Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

A Combinatorial Search Perspective on Diverse Solution Generation

Satya Gautam Vadlamudi and **Subbarao Kambhampati**
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
{gautam , rao}@asu.edu

Abstract

Finding diverse solutions has become important in many combinatorial search domains, including Automated Planning, Path Planning and Constraint Programming. Much of the work in these directions has however focussed on coming up with appropriate diversity metrics and compiling those metrics in to the solvers/planners. Most approaches use linear-time greedy algorithms for exploring the state space of solution combinations for generating a diverse set of solutions, limiting not only their completeness but also their effectiveness within a time bound. In this paper, we take a combinatorial search perspective on generating diverse solutions. We present a generic bi-level optimization framework for finding cost-sensitive diverse solutions. We propose complete methods under this framework, which guarantee finding a set of cost sensitive diverse solutions satisficing the given criteria whenever there exists such a set. We identify various aspects that affect the performance of these exhaustive algorithms and propose techniques to improve them. Experimental results show the efficacy of the proposed framework compared to an existing greedy approach.

In many real-world domains involving combinatorial search such as automated planning, path planning and constraint programming, generating diverse solutions is of much importance. In the case of automated planning, real-world scenario often involves working with unknown or partially known user preferences (Kambhampati 2007), as the user preferences are many times difficult to be articulated and specified completely. Such situations lead to multiple, often, large number of plans that satisfy a given problem instance. In order to facilitate serving the user with a closest plan possible as per her (hidden) preferences, presenting a diverse set of plans to the user is explored (Roberts, Howe, and Ray 2014; Nguyen et al. 2012) so that the user can make a well-informed decision. In the constraint programming domain, diverse (resp. similar) solutions are explored in order to handle unknown user preferences as well as to generate robust solutions (Hebrard et al. 2005).

Several methods have been proposed in the literature for finding a diverse set of plans. In the context of constraint programming, (Hebrard et al. 2005) presents a complete method which creates K copies of the Constraint Satisfaction Problem (CSP), each copy with a different set of variable names, adds $\binom{K}{2}$ additional constraints for handling the minimum distance requirement between all pairs, and uses off-the-shelf solvers to generate solutions. As one would expect, they report that this approach generates prohibitively large CSPs and therefore propose a greedy method. The greedy approach which has since been widely adopted (Petit and Trapp 2015; Bloem 2015; Roberts, Howe, and Ray 2014; Nguyen et al. 2012) works as follows:

Obtain a candidate solution satisfying any given cost criteria, add this to the solution K-set, provide feedback to the method finding candidate solutions about the current composition of the K-set so that it tries to find the next candidate solution distant to the current K-set. Upon obtaining the next candidate solution, the greedy method adds it to the K-set if it indeed satisfies the distance criteria and provides feedback to find the next solution, otherwise the candidate solution is simply discarded. This process is continued until a set of K diverse solutions is found. (Roberts, Howe, and Ray 2014; Eiter et al. 2013) and (Nguyen et al. 2012) consider the first solution generated to be the starting solution (permanent member) for constructing the K-set through the above greedy approach. (Bloem 2015) considers an optimal solution to be the starting solution of the greedy method. (Petit and Trapp 2015) attempt to address the issue of fixed starting solution by running the greedy approach multiple times with different optimal solutions as the starting solution on each occasion.

A pertinent issue with the above approaches is that the first solution (or an optimal solution) is always considered to be part of the solution set, which may often result in not finding a K-set even when there exists one, even with a very good feedback strategy to search for distant solutions after finding the initial solution. Note that an optimal solution (or the first found solution) need not be part of a diverse set of the required size at all. Figure 1 shows an example of an instance where the optimal solution is not part of the most diverse solution set of size 2 (assuming that the distance between two plans is inversely proportional to the number of edges/actions they have in common; p1 and p2 have edge a in common and p2 and p3 have edge b in common).

In this paper, we address this problem in depth by proposing complete algorithms which guarantee to find a set of K diverse solutions whenever there exists one. In order to ac-

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

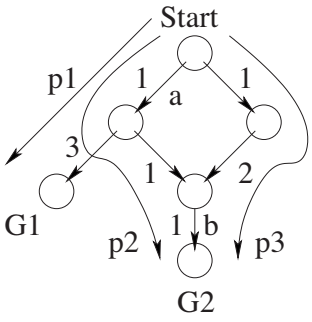

Figure 1: A state-space graph where the optimal cost path p2, does not belong to the most diverse solution set of size 2 {p1,p3}. G1 and G2 indicate goal nodes.

