Evaluating Diversity In Classical Planning*

Mark Roberts† and **Adele E. Howe** and **Indrajit Ray**
Computer Science Dept., Colorado State University
Fort Collins, CO 80524, USA
email: {mroberts,howe,indrajit}@cs.colostate.edu

Abstract

Applications that require alternative plans challenge the single solution, single quality metric assumptions upon which many classical planners are designed and evaluated. To evaluate the distinctness of alternative plans (i.e., plan sets), researchers have created diversity metrics that often measure the set difference between the actions of plans. Many approaches for generating plan sets embed the same diversity metric in a weighted evaluation function to guide the search mechanism, thus confounding the search process with its evaluation. We discover that two diversity metrics fail to distinguish similar plans from each other or to identify plans with extraneous actions, so we introduce two new diversity metrics, *uniqueness* and *overlap*, to capture these cases. We then examine the tradeoffs of producing diverse plans while we control for plan length and metric interaction and confirm that metric interaction can significantly impact search performance. We show that planners searching for plan sets must consider a third metric, *parsimony*, that prefers shorter plans while maximizing diversity.We evaluate three existing approaches for generating diverse plans and two new algorithms that are designed to explicitly manage diversity and interaction between the diversity and quality metrics. Our findings synthesize and extend recent results in plan diversity.

Evaluating Planning Systems

Search-based planning systems are often designed and evaluated under the assumptions of producing a single (optimal) plan according to a single objective function/metric. Typical performance evaluation metrics include: *coverage* over the number of problems solved, the computational *efficiency* of producing a plan, or the *quality* of the final plan. Computational efficiency usually measures CPU time or memory consumption, while plan quality is usually a one-dimensional criterion of either the plan length or the sum of a plan's action costs. This evaluative focus naturally drives planning system design, and state-of-the-art techniques improve performance along one or both of the efficiency-quality dimensions.

Recent work has extended planning systems to produce plan sets (e.g., (Srivastava et al. 2007; Coman and Muñoz-Avila 2011; Nguyen et al. 2012; Roberts et al. 2012; Khouadjia et al. 2013)). This work evaluates the quality of plan sets with a *diversity* metric, which is often measured as the distance between plans using set difference or set intersection of their actions.

To guide plan set generation, several approaches use a weighted evaluation function of the diversity metric and other quality (cost) metrics; this can confound analysis because diversity metrics are used to guide search and evaluate performance. Recent work examining the interaction of metrics and search behavior (e.g., (Cushing, Benton, and Kambhampati 2011; Wilt and Ruml 2011; Radzi 2011; Sroka and Long 2012; Roberts, Howe, and Ray 2013)), reveals that search performance is sensitive to the scale and interaction of multiple quality (cost) metrics, but non-temporal quality metrics have received less attention. Many approaches produce (diverse) temporal plans using a single cost metric (e.g., SAPA (Do and Kambhampati 2003), LPG-td (Gerevini, Saetti, and Serina 2006), COLIN (Coles et al. 2012)). Often, these temporal+cost planners embed deep reasoning to solve specific resource or time constraints at the junction of planning and scheduling. Yet, it is difficult to assess the contribution of the non-temporal metrics when they are combined with temporal concerns.

The goals of our work are to 1) tease apart guiding search for plan sets from evaluation, 2) explore diversity's interaction with non-temporal quality metrics, and 3) extend beyond single-objective non-temporal quality metrics. To this end, we create new evaluation metrics to analyze search performance. We begin with producing plan sets for a single quality metric, wherein we study four approaches to generating plan sets on five benchmark domains. We find that existing diversity metrics do not support comparison *between* plan sets produced by different approaches; so we create a metric, *overlap*, that takes the set intersection of two plan sets. Existing diversity metrics also do not characterize the distinctness of plans *within* a plan set (i.e., plans could be subsets of each other). So we create two more metrics: *uniqueness* captures the way in which plans do not subsume

*This research was partially supported by the National Science Foundation under Grant No. 0905232. We gratefully acknowledge the help of Ivan Serina, who helped us modify LPG-diffmax planner. We also thank the careful comments of the reviewers.

†Currently at the Naval Research Laboratory as an NRC Fellow.

each other and the *parsimony ratio* characterizes how much longer a given plan π_l is with respect to a "minimal plan," π_k. Two key results of this part of the study are showing that 1) quality and diversity metrics often interact, and 2) existing approaches often produce plans with low parsimony.

We next examine generating plan sets under metric interaction. We construct a new algorithm that (partially) alleviates the issues of metric scale and interaction. Instead of using a weighted evaluation function, Multi-Queue A* (*MQA*) manages each quality/diversity metric in its own queue. *MQA* drives diversity with a "parsimony queue" that first minimizes the heuristic estimate and then maximizes diversity. We will show that *MQA* often finds more unique solutions than other algorithmic approaches while maintaining good parsimony. It also better samples solutions along a quality trade-off front in some carefully constructed synthetic bicriteria problem instances.

Our study reveals that it is challenging to incorporate multiple quality metrics and diversity to drive search. The contributions of this paper are: 1) examining existing/new approaches to generating diverse alternatives according to two separate dimensions: plan diversity and plan quality; 2) presenting novel metrics for diversity that tease apart influences that had previously been entangled; 3) exploring quality-diversity tradeoffs in a synthetic domain and five prominent benchmarks; 4) presenting two new algorithms, ITA and MQA; 5) demonstrating the sensitivity of common planning algorithms to metric interaction; 6) characterizing a problem in previous approaches, i.e., parsimony; and 7) showing that a multi-queue approach, when paired with parsimony, can sometimes achieve a better spread of solutions. Our results lead us to conclude that the evaluation of a planner's performance must depend not only on the domain-independent metrics (i.e., plan length, diversity, or a linearly weighted sum of them) but also on the domain-dependent metrics.

