Domain Independent Approaches for Finding Diverse Plans

Biplav Srivastava[†] **Tuan A. Nguyen**[‡] **Alfonso Gerevini**[¶]
Subbarao Kambhampati[*] **Minh Binh Do**[§] **Ivan Serina**[¶]

[†]IBM India Research Laboratory, New Delhi and Bangalore, India, sbiplav@in.ibm.com

[*]Arizona State University, Tempe, AZ, USA 85287, rao@asu.edu

[‡]University of Natural Sciences, Ho Chi Minh, Vietnam, natuan@fit.hcmuns.edu.vn

[§]Palo Alto Research Center, USA, minhdo@parc.com

[¶]University of Brescia, Italy, {gerevini,serina}@ing.unibs.it

Abstract

In many planning situations, a planner is required to return a diverse set of plans satisfying the same goals which will be used by the external systems collectively. We take a domain-independent approach to solving this problem. We propose different domain independent distance functions among plans that can provide meaningful insights about the diversity in the plan set. We then describe how two representative state-of-the-art domain independent planning approaches – one based on compilation to CSP, and the other based on heuristic local search – can be adapted to produce diverse plans. We present empirical evidence demonstrating the effectiveness of our approaches.

1 Introduction

A typical automated planner takes as input the specifications of the initial and goal states and the set of available actions, and finds a plan that will satisfy the goals by efficiently searching in the space of possible state configurations or action orderings (plans). In many planning situations, a planner is required to return not one but a set of diverse plans satisfying the same goals which will be used by the external systems collectively. As an example, in adaptive web services composition, the web service engine wants to have a set of diverse plans/ compositions such that if there is a failure while executing one composition, an alternative may be used which is less likely to be failing simultaneously[Chafle *et al.*, 2006]. However, if a user is helping in selecting the compositions, the planner could be first asked for a set of diverse plans and when she selects one of them, the planner is next asked to find plans that are similar to the selected one. Another example is using planning for intrusion detection [Boddy *et al.*, 2005], where the aim is to detect diverse ways of possible intrusion attacks (represented as plans).

Although the need for finding similar or different plans has been noticed in the past, there has been little concrete work on formalizing and solving the problem. What little there is has concentrated on finding similar plans[Fox *et al.*, 2006] or are domain-specific approaches (See Section 5). For example, Myers [Myers & Lee, 1999] expects the domain modelers to provide a "meta-theory" of the domain (in addition to the domain transition model in terms of actions and their effects). In this paper, we focus on domain-independent means of finding (and comparing) diverse plans. This immediately brings up the issue: *on what basis should two plans be compared?* The first contribution of this paper is the proposal of a spectrum of distance measures that capture plan characteristics in terms of actions, behaviors (states that result from the plan execution) and causal structures.

Once the distance measures are in place, we turn to the issue of automatically generating sets of plans that have the desired diversity in terms of those distance measures. A naive approach for this would be to let the planner generate multiple solutions and filter out the solutions that do not satisfy the required diversity. Such a filtering approach is not very promising, particularly given the fact that the set of plans for a given problem can in principle be infinite.[1] Indeed, Boddy et. al. 2005, who use this type of filtering technique in the intrusion detection domain, explicitly acknowledge the need for approaches that take diversity constraints into account during search more actively.

The second contribution of our paper is thus an investigation of effective approaches for using distance measures to bias a planner's search to find diverse plans efficiently. The technical details of biasing the search do depend on the details of the underlying planner. To get a broader understanding, we decided to investigate two representative state-of-the-art planning approaches. The first, GP-CSP, typifies the issues involved in generating diverse plans in bounded horizon compilation approaches, while the second, LPG, typifies the issues involved in modifying the heuristic search planners. Our investigations with GP-CSP allow us to compare the relative difficulties of enforcing diversity with each of the three distance measures. We find that using action-based distance measures to find diverse plans usually results in plans that are also diverse with respect to their behavior and causal structures, but less likely in other permutations. With LPG, we focus only on the action-based distance, which can be treated in a natural way by a relatively simple modification of the heuristic function, to explore scaleup issues. We find that the proposed distance measures make LPG more effective in solving for diverse plans over large problem instances.

[*]Kambhampati's research is supported in part by an IBM Faculty Award, the NSF grant IIS–308139, the ONR grant N000140610058, and by a Lockheed Martin subcontract TT0687680 to ASU as part of the DARPA Integrated Learning program.

[1]To see this, note that there may be infinitely many non-minimal variations of a single plan.

Basis	Pros	Cons
Actions	Does not require problem information	No problem information is used
States	Not dependent on any specific plan representation	Needs an execution simulator to identify states
Causal links	Considers causal proximity of state transitions (action) rather than positional (physical) proximity	Requires domain theory

Table 1: The pros and cons of different bases to characterize plans.

In the rest of the paper, we start by formalizing the problem and then propose different plan distance functions. Next, we propose methods to find diverse plans and demonstrate their effectiveness using GP-CSP and LPG. We end with a discussion on related work and our main results.

2 Distance Measures

To talk formally about generation of diverse plans, we need to start with the notion of distance between plans. Let $\delta(S_i, S_j) \rightarrow [0, 1]$ denote a distance function between a pair of plans. A value of 0 represents complete similarity of plans while 1 represents complete diversity. Following the convention of [Hebrard *et al.*, 2005], for a given set **S** of plans, we define *max(δ,* **S***)* = $\underset{S_i,S_j \in \mathbf{S}}{max}$ $\delta(S_i, S_j)$ and *min(δ,* **S***)* = $\underset{S_i,S_j \in \mathbf{S}}{min}$ $\delta(S_i, S_j)$. The problem of finding k diverse/ similar plans for a problem PP, whose set of all plans is represented by **Plan**(PP), is then stated below.