complish completeness, we first need methods for exploring all possible cost sensitive solutions in the given domain. For this, we present extensions of the m-A* algorithm (Dechter, Flerova, and Marinescu 2012) and the Depth-First Branch and Bound algorithm (Lawler and Wood 1966) for finding all cost-bounded plans. One could adapt other types of methods such as anytime heuristic search algorithms as well for this purpose. Second, we present a simple strategy for exhaustively exploring the set of all plan combinations. This would guarantee completeness for finding a K-set whenever there exists one, thereby addressing the issues in existing methods. However, we note that the simple exhaustive method ends up having to explore a large number of combinations as one finds more and more candidate plans, severely impacting the performance of the overall algorithm. In order to address this problem, we propose a method which considers only a few most promising plan combinations whenever a new candidate solution is found, and postpones the exploration of remaining combinations for the end to guarantee completeness. This new method is advantageous over the widely followed greedy approach on two fronts: it explores a larger (compared to only one combination of the greedy approach) but limited number of combinations upon finding a candidate solution thereby increasing its likelihood of finding a diverse K-set quickly, and it keeps track of the combinations that are left postponed to revisit at the end thereby guaranteeing completeness.

Further, our method for exploring the plan-combinations space can be used in conjunction with any of the existing methods to improve their performance, as it is complementary in nature, replacing the weaker section of those methods where the greedy approach is present.

Related Work

Several applications have been related to using diverse solutions in recent years, such as, for course of action generation in cyber security (Boddy et al. 2005), personalized security agents (Roberts et al. 2012), diverse finite state machines for non-player characters in games (Coman and Muñoz-Avila 2013; 2012b), formal verification (Nadel 2011), mining group patterns (Vadlamudi, Chakrabarti, and Sarkar 2012), scheduling personal activities (Alexiadis and Refanidis 2013), air traffic control advisories (Bloem and Bambos 2014), and robotics (Voss, Moll, and Kavraki 2015).

An important measure in determining diversity is the distance between plans. In this paper, we assume that the distance measure is given as input by the user. Several distance measures have been proposed in the literature that are quantitative or qualitative (Scala 2014; Coman and Muñoz-Avila 2011; Goldman and Kuter 2015). Solution diversity is explored in both deterministic and non-deterministic domains using the distance metrics (Coman 2012). Distance measures for finding semantically distinct plans are explored in (Bryce 2014) based on landmarks. In the context of constraint programming, distance constraints in terms of ideal and non-ideal solutions are studied in (Hebrard, O'Sullivan, and Walsh 2007).

SAT-based heuristic methods for generating diverse solutions were proposed in (Nadel 2011). Methods through compilation to CSP, and using heuristic local search have been proposed in (Srivastava et al. 2007), which use GP-CSP planner (Do and Kambhampati 2001) and LPG planner (Gerevini, Saetti, and Serina 2003). Comparison of first-principle techniques and case-based planning techniques to find diverse plans is shown in (Coman and Muñoz-Avila 2012a). These algorithms too use the greedy approach presented before for exploring the space of plan combinations, leading to the same issues pointed in the Introduction.

Problem Setup

In this paper, we consider the problem of finding a set of solutions that are not only diverse but are also cost sensitive. In particular, we consider the problem of finding a set of K cost sensitive diverse (loopless) solutions. Cost sensitivity of the solutions is controlled by the input c (maximum cost of each of the solutions) and diversity of the solution sets is controlled by the input d (minimum distance between each pair of solutions; or an appropriate set based diversity metric). Both the cost metric and the distance metric are also assumed to be inputs from the user, hence, the studies on good quality cost metrics and distance metrics are orthogonal to our work. Further, we choose the planning domain to showcase our framework and methods, which could be adapted to other domains. Hence, the problem at-hand can be formally stated as: Given a planning problem with the set of loopless solution plans $\mathcal{S}$, a cost metric for the plans $\mathcal{C} : \mathcal{S} \rightarrow \mathbb{R}$ and a distance metric for the pairs of plans $\delta : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ (a set based diversity metric may also be used here), the problem is defined as:

$$c\text{COST}d\text{DISTANT}k\text{SET: Find } \mathcal{P} \text{ with } \mathcal{P} \subseteq \mathcal{S}, \\ |\mathcal{P}| = k, \min_{p,\, q \,\in\, \mathcal{P}} \delta(p,q) \geq d \text{ and } \mathcal{C}(p) \leq c \;\forall p \in \mathcal{P} \quad (1)$$

The problem is computationally hard given that the problem of finding cost-bounded plans is PSPACE-complete (Bylander 1991) and the problem of finding a diverse set of plans is NP-complete (Bloem 2015) with input size (number of plans) that can potentially be exponential in terms of the number of state variables. Finding a set of diverse solutions is shown to be FP$^{NP[log\ n]}$-complete in the context of constraint programming (Hebrard et al. 2005), where n is the size of the input.