Diversity Metrics

The diversity of a plan set, Π, can be measured in two ways: 1) *within* the individual plans of $\pi \in \Pi$, or 2) *between* two plan sets generated by distinct approaches.

Fox et al. (2006) study an online execution context, where plan stability is important to avoid large changes in the plan that could result in surprise, irritation, or wasted execution effort. They define $\mathrm{D}_{stability} = |(\pi_1 \setminus \pi_2)| + |(\pi_2 \setminus \pi_1)|$, where π_1 and π_2 are the actions for two plans. Coman and Muñoz-Avila generalize $\mathrm{D}_{stability}$ to a plan set,

$$\text{Diversity}_{\text{stability}}(\Pi) = \frac{\sum\limits_{\pi,\pi' \in \Pi} \mathrm{D}_{\text{stability}}(\pi, \pi')}{\frac{|\Pi| \times (|\Pi|-1)}{2}}. \quad (1)$$

Srivastava et al. (2007) (later also used by Nguyen et al. (2012)) normalizes the plan distance by the plan length,

$$\delta_a(\pi_1, \pi_2) = \frac{|(\pi_1 \setminus \pi_2)|}{|(\pi_1 \cup \pi_2)|} + \frac{|(\pi_2 \setminus \pi_1)|}{|(\pi_1 \cup \pi_2)|}. \quad (2)$$

Although the authors only use δ_a during search, we use it to calculate a diversity metric similar to Equation 1,

$$\text{Diversity}_{\text{norm}}(\Pi) = \frac{\sum\limits_{\pi,\pi' \in \Pi} \delta_a(\pi, \pi')}{\frac{|\Pi| \times (|\Pi|-1)}{2}}. \quad (3)$$

While Equations 1 and 3 summarize the average distance within a plan set, they have shortcomings: 1) they obscure the distinctness of plans within a plan set; and 2) they do not compare plan sets to each other. We contribute three additional diversity metrics designed to overcome these limitations: uniqueness, parsimony, and overlap.

Some plans pad a shorter plan with spurious actions while other plans are simply permutations of the actions. We capture this in a measure, **uniqueness**, that reduces the plan set to those plans that are not subsumed after removing padded and permuted plans,

$$u(\Pi) = \sum_{\pi_m, \pi_n \in \Pi, \pi_m \neq \pi_n} \begin{cases} 0, & \text{if } \pi_m \setminus \pi_n = \emptyset \\ 0, & \text{if } \pi_n \subset \pi_m \\ 1, & \text{otherwise.} \end{cases} \quad (4)$$

Uniqueness measures the efficiency with which the planner finds novel plans. But it does not capture all cases where a plan contains extraneous actions that do not produce a useful goal-directed alternative. The original plan, π_l, may contain a valid plan of $k < l$ actions, implying that π_l is not parsimonious. Although such plans can be highly diverse from others in the plan set, they often lie at the worst extreme of the efficiency-quality axes; these plans are neither efficiently produced nor of good quality. Driving search with a weighted objective function incorporating diversity can cause a planner that normally produces concise (optimal or satisficing) plans to sometimes produce plans of poorer quality with significant duplicated search effort. Or worse, the planner may produce plans containing action cycles that do not contribute to achieving the goal state.

Characterizing the extent to which a plan might contain spurious action cycles is as hard as planning itself in the worst case. Suppose we have a given plan, $\pi_l \in \Pi$, with length, $l = |\pi_l|$. We want to compute a minimal plan, $\pi_k \subseteq \pi_l$, of length, $k \leq l$. To obtain π_k, we run A* search with only the actions from π_l. However, given the number of solutions to process, we use a computational bound of 2 hours and 1GB of memory. So it is possible that search will not find π_k. For the plans where A* produces π_k, we calculate the **parsimony ratio** $s(\pi_k, \pi_l) = |\pi_k|/|\pi_l|$. We report summary statistics of this ratio over a plan set. The distribution of s estimates how often an approach finds minimal plans, and higher values are better.

Finally, we compare how complementary two approaches are to each other. To compare the plan sets *between* approaches, we assess their **overlap** as the set intersection of two plan sets, $o(\Pi_1, \Pi_2) = \Pi_1 \cap \Pi_2$. Note that plans can be distinct from each other while still being supersets of the unique plans.

Generating Plan Sets

In this section we summarize existing methods to generate plan sets, highlight two approaches of our own, and introduce our implementations (in bold) of the approaches we use in our study. Three approaches are variants of the A* algorithm, which is common in the search planning literature. The A* approaches are iterated, where each restart begins from the same initial state. This restart mechanism is justified by Richter et al. (2010), who show that restarts help avoid the dead ends or poor heuristic guidance in many planning domains.

One approach (described below as *LPGd*) is a full planning system. We implement four algorithms (*Div*, *RWS*, *ITA*, and *MQA*) on top of the LAMA-2008 planner (Richter and Westphal 2010) which won the 2008 International Planning Competition (IPC). All the algorithms employ only the successor function of LAMA using the FF heuristic function and preferred operators; landmarks are not used.

Each approach is given 10 hours and up to 4 GB memory to produce up to 1000 plans for each problem. We chose 1000 plans with the intention that the algorithms would have trouble achieving such a high number, thus partially controlling for an artificial ceiling in the results. Each algorithm/problem pair is run on a single processor from one of 48 dual quad-core Xeon 5450, 16 GB machines. Algorithms terminate at 1000 plans or at memory or time limits.

LPG-diffmax 2.0 (*LPGd*) is the version of LPG-diffmax (Nguyen et al. 2012) incorporated into our study. LPG-diffmax 2.0 extends the work of Srivastava et al. (2007) who generate diverse plans in a constraint-based planner called GraphPlan-CSP, and a local search planner called LPG-diffmax. We could not obtain the constraint-based planner for this study. Nguyen et al. (2012) extended the original effort to planning with incomplete preference models. Both planners search for k plans of at least d distance apart. They use three normalized distance metrics based on actions, states, and causal links. For each distance metric, the authors embed the diversity metric into the planners core heuristic to encourage diverse plans. They then compare the plan sets in terms of the three distance metrics, the average search cost to achieve the plans, and several preference functions. They focus on solving temporal metric and preference planning problems with large planning systems.