> *d*DISTANT*k*SET (resp. *d*CLOSE*k*SET): Find **S** with **S** $\subseteq$ **Plan**(PP), $|$ **S** $|$ = k and *min(δ,* **S***)* $\geq d$ (resp. *max(δ,* **S***)* $\leq d$).

At the heart of tackling this problem is the issue of defining criteria by which two plans are compared.[2] As mentioned earlier, we focus here on domain-independent measures for comparing plans. We can compare two plans in terms of:

1. Actions that are present in the two plans.
2. The behaviors resulting from the execution of the plans (where the behavior is captured in terms of the sequence of states the agent goes through).
3. The causal structures of the two plans measured in terms of the causal links representing how actions contribute to the goals being achieved.[3]

Table 1 gives the pros and cons of using the different comparison methods. We note that if actions in the plans are used as the basis for comparison, no additional problem or domain theory information is needed. If plan behaviors are used as the basis for comparison, the representation of the plans that bring about state transition becomes irrelevant since only the actual states that an execution of the plan will take is considered. Hence, we can now compare plans of different representations, e.g., 4 plans where the first is a deterministic plan, the second is a contingent plan, the third is a hierarchical plan and the fourth is a policy encoding probabilistic behavior. If causal links are used as the basis for comparison, the causal proximity among actions is now considered rather than just physical proximity in the plan.

[2]This issue is complicated by the fact that the plans being compared can be generated by an automated planner or found in other ways, e.g., given manually or manipulated after getting it from a planner[Srivastava, Vanhatalo, & Koehler, 2005]. Although we only consider generated plans, the distance measures apply to other provences as well. We make no apriori assumption about the plans like each goal has a single causal support structure.

[3]A causal link $A_1 \overset{p_1}{\rightarrow} A_2$ records that a predicate is produced at A_1 and consumed at A_2. A causal chain is a sequence of causal links of the form $A_1 \overset{p_1}{\rightarrow} A_2, A_2 \overset{p_2}{\rightarrow} A_3, ..., A_{n-1} \overset{p_{n-1}}{\rightarrow} A_n$. We use causal links for analysis but refer to a plan's causal links by causal chains, wherever possible, for convenience.

Name	Basis	Computation
δ_1 (δ_a)	Actions	Set-difference
δ_2	Actions	Prefixes Neighbourhood
δ_3 (δ_s)	States	Set-difference
δ_4	States	Prefixes Neighbourhood
δ_5 (δ_c)	Causal Links	Set-difference
δ_6	Causal Links	Prefixes Neighbourhood

Table 2: A spectrum of distance functions based on different bases and way of computations.

Aggregating Distances Once a basis for plan comparison is chosen, we still have different choices for aggregating the distances. For example, if we are interested in action based comparison of plans, then we could (i) view the plans as sets (or bags) of actions and consider set (bag) difference between the two plans, (ii) consider the plans as sequences of actions and consider measures such as "hamming distance" that are sensitive to the position of the actions. In Table 2, 6 distance functions are presented which use 3 different bases and 2 different ways of computation. We use δ_1, δ_3 and δ_5 and refer to them by δ_a, δ_s and δ_c, respectively, in the rest of the paper.

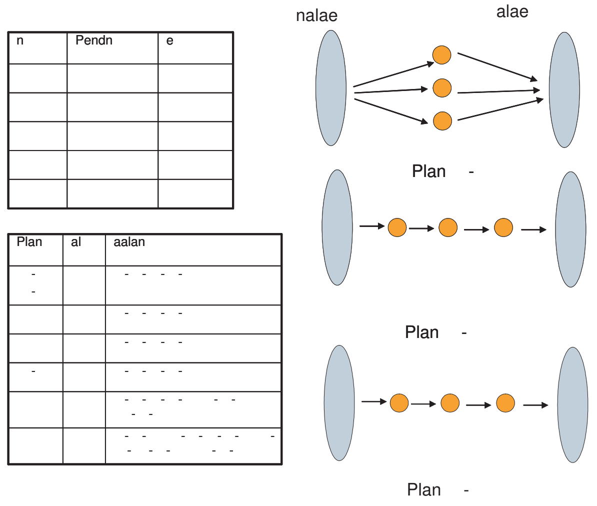

Figure 1: Example illustrating bases for distance measures. A_i and A_g denote dummy actions producing the initial state and consuming the goal state, respectively.

Example:

In Figure 1, three plans are shown for a planning problem where the initial state is $\langle p_1, p_2, p_3 \rangle$ and the goal state is $\langle g_1, g_2, g_3 \rangle$. The action models and the causal structures of plans are shown to the left. Plans S1-1 and S1-2 have the same actions but different ordering structures. S1-1 has parallel actions while S1-2 has them in sequence. The plan S1-3 has A_1 like the other plans but all other actions are different ($A_2^{'}$ and $A_3^{'}$). However, it also achieves the same goals.

An action based plan comparison method which uses position-based distance aggregation would find S1-1, S1-2 and S1-3 to be all different. This is because all the three plans have different sets of action prefixes. If instead, the action information is used with set differencing, S1-1 and S1-2 would be found to be identical.