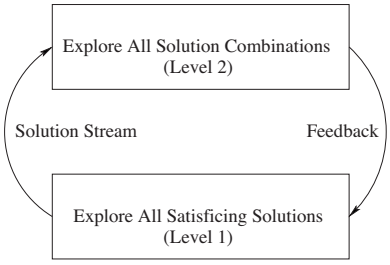

Figure 2: A framework for finding a diverse set of solutions that supports completeness.

Proposed Methods & Properties

Now, we present the proposed framework which supports completeness, and specific methods that obey the framework requirements. As mentioned above, the problem of finding a diverse set of plans is a bi-level optimization problem which involves exploring the set of all candidate plans (Level 1) and considering the set of all combinations of these plans (Level 2). Therefore, in order to guarantee completeness, we propose to have a framework with a complete method which guarantees finding all the candidate plans and outputs as a stream, and another complete method that takes the stream of plans being generated by the previous method as input and explores all combinations of plan sets as per the diversity criteria until a diverse set of size K is found. Such a framework ensures that all possible cases are considered and hence guarantees completeness. Figure 2 shows the framework with the control flow. In this paper, we emphasize mainly on how to explore all the solutions and all the solution combinations efficiently so as to guarantee completeness while not impeding the search progress due to their individual exhaustive nature. The feedback component shown in the figure is particular to the domain elements and their distance measures which we do not explore in this work leaving it as an option for the user to plug-in to the proposed framework and methods.

Algorithms for Finding All Cost Sensitive Solutions

For the Level 1, the problem is to find the set of all candidate plans that can potentially be part of a diverse solution set. In our case, the set of all candidate plans correspond to the set of all valid loopless plans whose cost is $\leq max\_cost$. We present two complete methods which guarantee generating the set of all candidate plans, one based on Depth-First Branch and Bound (DFBB) (Lawler and Wood 1966; Russell and Norvig 1995) and another based on a recent algorithm for finding M best solutions in graphical models, called m-A* (Dechter, Flerova, and Marinescu 2012). One may also use other complete methods and extend them to generate all candidate solutions whose cost is $\leq max\_cost$.

First, we present the DFBB based algorithm for finding all candidate plans, called DFA. It works similar to regular DFBB search on graph spaces except for the following two differences: (i) it does not stop after finding a single solution within the max_cost bound, and (ii) it does not conduct full duplicate detection (any state reached through a different path from the start state leads to a new node unless there is a loop, at which point it simply backtracks to find other solutions). It is easy to prove that:

Lemma 1 *DFA generates all valid loopless plans whose cost is $\leq max\_cost$, given that the edges of the search graph have positive costs and the heuristic used is admissible.*

Now, we describe the second algorithm for finding the set of all candidate plans, called A*A. This is based on the m-A* algorithm (Dechter, Flerova, and Marinescu 2012) which guarantees finding m best solutions/plans by expanding the minimum set of nodes. First, we describe the basic idea behind m-A* and then we extend it to find the set of all candidate plans for our problem. The basic idea behind m-A* is to proceed in a manner similar to A* and whenever a duplicate state is found, the most promising m nodes corresponding to that state are to be considered for expansion and the rest be discarded. For our problem, where we want to find the set of all cost-bounded plans, we will have to keep all the nodes corresponding to same state for expansion without the m-limit. This eliminates the requirement of full-scale duplicate detection all-together, instead suggests treatment of all children being generated as new. However, since we are only interested in loopless plans, we will discard all nodes which cause loops in the partial plan at any stage, through cycle checking. We call this adapted strategy- A*A (A* based approach for finding All cost-bounded solutions). Once again, it can be proven that:

Lemma 2 *A*A generates all valid loopless plans whose cost is $\leq max\_cost$, given that the edges of the search graph have positive costs and the heuristic used is admissible.*

Complete Algorithms for Finding a Diverse Solution Set

Now, we present the algorithms for Level 2 of our framework, where the stream of all candidate plans, the distance measure, the size of the diverse plan set needed and the minimum distance between any two plans of the solution set is given as input, to produce a set of satisficing diverse plans.

Algorithm 1 presents a simple strategy called ACER which explores all combinations of plans in each run for the in-coming new plan in conjunction with all of the existing valid plan sets. It is easy to prove that:

Lemma 3 *ACER finds a diverse plan set of size K from the set/stream of all candidate plans whenever there exists one.*

However, $\mathcal{G}$ can grow rapidly and become an exponential sized set in terms of K with base being the number of candidate plans (which itself can be of exponential size in terms of the planning problem input) before finding a diverse plan set. This severely limits its scalability when there are large number of candidate plans (even for moderate values of K), which is often the case in practice. Next, we will present a method which does not explore all plan set combinations in one shot instead only a select most promising sets at each stage, while keeping track of unexplored combinations that may be explored at the end (after processing the entire stream of candidate plans once) for completeness.