LPG-diffmax performs stochastic local search on planning graph subsets, which are partial plans that the authors call action graphs (Gerevini, Saetti, and Serina 2003; Gerevini and Serina 2002). At each branch in the search, the choice to add or remove an action is guided by a heuristic that rates each potential choice with respect to the current partial solution; the heuristic function for each potential action is the count of unsupported facts plus the count of actions that are mutex plus the diversity metric from Equation 2. LPG-diffmax produces plan sets of size $k = \{2, 4, 8, .., 32\}$ that are at least d distance apart.

LPGd checks for and eliminates duplicates during its search, in contrast to other approaches we examine. To fit our experimental methodology, we modified the planner to produce 1000 solutions. However, *LPGd* would sometimes terminate (due to exhausting time or memory) and fail to produce intermediate results. The authors helped us modify the planner to use an increment policy for k that starts at 20 and increments by 20 up to 1000 after each plan set is found. We use $d = 0.5$ because this appeared to be a reasonable value given previous results (Nguyen et al. 2012). The reader may note that we generate two orders of magnitude more solutions than the authors of LPG-diffmax originally studied, though it does perform quite well.

More recently, Khouadjia et al. (Khouadjia et al. 2013) wrap a population-based search mechanism around YAHSP (Vidal 2004). This planner produces diverse solutions and handles multiple quality metrics in temporal domains. We were unable to obtain the most recent version for our study, but hope to include it in future work.

Diversity-A* (*Div*) is our implementation of the algorithm from Coman and Muñoz-Avila (2011), which guides search using the heuristic, $h_{\text{d}} = \sum_{\pi \in \Pi} \text{D}_{\text{stability}}(\pi_{\text{relax}}, \Pi)/|\Pi|$, where π_{relax} is the final plan resulting from the relaxed plan heuristic. h_{d} measures the diversity of the current estimate of the best plan that can be achieved, π_{relax}, relative to the existing plan set Π found so far during the current search episode. The original heuristic h is combined with h_{d} to create the heuristic used during search, $h_{\text{new}}(\pi, \Pi) = (1-\alpha)h_d(\pi) - \alpha h(\pi)$, which is maximized in contrast to the usual practice of minimizing a given heuristic in planning. The tuning parameter, α, balances exploitation of h and exploration of more diverse plans. But the scale between h_{d} and h is not controlled, so one value could still dominate the final value of h_{new} despite α. The original diversity h is subtracted because the authors want to minimize the heuristic but maximize the diversity.

For *Div*, we attempt to maintain as much similarity as possible with theirs while also keeping the comparison with our other approaches fair. Similar to their planner, we leverage the $\text{D}_{stability}$ metric (Fox et al. 2006). We also set $\alpha = 0.7$ as was done in their original studies (Coman and Muñoz-Avila 2011), although we found that the best value for α was highly dependent on the domain and problem. The heuristic used in *Div* is: $h_{Div}(\pi, \Pi) = (2^{31}) - (1 - \alpha)h_{\text{d}}(\pi_{cur}, \Pi) + \alpha h(\pi)$. To maintain as much similarity as possible between the A* variants, we subtract h_{d} from a large constant and then add back the original heuristic so that the algorithm minimizes h_{Div}.

We were concerned about the bias of using the relaxed plan to compute both h and h_{d}. We explored using the partial plan of the current search node, π_{cur}, rather than the final estimated plan, π_{relax}, resulting from calculating the relaxed heuristic. We found that π_{relax} results in less unique solutions (i.e., worse performance) for `Depot`, `Driverlog`, and `Rover` problems without impacting the coverage. So π_{cur} is used in *Div*.

Random Walk Search (*RWS*) provides a natural baseline against which we can assess informed search because it selects actions without regard to their contribution to quality or diversity. Thus, it provides a fair way to assess where the bias of WA* and other algorithms is beneficial or harmful. Our version of *RWS* is inspired by the Arvand planner (Xie, Nakhost, and Müller 2012). *RWS* performs w walks where each walk performs p next descent probes of s steps. If an improving solution is not found after p probes then search restarts from the initial state. Our implementation of *RWS* performs $w = 1000$ walks where each walk performs $p = 7$ next descent probes of $s = 2000$ steps. Arvand also incorporated probe lengthening enhancements (Xie, Nakhost, and Müller 2012) that we do not employ. We expected *RWS* to produce more plans with greater diversity and lower quality than the other algorithms.

Iterated Tabu A* (*ITA*) augments the core A* algorithm with a Tabu list (Roberts et al. 2012). Algorithms 1 and 2 show the pseudocode for *ITA*; the main differences between Algorithm 1 and the original A* algorithm are in the addition of the Tabu list, $\mathcal{T}$ and the iteration in Algorithm 2. After each solution is found, EXTRACT-TABU-PAIRS() stores each $(state, operator)$ pair that is on the path to the solution into $\mathcal{T}$, where $state$ indicates the full state description. On the next iteration, when the EXPAND() function encounters a $(state, operator)$ pair that is already in $\mathcal{T}$, *ITA adds to* the g-value of that node an arbitrarily large constant, $gConstant$. This ensures that those nodes are prioritized later in the A* open list and thus are unlikely to be pulled off next. Our implementation of *ITA* uses $gConstant = 10^9$. There is no Tabu tenure so $\mathcal{T}$ grows as more plans are found. *ITA* is not optimal because adding *gConstant* for nodes in the Tabu list can contradict the (assumed) admissibility of the heuristic. Further this version of A* does not reopen nodes with a better g-value from the closed list when they are rediscovered from the open list. Completeness and soundness remain unaffected because the core A* algorithm is run in the first iteration of *ITA*.