A state based comparison method which uses any of the given computation choice would find S1-2 and S1-3 to be identical, and both of them to be different from S1-1. This is because the states after every transition in S1-2 and S1-3 are identical. S1-1, on the other hand, has (trivially) the same first and last states but no intermediate states.

A causal link based comparison method which uses set differencing would find S1-1 and S1-2 to be the same while S1-3 as different. This is because the causal links for goals g_2 and g_3 in S1-1 (S1-2) are different from those of S1-3.

3 Finding Diverse Plans with GP-CSP

The GP-CSP planner[Do & Kambhampati, 2001] converts Graphplan's planning graph into a CSP encoding, and solves it using a standard CSP solver. The solution of the encoding represents a valid plan for the original planning problem. In the encoding, the CSP variables correspond to the predicates that have to be achieved at different levels in the planning graph (different planning steps) and their possible values are the actions that can support the predicates. For each CSP variable representing a predicate p, there are two special values: i) $\perp$: indicates that a predicate is not supported by any action and is *false* at a particular level/planning-step; ii) "noop": indicates that the predicate is true at a given level i because it was made true at some previous level $j < i$ and no other action deletes p between j and i. Constraints encode the relations between predicates and actions: 1) mutual exclusion relations between predicates and actions; and 2) the causal relationships between actions and their preconditions.

3.1 Adapting GP-CSP to Different Distance Bases

When the above planning encoding is solved by any standard CSP solver, it will return a solution containing $\langle$*var, value*$\rangle$ of the form $\{\langle x_1, y_1\rangle, ...\langle x_n, y_n\rangle\}$. The collection of x_i where $y_i \neq \perp$ represents the facts that are made true at different time steps (plan trajectory) and can be used as a basis for the *state-based* distance measure; the set of $(y_i \neq \perp) \wedge (y_i \neq noop)$ represents the set of actions in the plan and can be used for *action-based* distance measure; lastly, the assignments $\langle x_i, y_i\rangle$ themselves represent the causal relations and can be used for the *causal-based* distance measure.

However, there are several complications we need to overcome before a specific distance measure between plans can be computed. First, the same action can be represented by different values in the domains of different variables. Consider a simple example in which there are two facts p and q, both supported by two actions a_1 and a_2. When setting up the CSP encoding, we assume that the CSP variables x_1 and x_2 are used to represent p and q. The domains for x_1 and x_2 are $\{v_{11}, v_{12}\}$ and $\{v_{21}, v_{22}\}$, both representing the two actions $\{a_1, a_2\}$ (in that order). The assignments $\{\langle x_1, v_{11}\rangle, \langle x_2, v_{21}\rangle\}$ and $\{\langle x_1, v_{12}\rangle, \langle x_2, v_{22}\rangle\}$ have a distance of 2 in traditional CSP because different values are assigned for each variable x_1 and x_2. However, they both represent the same action set $\{a_1, a_2\}$ and thus lead to the plan distance of 0 if we use the action-based distance in our plan comparison. Therefore, we first need to translate the set of values in all assignments back to the set of action instances before doing comparison using action-based distance. The second complication arises for the causal-based distance. A causal link $a_1 \xrightarrow{p} a_2$ between two actions a_1 and a_2 indicates that a_1 supports the precondition p of a_2. However, the CSP assignment $\langle p, a_1\rangle$ only provides the first half of each causal-link. To complete the causal-link, we need to look at the values of other assignments to identify action a_2 that occur at the later level in the planning graph and has p as its precondition.

3.2 Making GP-CSP Return a Set of Solutions

To make GP-CSP return a set of solutions satisfying the *d*DISTANT*k*SET constraint using one of the three distances, we add "global" constraints to each original encoding to enforce *d*-diversity between every pair of solutions. When each global constraint is called upon by the normal forward checking and arc-consistency checking procedures inside the default solver to check if the distance between two solutions is over a predefined value d, we first map each set of assignments to an actual set of actions (action-based), predicates that are true at different plan-steps (state-based) or causal-links (causal-based) using the method discussed in the previous section. This process is done by mapping all $\langle var, value\rangle$ CSP assignments into action sets using a call to the planning graph, which is outside of the CSP solver, but works closely with the general purpose CSP solver in GP-CSP. The comparison is then done within the implementation of the global constraint to decide if two plans are diverse enough.

We investigate two different ways to use the global constraints: 1) *parallel* strategy to return the set of k solutions all at once; and 2) *greedy* strategy to return them one after another. In the *parallel* approach, we create one encoding that contains k identical copies of each original planning encoding created using GP-CSP planner. The k copies are connected together using $k(k-1)/2$ pair-wise global constraints. Each global constraint between the i^{th} and j^{th} copies ensures that two plans represented by the solutions of those two copies will be at least d distant from each other. If each copy has n variables, then this constraint involves $2n$ variables.

In the *greedy* approach, the k copies are not setup in parallel up-front, but sequentially. We add to the i^{th} copy one global constraint to enforce that the solution of the i^{th} copy should be d-diverse from any of those $i-1$ solutions. The advantage of the greedy approach is that each CSP encoding is significantly smaller in terms of the number of variables (n vs. $k*n$), smaller in terms of the number of global constraints (1 vs. $k(k-1)/2$), and each global constraint also contains lesser number of variables (n vs. $2*n$).[4] Thus, each encoding in the greedy approach is easier to solve. However, because each solution depends on all previously found solutions, the encoding can be unsolvable if the previously found solutions comprise a bad initial solution set.