Algorithm 1 Explore All Possible Combinations of Solutions in Each Run (ACER)

INPUT :: A candidate plan p (from the stream of all candidate plans), a distance measure $dist()$, the minimum distance needed between any two plans of a set min_dist, and the size of the diverse plan set K.
OUTPUT :: A diverse plan set of size K (if exists), otherwise the largest diverse plan set.
$\mathcal{G} \leftarrow \phi$ (empty set); (initially) // Global data; the set of all valid plan sets.
for each plan set $P \in \mathcal{G}$ **do**
  if $dist(p, l) > min\_dist\ \forall l \in P$ **then**
    $P' \leftarrow P + \{p\}$;
    if $|P'| = K$ **then**
      return P';
    end if
    $\mathcal{G} \leftarrow \mathcal{G} \cup P'$;
  end if
end for
$\mathcal{G} \leftarrow \mathcal{G} \cup \{p\}$;
return largest $P \in \mathcal{G}$;

The second technique, which is an adaptation of the Anytime Pack Search method (Vadlamudi, Aine, and Chakrabarti 2015; 2013), focuses on exploring a limited set of seed nodes in each iteration in a beam search like manner. It processes the stream of candidate plans much faster than the previous approach by focusing only on a select number of most promising plan sets to begin with. The combinations which this technique ignores while processing the candidate plan stream are kept track of separately for processing at the end, which helps in guaranteeing the completeness. Algorithm 2 presents the proposed method MCER for faster processing of the stream of candidate plans. It takes as input, a plan from the stream of candidate plans and its sequence number for reasons that will become clear shortly, and the inputs for determining a diverse set similar to the previous approach, and the number of seed plan sets to be explored upon finding a new candidate plan.The plan-combinations space can be visualized as a set enumeration tree (Rymon 1992), where new branches come at all levels (> 0) dynamically as the stream of candidate plans is processed.

MCER maintains a global set of valid plan sets which have been produced until now, $\mathcal{G}$, that could be further expanded with new candidate plans. It expands n most promising nodes from this set, which are populated into $Children$. If a diverse set of size K is found, it terminates returning the set. Otherwise, n most promising plan sets from $Children$ are expanded, and then their n most promising children and so on until there are no further children to be expanded. It should also be mentioned at this point, as to what we mean by 'most promising', which would be based on f-value of a plan set (the largest being most promising), and f-value is in-turn computed as $g + h$ where g is the size of the plan set and h is the heuristic estimate denoting potential number of plans that can be added to this plan set. In this paper, we have explored using three heuristics: (i) 0, (ii) dispersion of the current set (arithmetic mean of all pair-wise distances (Myers and Lee 1999)) divided by min_dist, and

Algorithm 2 Explore Most-promising Combinations of Solutions in Each Run (MCER)

INPUT :: A candidate plan p (from the stream of all candidate plans), its sequence number in the stream i, a distance measure $dist()$, the minimum distance needed between any two plans of a set min_dist, the size of the diverse plan set K, and the number of seed plan sets to be explored n.
OUTPUT :: A diverse plan set of size K (if exists), otherwise the largest diverse plan set.
$\mathcal{G} \leftarrow \phi$ (empty set); (initially) // Global data; the set of all valid plan sets with satellite data.
$Open \leftarrow \phi$; $Children \leftarrow \phi$;
ExpandMostPromising($\mathcal{G}$, n, $Children$);
if a diverse set P of size K is found **then**
  return P;
end if
while $Children \neq \phi$ **do**
  Swap $Children$ and $Open$;
  ExpandMostPromising($Open$, n, $Children$);
  if a diverse set P of size K is found **then**
    return P;
  end if
  Move all plan sets in $Open$ to $\mathcal{G}$;
end while
$P \leftarrow \{p\}$; $P_{exp} \leftarrow i$;
$\mathcal{G} \leftarrow \mathcal{G} \cup P$;
return largest $P \in \mathcal{G}$;

(iii) quadratic mean of all distances divided by min_dist. The idea behind these heuristics is that the more dispersion the sets have the more accommodative they could be of new candidate plans. However, in our experiments, we did not observe gains of using the dispersion based heuristics in our experiments compared to the trivial heuristic possibly due to the limitation of the said heuristics in accounting for the actions that are not part of the plans found yet, at any given moment during the runtime. More distance metric based and domain based estimates can be explored here in future.

Algorithm 3 presents the pseudo-code of ExpandMost-Promising routine. It expands the n most promising nodes from the given list (either $\mathcal{G}$ or $Open$) and puts them in $Children$. One significant difference to note here is that, since all the candidate plans are not available apriori, one must add the expanded nodes back to $\mathcal{G}$ for future consideration with newer candidate plans. While doing so, in order to avoid repetition, we keep track of the last child generation attempt through the sequence number of candidate plan considered.