We did examine a hybrid of *Div* and *ITA*, but found it did not perform well, so we exclude it from further discussion (Roberts 2013). *Div* and *ITA* implement Weighted-A* (WA*) with a weight that begins at 10.0 and decreases by the factor 0.8 after a new solution is found.

Multi-queue A* (*MQA*) replaces the single open list, $\mathcal{O}$, of *ITA* with with $\mathcal{Q}$, which is a data structure holding multiple queues. The A* SEARCH loop sees inserts, deletes, and peeks as though $\mathcal{Q}$ were a single open list.

The evidence we collect in our study, along with results from the literature, suggests that we must manage quality and diversity metrics independently from each other *during search* rather than combine them. In short, we must maintain independence of a set of T metrics that consist of the quality, heuristic, and diversity metrics. To accomplish this, we can draw inspiration from existing planners that manage separate heuristics in distinct open lists (Helmert 2006; Richter and Westphal 2010). The key idea behind Multi-queue A* (*MQA*) is to use separate queues, Q_t for each metric $t \in T$, and to sort each queue as is appropriate for that metric. We call them queues instead of open lists because, as we will show, not every queue sorts the nodes using the canonical A* f function. The use of a queue rather than an open list has implications for the completeness, optimality, and runtime guarantees of A*, and we conjecture that it may be possible to bound these in a manner similar to A^*_ϵ or Multiobjective A* (Stewart and White 1991).

Algorithm 1 SEARCH($\mathcal{P}$,$\mathcal{T}$,$\mathcal{O}$,maxSteps): π or fail

```
closed ← $\emptyset$ ; stepsTaken ← 0
while stepsTaken++ $\leq$ maxSteps do
    if $\mathcal{O} = \emptyset$ then
        return failure
    node ← $\mathcal{O}$.removeNext()
    state ← node.getState()
    if $S_g \subseteq$ state then
        return SOLUTION( node )
    if state $\notin$ closed then
        closed ← closed $\cup$ state
        $\mathcal{O} \leftarrow \mathcal{O} \cup$ EXPAND( node, $\mathcal{P}$, $\mathcal{T}$ )
```

Algorithm 2 ITA($\mathcal{P}$,numSolutions,maxSteps): Π or fail

```
$\Pi \leftarrow \emptyset$ ; $\mathcal{T} \leftarrow \emptyset$ ; $\mathcal{O} \leftarrow$ NODE( $s_o$ )
while $|\Pi| \leq$ numSolutions do
    $\pi \leftarrow$ SEARCH( $\mathcal{P}$, $\mathcal{T}$, $\mathcal{O}$, maxSteps )
    $\mathcal{T} \leftarrow \mathcal{T} \cup$ EXTRACT-TABU-PAIRS( $\pi$ )
    $\Pi \leftarrow \Pi \cup \pi$
```

The two design decisions for *MQA* are how to select across the queues in $\mathcal{Q}$ during search and how to initialize $\mathcal{Q}$. For a queue selection strategy, we employ round-robin selection as justified by work showing it is sufficient for many planning problems (Helmert 2006; Roberts, Howe, and Flom 2007; Richter and Westphal 2010).

For initialization of $\mathcal{Q}$, we set up two types of queues for our study. *Quality queues* sort nodes according to the usual f function for A*. If x is some metric we want to minimize, then a quality queue Q_x sorts according to $f_x = g_x + h_x$. A quality queue may employ a composed metric (i.e., $z = x + y$). While there may be clear justification for combining quality metrics, the case is not so clear for combining quality and diversity. *Diversity queues* should be sorted according to the principle of parsimony, which suggests preferring short plans while maximizing diversity. If D_S is a distance metric that orders states by first minimizing h and then maximizing $D_{\text{stability}}$, then a *parsimonious diversity queue*, Q_S sorts according to this distance metric. As a baseline to understand whether parsimony is useful, we also consider a queue, Q_D that maximizes $D_{\text{stability}}$ alone. We expect that employing Q_S produces more diverse plans that maintain parsimony.

The six variants of *MQA* used are indicated by their subscripts and depend on two choices: 1) whether the Tabu list of ITA is enabled (*MQA*$_T$) or disabled (*MQA*), and 2) which diversity queue is enabled. Using Q_S is denoted as *MQA*$_S$ while using Q_D is denoted as *MQA*$_D$. So, *MQA*$_{TS}$ denotes the version of *MQA* where the Tabu list is enabled and the parsimonious queue, Q_S, is included. We focus on four of the six variants in this paper and refer the reader to Roberts (2013) for further details.