3.3 Empirical Evaluation

We implemented the parallel and greedy approaches discussed earlier for the three distance measures and tested them with the benchmark set of *Logistics* problems provided with the Blackbox planner[Kautz & Selman, 1998]. All experiments were run on a Linux Pentium 4, 3Ghz machine with

[4]However, each constraint is more complicated because it encodes *(i-1)* previously found solutions.

	Prob1	Prob2	Prob3	Prob4	Prob5	Prob6
δ_a	0.087	7.648	1.021	6.144	8.083	178.633
δ_s	0.077	9.354	1.845	6.312	8.667	232.475
δ_c	0.190	6.542	1.063	6.314	8.437	209.287
Random	0.327	15.480	8.982	88.040	379.182	6105.510

Table 3: Average solving time (in seconds) to find a plan using greedy (first 3 rows) and by random (last row) approaches

	Prob1	Prob2	Prob3	Prob4	Prob5	Prob6
δ_a	0.041/0.35	0.067/0.65	0.067/0.25	0.131/0.1*	0.126/0.15	0.128/0.2
δ_s	0.035/0.4	0.05/0.8	0.096/0.5	0.147/0.4	0.140/0.5	0.101/0.5
δ_c	0.158/0.8	0.136/0.95	0.256/0.55	0.459/0.15*	0.346/0.3*	0.349/0.45

Table 4: Comparison of the diversity in the solution sets returned by the random and greedy approaches

512MB RAM. For each problem[5], we test with different d values ranging from 0.01 (1%) to 0.95 (95%)[6] and k increases from 2 to n where n is the maximum value for which GP-CSP can still find solutions within plan horizon. The horizon (parallel plan steps) limit is 30.

We found that the greedy approach outperformed the parallel approach and solved significantly higher number of problems. Therefore, we focus on the greedy approach hereafter. For each combination of d, k, and a given distance measure, we record the solving time and output the average/min/max pairwise distances of the solution sets.

Baseline Comparison: As a baseline comparison, we have also implemented a *random* approach. In this approach, we do not use global constraints but use random value ordering in the CSP solver to generate k different solutions without enforcing them to be pairwise d-distance apart. For each distance d, we continue running the random algorithm until we find k_{max} solutions where k_{max} is the maximum value of k that we can solve for the greedy approach for that particular d value. In general, we want to compare with our approach of using global constraint to see if the random approach can effectively generate diverse set of solutions by looking at: (1) the average time to find a solution in the solution set; and (2) the maximum/average pairwise distances between $k \geq 2$ randomly generated solutions.

Figure 3 shows the comparison of average solving time to find one solution in the greedy and random approaches. The results show that on an average, the random approach takes significantly more time to find a single solution, regardless of the distance measure used by the greedy approach. To assess the diversity in the solution sets, Table 4 shows the comparison of: (1) the average pairwise minimum distance between the solutions in sets returned by the random approach; and (2) the maximum d for which the greedy approach still can find a set of diverse plans. The comparisons are done for all three distance measures. For example, the first cell $(0.041, 0.35)$ in Table 4, implies that the minimum pairwise distance averaged for all solvable $k \geq 2$ using the random approach is $d = 0.041$ while it is 0.35 (i.e. 8x more diverse) for the greedy approach using the δ_a distance measure. Except for 3 cases, using global constraints to enforce minimum pairwise distance between solutions helps GP-CSP return significantly more diverse set of solutions. On an average, the greedy approach returns 4.25x, 7.31x, and 2.79x more diverse solutions than the random approach for δ_a, δ_s and δ_c, respectively.

[5]log-easy=prob1, rocket-a=prob2, log-a = prob3, log-b = prob4, log-c=prob5, log-d=prob6.

[6]Increments of 0.01 from 0.01 to 0.1 and of 0.05 thereafter.

d	Prob1	Prob2	Prob3	Prob4	Prob5	Prob6
0.01	11,5, **28**	8,**18**,12	9,8,**18**	3,4,**5**	4,6,**8**	**8**,7,7
0.03	6,3,**24**	8,**13**,9	7,7,**12**	2,**4**,3	4,**6**,**6**	4,**7**,6
0.05	5,3,**18**	6,**11**,9	5,7,**10**	2,**4**,3	4,**6**,5	3,**7**,5
0.07	2,3,**14**	6,**10**,8	4,**7**,6	2,**4**,2	4,**6**,5	3,**7**,5
0.09	2,3,**14**	6,**9**,6	3,**6**,**6**	2,**4**,2	3,**6**,4	3,**7**,4
0.1	2,3,**10**	6,**9**,6	3,**6**,**6**	2,**4**,2	2,**6**,4	3,**7**,4
0.2	2,3,**5**	5,**9**,6	2,**6**,**6**	1,**3**,1	1,**5**,2	2,**5**,3
0.3	2,2,**3**	4,**7**,5	1,**4**,**4**	1,**2**,1	1,**3**,2	1,**3**,**3**
0.4	1,2,**3**	3,**6**,5	1,**3**,**3**	1,**2**,1	1,**2**,1	1,2,**3**
0.5	1,1,**3**	2,4,**5**	1,**2**,**2**	-	1,**2**,1	1,**2**,1
0.6	1,1,**2**	2,3,**4**	-	-	-	-
0.7	1,1,**2**	1,**2**,**2**	-	-	-	-
0.8	1,1,**2**	1,**2**,**2**	-	-	-	-
0.9	-	1,1,**2**	-	-	-	-

Table 5: For each given d value, each cell shows the largest solvable k for each of the three distance measures δ_a, δ_s, and δ_c (in this order). The maximum values in cells are in bold.