Finally, after the entire stream of candidate solutions has been processed one by one using MCER, if a diverse set of size K is not found, we continue to call MCER repeatedly (this time, without adding back the explored plan sets into $\mathcal{G}$) until it finds a K-set or terminates exhausting the exploration of all possible combinations.

Below, we present some of the properties of the proposed method MCER:

Lemma 4 *MCER does not generate the same combination of plans more than once.*

Proof outline: This is ensured by keeping track of the

Algorithm 3 ExpandMostPromising

1: **INPUT ::** A set of valid plan sets S to expand, $Children$, a distance measure $dist()$, the minimum distance needed between any two plans of a set min_dist, the size of the diverse plan set K, and the number of plan sets to be explored n.
2: **OUTPUT ::** Populates $Children$ with new valid plan sets, returns a diverse plan set of size K if found.
3: $Temp \leftarrow \phi$ (empty set);
4: **for** n times **do**
5: $P \leftarrow$ most promising plan set from S;
6: **for** each candidate plan in the stream from sequence number $i = P_{exp} + 1$ to the latest **do**
7: **if** $dist(p, l) > min\_dist \; \forall l \in P$ **then**
8: $P' \leftarrow P + \{p\}$; $P'_{exp} \leftarrow i$;
9: **if** $|P'| = K$ **then**
10: **return** P';
11: **end if**
12: $Children \leftarrow Children \cup P'$;
13: **end if**
14: $P_{exp} \leftarrow i$;
15: **end for**
16: $Temp \leftarrow Temp \cup P$;
17: **end for**
18: $\mathcal{G} \leftarrow \mathcal{G} \cup Temp$;

sequence number of candidate plan from the last child generation attempt while expanding a plan set P via P_{exp}, which increases by 1 at each step during expansion (see Line 14 in Algorithm 3) and the child generation attempts start from $P_{exp} + 1$ every time (see Line 6 in Algorithm 3), thereby avoiding repetition. □

Lemma 5 *MCER expands at-most $n \times (K-1)+1$ number of plan sets in each execution.*

Proof outline: Note that, after expansion of n most promising nodes from $\mathcal{G}$, their n most promising children, and then their n most promising children and so on are expanded, until a child of size K is found. Further, size of the children at each step increases by 1 since a new candidate plan gets added to the plan set. Therefore, even if we assume that the initial set of seed nodes are all of size 1, MCER executes at-most K steps at which point a diverse set of size K will be found if possible through that set. And at each step, at-most n number of children are expanded, with only 1 at level K. Hence, together, at-most $n \times (K-1)+1$ number of plan sets are expanded in each execution of MCER. □

Lemma 6 *MCER guarantees finding a diverse set of plans of size K if there exists one.*

Proof outline: Note that, while we execute MCER several times with incoming plans from the stream of candidate plans, each time without exhausting all possible combinations, we keep track of the last expansion attempt for each node (plan set), and store them in $\mathcal{G}$. Hence all plan sets which may not have been exhaustively explored with the candidate plans are present in $\mathcal{G}$ when the entire set of candidate plans has been generated. These plan sets are then exhaustively explored without re-inserting back in to $\mathcal{G}$ thereby guaranteeing completeness and termination. □

Now, given a planning problem, a cost metric, a distance measure, max_cost, min_dist, and K, for finding a set of K cost sensitive diverse plans, one could use any one of the following four combinations: 1) DFA with ACER, wherein the DFA is executed and whenever a valid plan with cost $< max\_cost$ is found, ACER is invoked to find a diverse set, and then the execution of DFA is continued if a diverse set with the given requirements is not found, and the process is repeated until termination. We call this combination **DFAA**. 2) DFA with MCER, similar to the above strategy of invoking MCER whenever DFA find a valid cost sensitive plan, followed by repeated calls to MCER at the end to explore all the remaining plan combinations until termination. This is denoted by **DFAM**. 3) A*A with ACER (denoted by **A*AA**), and 4) A*A with MCER (denoted by **A*AM**). Next section presents the comparison of performances of the above combinations of methods.

Experimental Results

In this section, we present the experimental results comparing the performances of various proposed algorithms among themselves as well as with a greedy approach proposed in the literature. We have implemented all our methods on top of the Fast Downward planning environment (Helmert 2006), and hence could run problem instances from any of the supported planning domains. Accordingly, we have conducted experiments on several domains, including, *blocks, rovers, pathways-noneg, airport, driverlog, tpp, zenotravel*. We present the representative results in this paper. All the experiments have been performed on a machine with Intel(R) Xeon(R) CPU E5-1620 v2 at 3.70GHz and 64GB RAM. The following distance measure for measuring diversity has been adopted from (Nguyen et al. 2012):

$$dist(p_1, p_2) = 1 - \frac{A(p_1) \cap A(p_2)}{A(p_1) \cup A(p_2)} \quad (2)$$

where $A(p)$ denotes the set of all actions in plan p.