	Problem			Uniqueness and Overlap						Parsimony				MQA					MQA Parsimony			
Domain	n	Avg	Total	u	Ratio	R	D	I	L	$\bar{x}$	SD	Med	Min	n	Avg	Total	u	Ratio	$\bar{x}$	SD	Med	Min
RWS														MQA_{D}								
Depot	12	376.4	*4517	*3181	0.704		0	1	262	0.74	0.106	0.75	0.34	7	671.9	4703	133	0.028	0.98	0.111	1.00	0.27
DriverLog	17	776.5	***13201**	*11011	0.834		0	0	2	0.64	0.121	0.63	0.25	11	661.2	7273	1908	0.262	0.91	0.149	1.00	0.20
Rover	14	706.6	***9893**	*6232	0.630		0	1	4	0.82	0.114	0.81	0.44	20	402.6	8052	2210	0.274	0.98	0.039	1.00	0.70
Cybersec	21	5.6	118	118	1.000		0	0	0	-	-	-	-	30	768.2	23047	843	0.037	-	-	-	-
Transport	20	734.1	*14682	*14549	0.991		0	0	19	0.74	0.096	0.74	0.37	20	969.0	19380	5743	0.296	0.97	0.057	1.00	0.52
Div														MQA_{S}								
Depot	8	231.0	1848	54	0.029	0		11	57	0.93	0.089	1.00	0.77	9	830.1	7471	329	0.044	0.97	0.049	1.00	0.63
DriverLog	12	423.6	5083	855	0.168	0		9	64	0.98	0.046	1.00	0.81	12	803.1	9637	482	0.050	0.98	0.048	1.00	0.72
Rover	20	275.3	5506	175	0.032	0		29	2	1.00	0.002	1.00	0.90	20	927.2	18544	332	0.018	0.99	0.021	1.00	0.83
Cybersec	28	143.7	4023	563	0.140	0		94	5	-	-	-	-	28	968.9	27129	340	0.013	-	-	-	-
Transport	20	576.0	11519	4356	0.378	0		10	14	0.98	0.043	1.00	0.82	20	981.0	19619	196	0.010	0.98	0.046	1.00	0.74
ITA														MQA_{TD}								
Depot	8	282.9	2263	246	0.109	1	11		140	1.00	0.018	1.00	0.72	7	928.3	6498	8	0.001	0.71	0.236	0.67	0.36
DriverLog	13	205.2	2667	375	0.141	0	9		98	0.99	0.040	1.00	0.61	12	798.1	9577	13	0.001	0.77	0.187	0.77	0.40
Rover	20	226.7	4533	314	0.069	1	29		5	1.00	0.011	1.00	0.83	20	947.2	18944	21	0.001	0.91	0.079	0.90	0.78
Cybersec	28	261.8	7331	***1562**	0.213	0	94		4	-	-	-	-	30	1000.0	30000	30	0.001	-	-	-	-
Transport	20	329.8	6595	5467	0.829	0	10		17	0.95	0.070	1.00	0.50	20	1000.0	20000	543	0.027	0.90	0.083	0.90	0.63
LPGd														MQA_{TS}								
Depot	22	819.9	**18038**	**17991**	0.997	262	57	140		0.53	0.233	0.44	0.18	8	336.0	2688	699	0.260	0.96	0.055	1.00	0.48
DriverLog	20	582.9	11657	**11650**	0.999	2	64	98		0.82	0.142	0.86	0.20	12	201.8	2422	391	0.161	0.99	0.042	1.00	0.57
Rover	20	432.2	8644	**8644**	1.000	4	2	5		0.95	0.054	0.96	0.72	20	519.2	10384	3713	0.358	1.00	0.016	1.00	0.77
Cybersec	24	48.0	1152	1152	1.000	0	5	4		-	-	-	-	28	709.9	19876	1078	0.054	-	-	-	-
Transport	20	974.9	**19498**	**19491**	1.000	19	14	17		0.73	0.165	0.76	0.16	20	292.4	5849	3995	0.683	0.96	0.069	1.00	0.46

Table 1: Selected solution counts, overlap, and parsimony results on the benchmark domains for *RWS*, *Div*, *ITA*, and *LPGd* (left) and the *MQA* variants (right). See prose for column descriptions.

Results on Benchmark Domains

We summarize our results on applying four approaches to generating plan sets for the five benchmark domains from the International Planning Competitions (IPCs). Three IPC-2002 domains were used by Coman and Muñoz-Avila (2011) : `Driverlog`, `Depot`, and `Rover`. `Driverlog` was also used in Nguyen et al. (2012). We add the Cyber-security (`Cybersec`) domain from IPC-2008 inspired by the work of Boddy et al. (2005) that identified the issue of generating alternatives. Finally, we include the seq-opt `Transport` domain from IPC-2011 seq-opt set because it is a newer "logistics" style benchmark, makes sense in a mixed-initiative setting, and includes action costs that we can use to examine plan quality in later experiments.

Uniqueness and Overlap Table 1 shows solution counts for each algorithm by domain. The *n* column displays how many problems the algorithm solved for a domain followed by *Avg*, the average number of plans generated in those domains. *Total* is how many plans each algorithm generated and *U* is the number of unique plans generated (Eq. 4); *Ratio* is $Unique/Total$. The best overall approaches are noted in bold. An asterisk indicates the best algorithmic (A* variant or *RWS*) approach. The *overlap* columns compare the overlap between approaches; the columns R, D, I, L represent the first letter of each approach studied.

LPGd solves more problems in `Depot` and `Driverlog` than the algorithmic approaches. It also generates the largest total of solutions for `Depot` and `Transport` and generates the most unique plans in every domain except `Cybersec`. Finally, it produces the highest unique ratio of the algorithms. One possible explanation for the performance gap between the algorithmic approaches and *LPGd* is that *LPGd* skips reporting of duplicate solutions during its search. Skipping duplicates during search improves the final uniqueness evaluation, but at an increased search cost for novel plans.

When comparing just the algorithmic approaches (i.e, *RWS*, *Div*, and *ITA*) to each other, *RWS* finds more plans and more unique plans for the IPC-2002 and `Transport` benchmarks than the A* approaches. This may be a consequence of its simplicity (very low memory overhead and lack of open/closed lists) so it can process more potential plans. The algorithmic approaches fail to solve as many problems as a state-of-the-art planner; this is evident by examining the n column between the algorithms and *LPGd*. This is likely due to reverting to only WA* without other search enhancements such as lazy initialization or multiple queues. For `Cybersec`, we note that the plan lengths are quite long (40–60 steps) and it is likely that *RWS* has trouble finding these longer plans.

Many plans are duplicates, and the unique-to-total ratio varies a great deal between the algorithms. *Div* has the lowest ratio of the WA* algorithms except in `Driverlog` while *RWS* has the highest ratio of all algorithms. *ITA* finds the most (unique) plans in `Cybersec`, which is expected since the algorithm was originally designed to produce alternatives for a security domain very similar to `Cybersec`. In general, *ITA* produces more unique plans than *Div*.

Overlap is generally low (or zero) suggesting that each algorithm generates unique plans not found by the others. For the algorithmic approaches, the intersection is higher among the WA* algorithms compared to *RWS*. *LPGd* has the greatest overlap with the other approaches for `Depot`, `Driverlog`, and `Transport`, which likely results from *LPGd* generating more solutions.