Analysis of the different distance-bases: Overall, we were able to solve 1264 (d, k) combinations for three distance measures $\delta_a, \delta_s, \delta_c$ using the greedy approach. We were particularly interested in investigating the following issues:

- **H1** - Is it easy or difficult to find a set of diverse solutions using different distance measures? Thus, (1) for the same d and k values, which distance measure is more difficult (time consuming) to solve; and (2) given an encoding horizon limit, how high is the value of d and k that we can still find set of solutions for a given problem using different distance measures.
- **H2** - What, if any, is the correlation/sensitivity between different distance measures? Thus, how diverse the solutions returned when using distance measure δ' are according to another $\delta'' \neq \delta'$ (where $\delta', \delta'' \in \{\delta_a, \delta_s, \delta_c\}$)

Regarding *H1*, Table 5 shows the highest solvable k value for each distance d and base δ_a, δ_s, and δ_c. For a given (d, k) pair, enforcing δ_a appears to be the most difficult, then δ_s, and δ_c is the easiest. GP-CSP is able to solve 237, 462, and 565 combinations of (d, k) respectively for δ_a, δ_s and δ_c. GP-CSP solves *d*DISTANT*k*SET problems more easily with δ_s and δ_c than with δ_a due to the fact that solutions with different action sets (diverse with regard to δ_a) will likely cause different trajectories and causal structures (diverse with regard to δ_s and δ_c). Between δ_s and δ_c, δ_c solves more problems for easier instances (Problems 1-3) but less for the harder instances, as shown in Table 5. We conjecture that for solutions with more actions (i.e. in bigger problems) there are more causal dependencies between actions and thus it is harder to reorder actions to create a different causal-structure.

For running time comparisons, among 216 combinations of (d, k) that were solved by all three distance measures, GP-CSP takes the least amount of time for δ_a in 84 combinations, for δ_s in 70 combinations and 62 for δ_c. The first three lines of Table 3 show the average time to find one solution in d-diverse k-set for each problem using δ_a, δ_s and δ_c (which we call t_a, t_s and t_c respectively). In general, t_a is the smallest and $t_s > t_c$ in most problems. Thus, while it is harder to enforce δ_a than δ_s and δ_c (as indicated in Table 5), when the encodings for all three distances can be solved for a given (d, k), then δ_a takes less time; this can be due to tighter constraints (more pruning power for the global constraints) and simpler global constraint setting.

To test *H2*, in Table 6, we show the cross-validation between different distance measures δ_a, δ_s, and δ_c. In this table,

	δ_a	δ_s	δ_c
δ_a	-	1.262	1.985
δ_s	0.485	-	0.883
δ_c	0.461	0.938	-

Table 6: Cross-validation of distance measures δ_a, δ_s, and δ_c.

cell $\langle$*row, column*$\rangle = \langle \delta', \delta'' \rangle$ indicates that over all combinations of (d, k) solved for distance δ', the average value d''/d' where d'' and d' are distance measured according to δ'' and δ', respectively ($d' \geq d$). For example, $\langle \delta_s, \delta_a \rangle = 0.485$ means that over 462 combinations of (d, k) solvable for δ_s, for each d, the average distance between k solutions measured by δ_a is $0.485 * d_s$. The results indicate that when we enforce d for δ_a, we will likely find even more diverse solution sets according to δ_s ($1.26 * d_a$) and δ_c ($1.98 * d_a$). However, when we enforce d for either δ_s or δ_c, we are not likely to find a more diverse set of solutions measured by the other two distance measures. Nevertheless, enforcing d using δ_c will likely give comparable diverse degree d for δ_s ($0.94 * d_c$) and vice versa. We also observe that d_s is highly dependent on the difference between the parallel lengths of plans in the set. d_s seems to be the smallest (i.e $d_s < d_a < d_c$) when all k plans have the same/similar number of time steps. This is consistent with the fact that δ_a and δ_c do not depend on the steps in the plan execution trajectory while δ_s does.

4 Finding Diverse Plans with LPG

LPG is a local-search-based planner, that incrementally modifies a partial plan in a search for a plan that contains no flaws [Gerevini, Saetti, & Serina, 2003]. The behavior of LPG is controlled by an evaluation function that is used to select between different plan candidates in a neighborhood generated for local search. At each search step, the elements in the search neighborhood of the current partial plan π are the alternative possible plans repairing a selected flaw in π. The elements of the neighborhood are evaluated according to an *action evaluation function* E [Gerevini, Saetti, & Serina, 2003]. This function is used to estimate the cost of either adding or of removing an action node a in the partial plan π being generated. In the next two subsections, we present some modifications to LPG for computing diverse plans using the action-based plan distance.