Table 1 shows the comparison of DFAA and A*AA methods on problems (denoted by P.no.) from *Blocks* domain. We have used the $LMcut$ heuristic which is admissible, to guide the search. The algorithms are given a maximum time of 60sec for solving each problem. Given a set of inputs, the output shows whether a diverse set of plans of size K is found (otherwise the size of the largest diverse set found in parenthesis), the time taken, and the number of plans generated during the process. * denotes that the algorithm stopped due to the time limit. We see that the DFA based method generates the cost sensitive plans faster than the A*A based method in this case, resulting in the processing of more number of plans in a given time.

The difference in the number of plans generated to find a diverse set of same size highlights the importance of the order in which the candidate plans are generated. Depending on the order of the plans generated, the number of plans required to be processed by an exhaustive algorithm to produce a diverse set of specific size varies. As mentioned before, this could be influenced by devising an appropriate distance metric and domain dependent feedback mechanism.

Table 1: Comparison of DFAA and A*AA complete algorithms. Domain: Blocks. $max\_time = 60sec.$

Input				DFAA			A*AA		
P. no.	K	max_cost	min_dist	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.
4-0	4	20	0.6	Yes	0.00	22	Yes	0.00	26
	8			No (4)	0.00	43	No (4)	0.00	43
	8	30	0.5	Yes	32.90	231	Yes	11.06	130
			0.6	No (5)	2.14	323	No (5)	3.42	323
5-0	8	30	0.5	No* (4)	60.00	2567	No* (6)	60.00	923
			0.6	No* (3)	60.00	5140	No* (3)	60.00	4115

Table 2: Comparison of DFAM and A*AM complete algorithms. Domain: Blocks. $max\_time = 60sec.$

Input				DFAM			A*AM		
P. no.	K	max_cost	min_dist	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.
4-0	4	20	0.6	Yes	0.00	26	Yes	0.00	26
	8			No (4)	0.00	43	No (4)	0.00	43
	8	30	0.5	Yes	0.58	323	Yes	0.04	172
			0.6	No (5)	1.74	323	No (5)	2.62	323
5-0	8	30	0.5	No* (6)	60.00	11680	No* (7)	60.00	12226
			0.6	No* (2)	60.00	11908	No* (3)	60.00	12311

Table 3: Comparison of DFA based and A*A based greedy algorithms. Domain: Blocks. $max\_time = 60sec.$

Input				DFAG			A*AG		
P. no.	K	max_cost	min_dist	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.
4-0	4	20	0.6	No (3)	0.00	43	No (3)	0.00	43
	8			No (3)	0.00	43	No (3)	0.00	43
	8	30	0.5	No (6)	0.04	323	No (7)	0.06	323
			0.6	No (4)	0.02	323	No (3)	0.04	323
5-0	8	30	0.5	No (6)	16.70	104712	No* (6)	Mem-limit	101908
			0.6	No (2)	24.24	104712	No* (2)	Mem-limit	101908

Table 2 presents the comparison of DFAM and A*AM methods (with plan-combinations seed set size equal to 30 in each execution). Note that, both algorithms guarantee to find a diverse set of required size if there exists one, given they are given enough time to terminate. Our objective with the MCER based methods is to quickly process the incoming candidate plans so as to find a diverse set quicker, postponing the exhaustive exploration to the end. Accordingly, we see that both the methods perform much better than they did compared to Table 1 by processing larger number of plans. They are able to find larger sized sets of diverse plans in the given time than before, although, as one can observe it may also happen that the select exploration could occasionally (see DFAM vs DFAA in last rows of both the tables) delay finding large sets compared to ACER based approaches.

Next, we present the results obtained by integrating the greedy approach discussed in the Introduction with DFA and A*A, in Table 3. We call these methods DFAG and A*AG respectively. As one can observe, while the greedy methods process the incoming candidate plans very fast, they terminate without finding a diverse set of given size even when there exists one. Furthermore, even for finding the diverse sets that they produced, they involve generating far more number of plans compared to the complete algorithms. This can be a crucial element when finding multiple plans is difficult for a domain. Also, note that, A*A runs out of memory (4GB per instance) in this case which makes the case for memory bounded methods while attempting to generate multiple solutions. Although, one can improve the performance of the greedy approach through feedback mechanisms, considering only one seed plan set for exploration is likely to continue to affect the performance. Thus, it would be beneficial to have multiple seed plan sets to be explored at each stage for better performance.

