4, 16
Short-Move-Karel	1, 27	1, 23	1, 12

Table 7. Minimum and maximum number of actions in plans found by the 3 planners on problems in 2 domains.

Domain	S-MEP	MFF (EHC)	MFF (Befs)
Jugs	142 / 400	121 / 400	178 / 400
Short-Move-Karel	90 / 120	62 / 120	7 / 120
Long-Move-Karel	90 / 120	27 / 90	26 / 90

Table 8. Number of problems solved and attempted by the 3 planners on problems in 3 domains.

subgoals are of the forms $((x - a)^2 + (y - b)^2) < c^2$ and $((x - a)^2 + (y - b)^2) > c^2$ where c is a constant. These express that the final position of the robot or a beeper is located inside or outside a circle of radius c , centered at (a, b) . We created 5 NL-karel problems and 5 NL-Jugs problems. Both versions of Metric-FF terminated on these problems without solutions, reporting that they had non-linear expressions. S-MEP solved two out of the 10 problems.

Domain	SNE	SNB	SNEB
Jugs	77	44	43
Short-Move-Karel	49	83	49
Long-Move-Karel	69	69	68

Table 9. Number of problems from the 3 domains, solved by S-MEP and not solved by Metric-FF (EHC) and/or Metric-FF (Befs)

The work on S-MEP can be extended in several directions. One direction is to make the relaxed plans more informative by considering interactions between actions, e.g. mutual exclusion relations. Another direction is to compute the intervals of relaxed values more accurately. Maintaining just minimum and maximum values of a variable can allow application of many irrelevant actions in the relaxed planning graphs. Alternative representations are lists of values and sets of small intervals. We intend to pursue these directions.

Domain	FE	FB	FEB
Jugs	92	62	56
Short-Move-Karel	74	86	74
Long-Move-Karel	82	78	78

Table 10. Number of problems from the 3 domains, solved by S-MEP faster than Metric-FF (EHC) and/or Metric-FF (Befs)

5 Conclusion

Planning in real world involves numerical variables and numerical goals. Planning in numerical domains is challenging since the sizes of the state spaces are very high. In this paper we reported on a sound planner S-MEP which solves problems containing numerical variables and/or numerical goal. S-MEP handles non-linear as well as linear expressions in the preconditions and effects of actions and in the goal. No other planner does this. S-MEP uses interval representation to capture relaxed values of variables

in relaxed planning graphs. This makes it easy to take into account the decrease effects of actions and also handle non-linear expressions, in computing the heuristic information. Another contribution of our work is the sum of absolute differences heuristic. This is very effective in navigation and transportation domains like Karel domains. We evaluated S-MEP on 650 problems from 5 domains. These include 640 problems from 3 linear domains and 10 problems from 2 non-linear domains. We also ran both versions of Metric-FF on the 650 problems. The evaluation on the 650 problems shows that s-MEP solved 114 and 113 more problems than the versions of Metric-FF doing enforced hill climbing and best-first search respectively. S-MEP solved 160 problems that neither version of Metric-FF solved. S-MEP solved 208 problems faster than both versions of Metric-FF. This shows that the ideas of interval representations of relaxed values and driving search by differences between current values and required values are very useful in solving numerical planning problems. Our future work includes investigating variants and combinations of these two ideas.

Acknowledgement: This work has been funded by NSF grant IIS-0119630 (Pruning Techniques for More Expressive Planning) to Amol Dattatraya Mali. The authors thank Minh Tang for creating NL-Karel and NL-Jugs domains and testing the planners on these domains. The authors thank John Boyland and Susan McRoy for useful comments on S-MEP and P-MEP. This work was performed when the first author was at UW, Milwaukee for his graduate study.

- [1] Minh Binh Do and Subbarao Kambhampati, Sapa: A domain-independent metric temporal planner, *Proceedings of European Conference on Planning (ECP)*, 2001.
- [2] Patrik Haslum and Hector Geffner, Heuristic planning with time and resources, *Proceedings of European Conference on Planning (ECP)*, 2001.
- [3] Alfonso Gerevini, Ivan Serina, Alessandro Saetti, and Sergio Spinoni, Local search techniques for temporal planning in LPG, *Proceedings of International conference on automated planning and scheduling (ICAPS)*, Trento, Italy, 2003.
- [4] Jorg Hoffmann, FF: The fast forward planning system, *AI Magazine*, Volume 22, Number 3, Fall 2001, pp. 57-62.
- [5] Jorg Hoffmann, Extending FF to handle numerical state variables, *Proceedings of European Conference on Artificial Intelligence (ECAI)*, 2002.
- [6] Javier Sanchez and Amol Dattatraya Mali, Heuristic search planning for numerical goals, To be submitted to *Journal of Artificial Intelligence Research (JAIR)*.