4.1 Revised Evaluation Function for Diverse Plans

In order to manage dDISTANCEkSET problems, the function E has been extended to include an additional evaluation term that has the purpose of penalizing the insertion and removal of actions that *decrease* the distance of the current partial plan π under adaptation from a reference plan π_0. In general, E consists of four weighted terms, evaluating four aspects of the quality of the current plan that are affected by the addition ($E(a)^i$) or removal ($E(a)^r$) of a

$$E(a)^i = \alpha_E \cdot Execution\_cost(a)^i + \alpha_T \cdot Temporal\_cost(a)^i + \\ + \alpha_S \cdot Search\_cost(a)^i + \alpha_D \cdot |(\pi_0 - \pi) \cap \pi^i_{\mathsf{R}}|$$

$$E(a)^r = \alpha_E \cdot Execution\_cost(a)^r + \alpha_T \cdot Temporal\_cost(a)^r + \\ + \alpha_S \cdot Search\_cost(a)^r + \alpha_D \cdot |(\pi_0 - \pi - a) \cap \pi^r_{\mathsf{R}}|.$$

The first three terms of the two forms of E are unchanged from the standard behavior of LPG. The fourth term, used only for computing diverse plans, is the new term estimating how the proposed plan modification will affect the distance from the reference plan π_0. Each cost term in E is computed using a relaxed temporal plan π_{R} [Gerevini, Saetti, & Serina, 2003].

The π_{R} plans are computed by an algorithm, called RelaxedPlan, formally described and illustrated in [Gerevini, Saetti, & Serina, 2003], that we have slightly modified to penalize the selection of actions decreasing the plan distance from the reference plan. The specific change to RelaxedPlan for computing diverse plans is very similar to the change described in [Fox *et al.*, 2006], and it concerns the heuristic function for selecting the actions for achieving the subgoals in the relaxed plans. In the modified function for RelaxedPlan, we have an extra 0/1 term that penalizes an action b for π_{R} if its addition decreases the distance of $\pi + \pi_{\mathsf{R}}$ from π_0 (in the plan repair context investigated in [Fox *et al.*, 2006], b is penalized if its addition *increases* such a distance).

The last term of the modified evaluation function E is a measure of the decrease in plan distance caused by adding or removing a: $|(\pi_0 - \pi) \cap \pi^i_{\mathsf{R}}|$ or $|(\pi_0 - \pi - a) \cap \pi^r_{\mathsf{R}}|$, where π^i_{R} contains the new action a. The α-coefficients of the E-terms are used to weigh their relative importance.[7] The values of the first 3 terms are automatically derived from the expression defining the plan metric for the problem [Gerevini, Saetti, & Serina, 2003]. The coefficient for the fourth new term of E (α_D) is automatically set during search to a value proportional to $d/\delta_a(\pi, \pi_0)$, where π is the current partial plan under construction. The general idea is to dynamically increase the value of α_D according to the number of plans n that have been generated so far: if n is much higher than k, the search process consists of finding many solutions with not enough diversification, and hence the importance of the last E-term should increase.

4.2 Making LPG Return a Set of Plans

In order to compute a *set* of k d-distant plans solving a dDISTANCEkSET problem, we run the LPG search multiple times, until the problem is solved, with the following two additional changes to the standard version of LPG: (i) the preprocessing phase computing mutex relations and other reachability information exploited during the relaxed plan construction is done only once for all runs; (ii) we maintain an incremental set of valid plans, and we dynamically select one of them as the reference plan π_0 for the next search. Concerning (ii), let $P = \{S_1, ..., S_n\}$ be the set of n valid plans that have been computed so far, and *CPlans*(S_i) the subset of P containing all plans that have a distance greater than or equal to d from a reference plan $S_i \in P$. The reference plan π_0 used in the modified heuristic function E is a plan $S_{max} \in P$ which has a maximal set of diverse plans in P, i.e.,

$$S_{max} = ARGMAX_{\{S_i \in P\}} \{|CPlans(S_i)|\}.$$

S_{max} is incrementally computed each time the local search finds a new solution. In addition to being used to identify the reference plan in E, S_{max} is also used for defining the initial state (partial plan) of the search process. Specifically, we initialize the search using a (partial) plan obtained by randomly removing some actions from a (randomly selected) plan in the set *CPlans*$(S_{max}) \cup \{S_{max}\}$.

The process of generating diverse plans starting from a dynamically chosen reference plan continues until at least k

[7] These coefficients are also normalized to a value in $[0, 1]$ using the method described in [Gerevini, Saetti, & Serina, 2003].

plans that are all d-distant from each other have been produced. The modified version of LPG to compute diverse plans is called LPG-d.

4.3 Experimental Analysis with LPG-d

The distance function δ_a, using set-difference, can be written as the sum of two terms:

$$\delta_a(S_i, S_j) = \frac{|S_i - S_j|}{|S_i| + |S_j|} + \frac{|S_j - S_i|}{|S_i| + |S_j|.}$$

The first term represents the contribution of the actions in S_i to the plan difference, while the second term indicates the contribution of S_i to δ_a. We experimentally observed that in some cases the differences between two diverse plans computed using δ_a are mostly concentrated in only one of the δ_a components. This asymmetry means that one of the two plans can have many more actions than the other one, which could imply that the quality of one of the two plans is much worse than the quality of the other plan. In order to avoid this problem, we can parametrize δ_a by imposing the two extra constraints

$$\delta_a^A \geq d/\gamma \text{ and } \delta_a^B \geq d/\gamma$$

where δ_a^A and δ_a^B are the first and second terms of δ_a, respectively, and γ is an integer parameter "balancing" the diversity of S_i and S_j.

In this section, we analyze the performance of the modified version of LPG, called LPG-d, in three different benchmark domains from the 3rd and 5th IPCs.[8] The main goals of the experimental evaluation were (i) showing that LPG-d can efficiently solve a large set of (d, k)-combinations, (ii) investigating the impact of the δ_a γ-constraints on performance, (iii) comparing LPG-d and the standard LPG.