Diversity With respect to comparing diversity, we summarize the full set of results we obtained[1]. We compare diversity on the first 10 unique plans because this was a low enough number to ensure a fair comparison to approaches that did not produce many plans. *RWS* solution sets have higher diversity than the other algorithmic approaches except in `Cybersec`. This finding seems to contradict (Nguyen et al. 2012), who state that their randomized approach produces *less* diverse plans. Without comparing the actual solutions from the original experiments, it is difficult to determine the impact of the extra actions in the results from (Nguyen et al. 2012). But it is likely that their random approach fared worse because it included fewer extraneous actions than the standard *LPGd* approach.

None of the WA* algorithms dominate each other in diversity. The relatively poor performance of *Div* may be due to the fact that the diversity values detract from the original heuristic. The diversity values seen during search differ from those we can observe in a post-hoc analysis of complete solutions; recall that we use the partial solution π_{cur} during search. When we examine the pairwise diversity values for Π given by Equation 1, which is an average, we note that the range of values seen by *RWS* appears to be much larger than the other three algorithms. Further, there is an extreme skew of values toward the low end, which is most likely the result of duplicated plans. This suggests that 1) algorithms that produce numerous duplicate plans actually dilute the value of using diversity to guide search, and 2) a "solution Tabu" mechanism may mitigate the dilution for search algorithms employing diversity. Part of the success of *LPGd* in producing greater diversity is likely due to eliminating duplicate solutions during search.

Parsimony is a central concept to understanding our results. Plans produced by nearly every approach consistently contain cycles of actions that do not contribute to achieving the goal. For example, the `lift`/`drop` actions for `crate` form a 2-way cycle in `Driverlog` . We have strong reason to believe that even higher n-way action sequences occur within many plans of the approaches we studied. In some cases, a plan might have 10-30 extra actions.

To understand how much longer plans are, we calculate the parsimony ratio where possible. Table 1 summarizes the distributions using the sample mean ($\bar{x}$), standard deviation (SD), median (Med) and minimum (Min) ratios for the s-ratio on the solutions for which we could find a minimal plan. We can see that *LPGd* and *RWS* produce the lowest average and minimum ratios, indicating that plans often contain extraneous actions. Both of these algorithms also have the highest variance, indicating that a wider range of parsimony ratio values is seen in the solutions. As expected, the A* approaches produce minimal plans more often (higher means in the *Parsimony* column) with more reliability (lower variance). The algorithmic approaches also show a better minimum performance.

Tradeoffs in Diversity, Quality, and Efficiency

Tradeoffs are inherently about understanding the interactions between one or more axes of evaluation. To examine whether more search effort can result in higher quality or diversity, we partition the plan sets of the unique plans produced by each algorithm into cumulative intervals $i \in \{10, 25, 50, 100, 250, 500, 1000\}$. For each interval, we take the first i *unique* plans from each algorithm (i.e., we drop any duplicates until we reach i unique plans). We will summarize our observations for `Transport` domain since it has the most plans across all algorithms[2]. We lack enough unique solutions from the algorithmic approaches to draw fair conclusions for the three IPC-2002 problems, `Depot`, `Driverlog`, and `Rover`. Similarly, while the number of unique solutions for `Cybersec` is higher, few approaches create more than 200 unique solutions. Out of the 30 `Cybersec` problems, *Div* alone achieves this four times while *ITA* achieves this ten times. These low counts of unique solutions limit analysis of long-term behavior and thus should be more rigorously confirmed by future work.

In terms of diversity across the intervals of each algorithm, *RWS* and *LPGd* produce more diverse sets for 18 of the 20 problems. In five problems *Div* is close to either *RWS* or *LPGd*. In four problems, the plan sets of *Div* are as diverse as those from *RWS* or *LPGd*. *RWS* produces higher diversity plan sets much of the time, but it sacrifices quality to do so. Except for four problems, *RWS* and *LPGd* tend to establish this high diversity by the first ten plans and maintain it for the duration of the intervals. In nine of the problems, *Div* starts with low quality that increases to the 200th unique solution, where it plateaus.

In terms of quality, *LPGd* consistently produces the best quality while *RWS* produces the worst; the algorithms lie in the middle of these extremes. For all algorithms, the quality rarely improves over the initial plan and sometimes gets progressively worse. This is expected given that we are not bounding the solutions returned based on solutions we have already seen. These observations lead us to more carefully examine tradeoff of quality and diversity.

How often is high diversity paired with poor quality? To assess this tradeoff, we compute the *average* quality and *average* diversity for the first 10 unique plans of each algorithm (or fewer if the algorithm did not find 10 unique plans). Figure 1 (left) plots the average quality and diversity for all problems from `Transport`. It is evident there is a strong tradeoff between high diversity and good quality. *RWS* exhibits a broad range of quality and diversity values, *Div* tends to cluster toward the origin and *ITA* fits in between with somewhat similar tradeoff. For problems solved by at least two algorithms, this trend was seen in 19 of 20 problems in `Transport` and all problems in `Depot`, `Driverlog`, and `Rover`. Figure 1 (right) plots all the problems for `Cybersec`, where three distinct groups of quality each provide different diversity values and solutions for each algorithm are more evenly distributed.

[1] Refer to Section 4.4 of Roberts (2013) for details.

[2] The full data are presented in Appendix B of Roberts (2013).

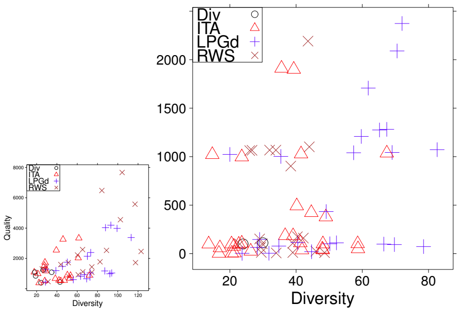

Figure 1: Demonstrating the tradeoff of domain-specific quality (lower is better) and diversity (Eq. 1, higher is better) in `Transport` (left), where the quality sums road distances plus one for each load/unload, and `Cybersec` (right), where the quality is a single action cost.