Now, we present the results obtained on two other domains, namely, Rovers and Zeno-Travel. We show the results with DFA as the base method (for generating all cost-bounded solutions) in these cases since A*A based methods were quickly reaching the memory limit on these instances. Table 4 shows the comparison of DFAA, DFAM and DFAG methods on a problem from the Rovers domain with 14 objects. Here, a diverse set of size 8 with cost bound 20 is to be found within 60 seconds. Three sets of results comparing the above three algorithms are presented with different diversity criterion in each case. Note that, amongst the three methods, DFAA spends the most amount of effort on exploring plan combinations (exhaustive) whenever a new plan is found, therefore is only able to generate and process a small number of plans. Since DFAG spends least amount of effort on exploring plan combinations (greedy) upon finding a new plan, it is able to generate and scan through a large number of plans. Whereas DFAM distributes its effort intelligently across plan generation and plan combination exploration, by adjusting the number of seeds n as per domain and

Table 4: Comparison of DFAA, DFAM and DFAG methods. Domain: Rovers. Problem: roverprob4213 (14 objects), K: 8, max_cost : 20, $max\_time = 60sec$.

Input	DFAA			DFAM			DFAG		
min_dist	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.
0.4	Yes	0.88	64	Yes	0.00	65	Yes	0.00	64
0.5	No* (6)	60.00	372	Yes	9.84	306061	Yes	1.22	304553
0.6	No* (4)	60.00	1048	No* (6)	60.00	2047871	No* (4)	60.00	15988265

Table 5: Comparison of DFAA, DFAM and DFAG methods. Domain: Zeno-Travel. Problem: ZTRAVEL-2-5 (17 objects), K: 8, max_cost : 15, $max\_time = 60sec$.

Input	DFAA			DFAM			DFAG		
min_dist	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.	K-set found?	Time (Sec.)	Plans gentd.
0.4	No* (6)	60.00	294	Yes	0.18	1447	Yes	0.20	1447
0.5	No* (4)	60.00	579	Yes	0.62	7884	Yes	0.44	8545
0.6	No* (3)	60.00	1171	Yes	21.46	505186	Yes	10.30	619579

problem size (in this case, $n = 30$). Note that, MCER with $n = 1$ would result in an exploration similar to that of the greedy method, with the exception of going further and guaranteeing completeness. And, MCER with $n = \infty$ would result in an exploration similar to that of ACER. Results show that DFAA performs poorly on large instances due to its exhaustive exploration. Between DFAM and DFAG, on easier problems (with low diversity/distance requirement; first 2 instances), greedy and MCER based methods fare similarly, with the greedy method slightly outperforming the MCER based method (note that, one can change the n value to 1 here, to make MCER deliver results similar to that of the greedy method, however, this is not beneficial in general). On the other hand, when the diversity required is higher (third instance), MCER based method outperforms greedy approach by finding a larger diverse set using only 12.81% of the plans generated by that of the greedy method, showcasing the advantage of exploring plan combinations more thoroughly.

Table 5 presents the comparison of DFAA, DFAM and DFAG methods on a problem from the Zeno-Travel domain with 17 objects. Once again, we observe similar results as that of the previous two domains. ACER based method fares poorly due to its exhaustive nature which limits its reach in scanning through the full space plans in the given time. And, between DFAM and DFAG, while DFAG may find the K-set in shorter time in some cases, DFAM continues to leverage the advantage of exploring plan combinations thoroughly and is able to find required K-sets using lesser number of plans. This is a crucial element in working with domains where producing individual plans itself is very difficult, which is especially the case when large problem sizes are involved.

Before we conclude, we present a note on solving large sized problems. In such cases, while completeness may not be a practical expectation, one should be able to gain performance over using the greedy approach by carefully integrating the proposed MCER approach with the state-of-the-art solvers/planners, feedback mechanisms, and using efficient heuristics for exploring the space of plan combinations. Furthermore, the proposed methods can be easily extended to solve related problems such as, finding a K-set with maximum diversity, finding largest K-set with a given diversity, finding high quality (in terms of the cost of the plans) K-sets with given diversity, and a combination thereof involving the generation of multi-objective pareto fronts.

Conclusion

In this paper, we take a combinatorial search perspective of the widely studied diverse solution generation problem. We observe that many of the approaches proposed in various domains such as automated planning and constraint satisfaction use a linear-time greedy method for exploring plan set combinations, that makes them fail while searching for a diverse set of required size even when there exists one. We propose a bi-level optimization framework and methods to find cost-sensitive diverse solutions which guarantee to find a diverse set of required size whenever there exists one. We identify the critical elements that affect the performance in such scenarios and propose efficient methods to handle them. We showcased the efficacy of the proposed methods by implementing our methods as part of the Fast Downward planning system and comparing with the existing greedy approach across various domains. The proposed methods have found larger sets of diverse solutions compared to the greedy approach on almost all problem instances, within the same time bound, proving their utility.

Acknowledgments

This research is supported in part by the ONR grants N00014-13-1-0176, N00014-13-1-0519 and N00014-15-1-2027, and the ARO grant W911NF-13- 1-0023.

References

Alexiadis, A., and Refanidis, I. 2013. Generating alternative plans for scheduling personal activities. In *Proceedings of the Seventh Scheduling and Planning Applications workshop*, 35–40.