We tested LPG-d using both the default and parametrized versions of δ_a, with $\gamma = 2$ and $\gamma = 3$. We give detailed results for $\gamma = 3$ and a more general evaluation for $\gamma = 2$ and the original δ_a. We consider d that varies from 0.05 to 0.95, using 0.05 increment step, and with k = 2...5, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32 (overall, a total of 266 (d, k)-combinations). Since LPG-d is a stochastic planner, we use the median of the CPU times (in seconds) and the median of the average plan distances (over five runs). The average plan distance for a set of k plans solving a specific (d, k)-combination (δ^{av}) is the average of the plans distances between all pairs of plans in the set. The tests were performed on an AMD Athlon(tm) XP 2600+, 512 Mb RAM. The CPU-time limit was 300 seconds.

The 1st column of Figure 2 gives results for the largest problem in IPC-3 `DriverLog-Time` (fully-automated track). LPG-d solves 109 (d, k)-combinations, including combinations $d \leq 0.4$ and $k = 10$, and $d = 0.95$ and $k = 2$. The average CPU time (top plots) is 162.8 seconds. The average δ^{av} (bottom plots) is 0.68, with δ^{av} always greater than 0.4. With the original δ_a function LPG-d solves 110 (d, k)-combinations, the average CPU time is 160 seconds, and the average δ^{av} is 0.68; while with $\gamma = 2$ LPG-d solves 100 combinations, the average CPU time is 169.5 seconds, and the average δ^{av} is 0.69.

The 2nd column of Figure 2 gives results for the largest problem in IPC-3 `Satellite-Strips`. LPG-d solves 211 (k, d)-combinations; 173 of them require less than 10 seconds. The average CPU time is 12.1 seconds, and the average δ^{av} is 0.69. We observed similar results when using the original δ_a function or the parametrized δ_a with $\gamma = 2$ (in the second case, LPG-d solves 198 problems, while the average CPU time and the average δ^{av} are nearly the same as with $\gamma = 3$).

The 3rd column of Figure 2 gives results for a middle-size problem in IPC-5 `Storage-Propositional`. LPG-d solves 225 (k, d)-combinations, 39 of which require less than 10 seconds, while 128 of them require less than 50 seconds. The average CPU time is 64.1 seconds and the average δ^{av} is 0.88. With the original δ_a LPG-d solves 240 (k, d)-combinations, the average CPU time is 41.8 seconds, and the average δ^{av} is 0.87; with $\gamma = 3$ LPG-d solves 206 combinations, the average CPU time is 69.4 seconds and the average δ^{av} is 0.89.

The local search in LPG is randomized by a "noise" parameter that is automatically set and updated during search [Gerevini, Saetti, & Serina, 2003]. This randomization is one of the techniques used for escaping local minima, but it also can be useful for computing diverse plans: if we run the search multiple times, each search is likely to consider different portions of the search space, which can lead to different solutions. It is then interesting to compare LPG-d and a method in which we simply run the standard LPG until k d-diverse plans are generated. An experimental comparison of the two approaches show that in many cases LPG-d performs better. In particular, the new evaluation function E is especially useful for planning problems that are easy to solve for the standard LPG, and that admit many solutions. In these cases, the original E function produces many valid plans with not enough diversification. This problem is significantly alleviated by the new term in E. An example of domain where we observed this behavior is `logistics`.[9]

5 Related Work

Researchers including Tate [Tate, Dalton, & Levine, 1998] and Myers [Myers, 2005; Myers & Lee, 1999] have articulated the need for finding dissimilar plans. Myers, in particular, presents an approach to generate diverse plans in the context of her HTN planner by requiring the meta-theory of the domain to be available and using bias on the meta-theoretic elements to control search [Myers & Lee, 1999]. The metatheory of the domain is defined in terms of predefined attributes and their possible values covering roles, features and measures. Our work differs from hers in two respects. First, we focus on domain-independent distance measures. Second we consider the computation of diverse plans in the context of state of the art domain independent planners.

The problem of finding similar plans has been considered in the context of replanning. A recent effort in this direction is [Fox *et al.*, 2006]. Our work focuses on the problem of finding diverse plans by a variety of distance measures.

Outside the planning literature, our closest connection is to the work by Hebrard et al 2005, who solve the problem

[8] We tested LPG-d with other domains and problems, obtaining generally good results, that we omit for lack of space.

[9] E.g., for `logistics_a` (**prob3** of Table 3) LPG-d solved 128 instances, 41 of them in less than 1 CPU second and 97 of them in less than 10 CPU seconds; the average CPU time was 16.7 seconds and the average δ^{av} was 0.38. While using the standard LPG, only 78 instances were solved, 20 of them in less than 1 CPU seconds and 53 of them in less than 10 CPU seconds; the average CPU time was 23.6 seconds and the average δ^{av} was 0.27.