Metric Interaction and Diversity

Recent results have linked the sensitivity of search to the operator costs and metric interaction. Wilt and Ruml (2011) show that cost-based search is sensitive to the ratio of the operator costs. Cushing et al. (2011) show that cost-based search can be misled by cost functions in the actions that work against heuristic distance. Sroka and Long (2012) show that some planners are more sensitive to metric variance; e.g., MetricFF (among other planners) can generate more diverse solutions by varying the constraintedness of resources in a logistics domain. Relatively little research examines interaction between quality metrics. Radzi (2011) shows that many domains have metrics that interact with plan length in a *straightforward* way. In prior work, we examined the performance of A^*_ϵ on a synthetic domain designed to vary the kinds of interactions of two metrics (Roberts, Howe, and Ray 2013). We found that A^*_ϵ is sensitive to metric interaction and scaling.

The weighted metric, $z = x + y$, from that study provides a tool to isolate metric interaction within the context of producing plan sets. That study used a Bicriteria Synthetic Domain (`BiSyn`), where all solutions have the same plan length and two metrics, x and y, are systematically varied across the operators of the domain. `BiSyn` is essentially a fully connected $m \times n$ planning graph (see Figure 2, top) with m layers, a single initial state at $m = 0$, and a single goal at m. Each layer contains n states, and transitions between layers 0 to m are fully connected with the next layer. Two metrics, x and y, are placed on the transitions such that they interact as plotted in Figure 2 (bottom). Aside from the ran function, which is a baseline, three additional functions are studied along with their "mirrored inverses" (letters are reversed for the inverse functions): linear, lin (nil); sigmoidal, sig (gis); and polynomial, pol (lop). The functions are roughly ordered by an easy-to-hard ranking where $\{ran, lin, nil, sig\}$ show no real tradeoff while $\{gis, pol, lop\}$ exhibit a trade-off with a small number of minimal solutions. Metrics are applied in PDDL problems via the operator effects using `(increase (f) value)`. We create 21 problem variants (i.e., seven functions, three metrics) and present results on $m \times n = 15 \times 15$ problems.

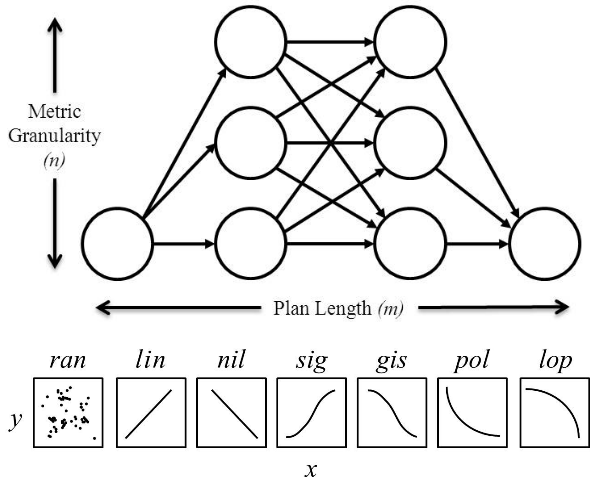

Figure 2: Example of the 3×3 `BiSyn` domain (top) and the x-y interactions (bottom) studied in Roberts et al. (2013).

Previously, we found that there is often a significant cost (10 to 100 times more) to minimizing z over just x or y except in the simplest collinear functions lin and sig. We showed that A^*_ϵ is sensitive to metric interaction (i.e., A^*_ϵ behaves differently depending on the the interaction of x and y). Finally, we showed that A^*_ϵ is sensitive to metric scale by scaling either x or y when minimizing z (i.e., they introduce functions that vary the weight of x or y by $\{2, 5, 10, 25, 50\}$ times as much, as in $z_{2x} = 2x + y$ or $z_{5y} = x + 5y$). Solution quality worsens as the scale increases along x or y. That trend is less pronounced (or absent) in the random and collinear functions (ran, lin, and sig) while evident in the remaining functions.

In this study, we run the family of *MQA* algorithms on the `BiSyn` domain. As mentioned in the description of *MQA*, the use of the Tabu list, $\mathcal{T}$, is indicated by a subscript T (i.e., *MQA*$_T$), the use of the stability queue is denoted with a subscript D (i.e., *MQA*$_D$), and the use of the parsimony queue is denoted with a subscript S (i.e., *MQA*$_S$). In the case of studying the z function of the `BiSyn` domain, we employ three different queue configurations. All three configurations evaluate the solutions according to z but vary the queues: one uses only z, the second (xy) uses x and y only, and the third (xyz) uses all three quality queues x, y, and z. Similar to A^*_ϵ, *MQA* is sensitive to metric interaction in `BiSyn`. Figure 3 shows a subset of the results of running *MQA*$_{TD}$ (top) and *MQA*$_{TS}$ (bottom) on `BiSyn`. In each, the top row is nil and the bottom row is lop; each column indicates the quality queues used. For both functions, *MQA*$_{TS}$ produces a wider variety of solutions across the frontier of possible solutions for zxy and $zxyz$. The use of a parsimony queue helps direct search to produce plan sets that better span the xy tradeoffs.