Bloem, M., and Bambos, N. 2014. Air traffic control area configuration advisories from near-optimal distinct paths. *Journal of Aerospace Information Systems* 11(11):764–784.

Bloem, M. J. 2015. *Optimization and Analytics for Air Traffic Management*. Ph.D. Dissertation, Stanford University.

Boddy, M. S.; Gohde, J.; Haigh, T.; and Harp, S. A. 2005. Course of action generation for cyber security using classical planning. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005), June 5-10 2005, Monterey, California, USA*, 12–21.

Bryce, D. 2014. Landmark-based plan distance measures for diverse planning. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Bylander, T. 1991. Complexity results for planning. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, 274–279. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Coman, A., and Muñoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

Coman, A., and Muñoz-Avila, H. 2012a. Diverse plan generation by plan adaptation and by first-principles planning: A comparative study. In *Case-Based Reasoning Research and Development - 20th International Conference, ICCBR 2012, Lyon, France, September 3-6, 2012. Proceedings*, 32–46.

Coman, A., and Muñoz-Avila, H. 2012b. Plan-based character diversity. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-12, Stanford, California, October 8-12, 2012*.

Coman, A., and Muñoz-Avila, H. 2013. Automated generation of diverse npc-controlling fsms using nondeterministic planning techniques. In *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-13, Boston, Massachusetts, USA, October 14-18, 2013*.

Coman, A. 2012. Solution diversity in planning. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Dechter, R.; Flerova, N.; and Marinescu, R. 2012. Search algorithms for m best solutions for graphical models. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

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

Eiter, T.; Erdem, E.; Erdogan, H.; and Fink, M. 2013. Finding similar/diverse solutions in answer set programming. *Theory and Practice of Logic Programming* 13(03):303–359.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in lpg. *J. Artif. Int. Res.* 20(1):239–290.

Goldman, R. P., and Kuter, U. 2015. Measuring plan diversity: Pathologies in existing approaches and A new plan distance metric. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 3275–3282.

Hebrard, E.; Hnich, B.; O'Sullivan, B.; and Walsh, T. 2005. Finding diverse and similar solutions in constraint programming. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1*, AAAI'05, 372–377. AAAI Press.

Hebrard, E.; O'Sullivan, B.; and Walsh, T. 2007. Distance constraints in constraint satisfaction. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 106–111. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res. (JAIR)* 26:191–246.

Kambhampati, S. 2007. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, 1601–1605.

Lawler, E. L., and Wood, D. E. 1966. Branch-and-bound methods: A survey. *Operations Research* 14(4):699–719.

Myers, K. L., and Lee, T. J. 1999. Generating qualitatively different plans through metatheoretic biases. In *AAAI*, 570–576. American Association for Artificial Intelligence.

Nadel, A. 2011. Generating diverse solutions in sat. In *Proceedings of the 14th International Conference on Theory and Application of Satisfiability Testing*, SAT'11, 287–301. Berlin, Heidelberg: Springer-Verlag.

Nguyen, T. A.; Do, M. B.; Gerevini, A.; Serina, I.; Srivastava, B.; and Kambhampati, S. 2012. Generating diverse plans to handle unknown and partially known user preferences. *Artif. Intell.* 190:1–31.

Petit, T., and Trapp, A. C. 2015. Finding diverse solutions of high quality to constraint optimization problems. In *IJCAI. International Joint Conference on Artificial Intelligence*.

Roberts, M.; Howe, A.; Ray, I.; and Urbanska, M. 2012. Using planning for a personalized security agent. In *AAAI Workshops*.

Roberts, M.; Howe, A. E.; and Ray, I. 2014. Evaluating diversity in classical planning. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Russell, S., and Norvig, P. 1995. *Artificial intelligence: a modern approach*. Prentice Hall.

Rymon, R. 1992. Search through systematic set enumeration. *Technical Reports (CIS)* 297.

Scala, E. 2014. Plan repair for resource constrained tasks via numeric macro actions. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2016–2022.

Vadlamudi, S. G.; Aine, S.; and Chakrabarti, P. P. 2013. Anytime pack heuristic search. In *Pattern Recognition and Machine Intelligence - 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings*, 628–634.

Vadlamudi, S.; Aine, S.; and Chakrabarti, P. 2015. Anytime pack search. *Natural Computing* 1–20.

Vadlamudi, S. G.; Chakrabarti, P. P.; and Sarkar, S. 2012. Anytime algorithms for mining groups with maximum coverage. In *Tenth Australasian Data Mining Conference, AusDM 2012, Sydney, Australia, December 5-7, 2012*, 209–220.

Voss, C.; Moll, M.; and Kavraki, L. E. 2015. A heuristic approach to finding diverse short paths. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, 4173–4179.