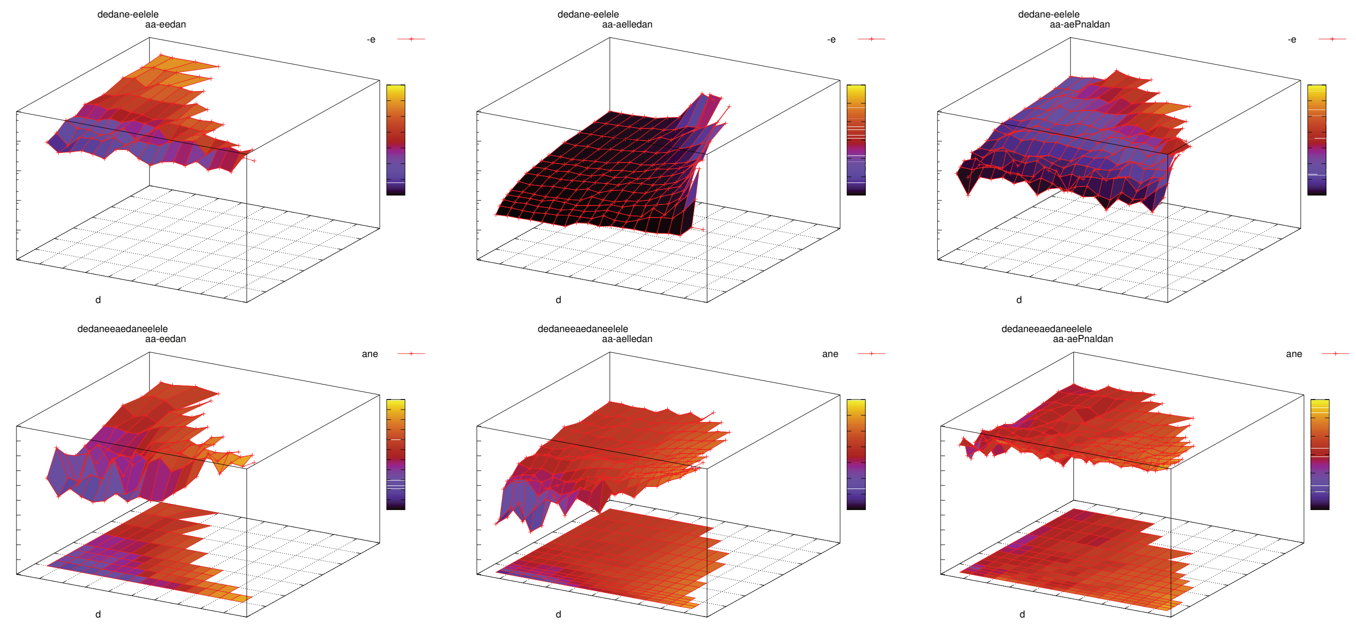

Figure 2: Performance of LPG-d (CPU-time and plan distance) for there problems in `DriverLog-Time`, `Satellite-Strips` and `Storage-Propositional`.

of finding similar/ dissimilar solutions for CSPs without additional domain knowledge. It is instructive to note that unlike CSP, where the number of potential solutions is finite (albeit exponential), the number of distinct plans for a given problem can be infinite (since we can have infinitely many non-minimal versions of the same plan). Thus, effective approaches for generating diverse plans are even more critical. The challenges in finding interrelated plans also bear some tangential similarities to the work in information retrieval on finding similar or dissimilar documents (c.f. [Callan & Minka, 2002]).

6 Conclusion

In this paper, we focused on domain-independent approaches for finding diverse plans. We articulated three different domain-independent distance measures for comparing plans. We then developed effective approaches for using these distance measures to bias the performance of two state-of-the-art planning approaches — GP-CSP and LPG. The approaches we developed for supporting the generation of diverse plans in GP-CSP are broadly applicable to other planners based on bounded horizon compilation approaches for planning. Similarly, the techniques we developed for LPG, such as biasing the relaxed plan heuristics in terms of distance measures, are applicable to other heuristic planners. The experimental results with GP-CSP explicate the relative difficulty of enforcing the various distance measures, as well as the correlation among the individual distance measures (as assessed in terms of the sets of plans they find). The experiments with LPG demonstrate the potential of heuristic planning in producing large sets of highly diverse plans.

References

[Boddy *et al.*, 2005] Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of action generation for cyber security using classical planning. In *Proc. ICAPS*. AAAI.

[Callan & Minka, 2002] Callan, J., and Minka, T. 2002. Novelty and redundancy detection in adaptive filtering. In *Proc. SIGIR*. ACM Press.

[Chafle *et al.*, 2006] Chafle, G.; Dasgupta, K.; Kumar, A.; Mittal, S.; and Srivastava, B. 2006. Adaptation in Web Services Composition and Execution. In *Proc. ICWS*.

[Do & Kambhampati, 2001] Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *AI* 132(2):151–182.

[Fox *et al.*, 2006] Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proc. ICAPS*.

[Gerevini, Saetti, & Serina, 2003] Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research (JAIR)* 20:pp. 239–290.

[Hebrard *et al.*, 2005] Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proc. AAAI*.

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

[Myers & Lee, 1999] Myers, K., and Lee, T. J. 1999. Generating qualitatively different plans through metatheoretic biases. In *Proc. AAAI*.

[Myers, 2005] Myers, K. 2005. Metatheoretic plan summarization and comparison. In *Proc. ICAPS WK. Mixed-initiative Planning and Scheduling*.

[Srivastava, Vanhatalo, & Koehler, 2005] Srivastava, B.; Vanhatalo, J.; and Koehler, J. 2005. Managing the life cycle of plans. In *Proc. IAAI, pg 1569–1575*.

[Tate, Dalton, & Levine, 1998] Tate, A.; Dalton, J.; and Levine, J. 1998. Generation of multiple qualitatively different plan options. In *Proc. AIPS-98, Pittsburgh*. AIAI.