We also run the family of *MQA* algorithms on the benchmark domains. In these problems, *MQA* employs a single quality queue, where the quality metric is domain de-

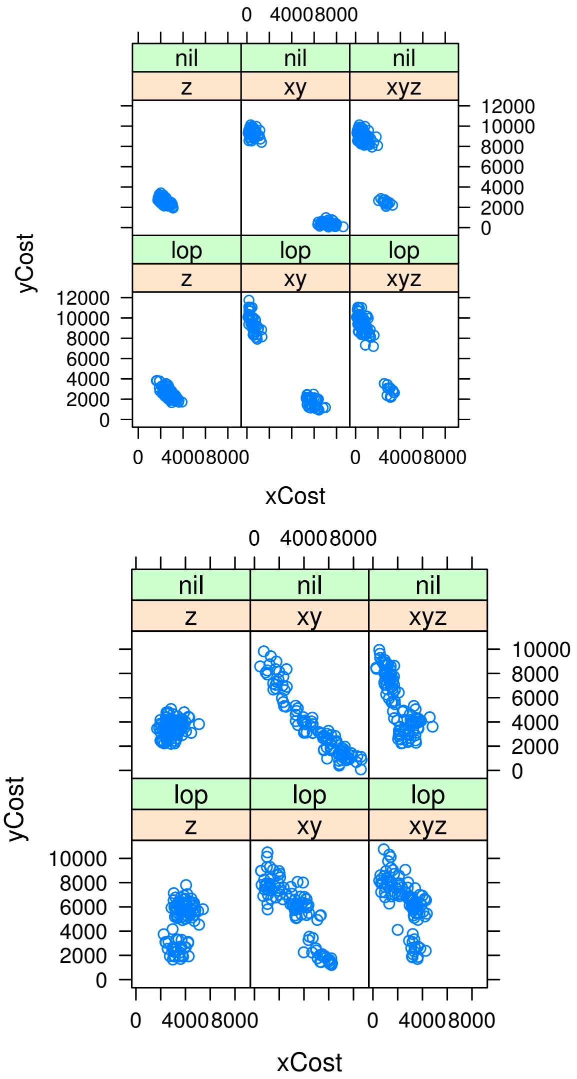

Figure 3: Results for MQA_{TD} (top) and MQA_{TS} (bottom).

pendent: plan length in the older IPC-2002 problems of `Depot`, `Driverlog`, and `Rover`, and the sum of action costs in the newer IPC-2008 `Cybersec` and IPC-2012 `Transport` problems. Table 1 (right subplot) shows the total, unique and s values for four *MQA* variants. We can make two observations from these results. First, when comparing the *MQA* variants, MQA_{TS} produces the highest number of unique solutions for all the domains except `Driverlog`, where MQA_D solved more in spite of achieving less coverage. We can also see that using the parsimony queue leads to much higher values of s with higher minimums. This suggests that embedding a parsimony queue led to better results for uniqueness and parsimony. Second, we note that *ITA* was still able to achieve better uniqueness in the security domain for which it was designed.

Summary and Future Work

We examine methods for generating plan sets in single-metric and bi-metric problems. Rather than showing that any single approach dominates, our findings highlight the strengths of each approach and directions for possible future work. The randomized approach of *RWS* produces many solutions at a low search cost with relatively high uniqueness, but at the expense of plan quality. *LPGd* performs strongly and produces the most plans and unique plans of any approach except in one domain; clearly a full planning system excels at solving many of the problems posed to it. *ITA*'s Tabu list is well suited to producing unique solutions for the security domain, `Cybersec`. The general approach of *Div* did not perform as well as other approaches according to the metrics used in this study; however we point out that this approach performs well in its original setting of incorporating domain-specific knowledge into the search process (Coman and Muñoz-Avila 2011). Using multiple queues in *MQA* seems to resolve (some) limitations of quality metric interaction, metric scale, and quality-diversity interaction.

Our results synthesize and clarify some of the work on producing plan sets. Standard practice is to find simple ways of combining multiple metrics, but that clearly has significant consequences when those metrics are correlated, uncorrelated, or competing. The tension between diversity and quality is not easily removed. Moreover, the plan length quality metric is intrinsic to the design and implementation of planners so much so that any attempt to include diversity or multiple quality metrics is in effect fighting against it. All of these have implications for other approaches to combining interacting metrics in a weighted objective function. The knowledge that quality and diversity interact with each other combined with the knowledge of how specific metric interactions impact search led us to the key insight that parsimony – preferring shorter plans – is central to maximizing diversity while maintaining reasonable quality and efficiency. Combining Q_S with the Tabu mechanism of *ITA* results in more unique *and parsimonious* plans for many problems in `BiSyn` and the benchmarks. However, this is not always the case; *ITA* still excels on the security domain for which it was originally designed.

In practical terms, these results suggest that the objectives of the application should direct the choice of the algorithm and choice of the search control mechanisms for generating plan sets. We find heuristic modifications alone do not account for the success of producing diverse plan sets. Instead, we show that parsimony must be a central concern and that using a portfolio-like multi-queue approach can lead to good results – depending on the application.

We present several lessons learned. Firstly, we find that the plan-set metrics can be misleading; this seems to correspond to the findings of (Bryce 2014), and future work should further explore this connection. Secondly, from a multi-objective perspective, it is not clear that evaluating Π with a diversity metric at end of search is always appropriate. A user may be more interested in distinct solutions that explore domain-specific quality tradeoffs than the value of a diversity metric. Future work should focus on using metrics that assess the diversity from a user-oriented perspective; the

Integrated Convex Preference function (Fowler et al. 2005) used by Nguyen et al. (2012) to handle unknown user preferences in temporal planning provides one possible direction for such analysis. Thirdly, we show that (although our study is a focused sample) the current benchmarks may be inadequate for studying plan diversity, which echos the findings of Radzi (2011). Future work should consider benchmarks that address these issues. Finally, we show that analysis-directed changes to an algorithm can lead to improved performance. However, the ubiquitous objective of minimizing plan length leads to subtle confounding factors. Future work should 1) explore alternative ways to drive diversity that do not rely so heavily on plan length and 2) examine whether using domain-dependent metrics would lead to similar results.

References

Boddy, M.; Gohde, J.; Haigh, J. T.; and Harp, S. Course of action generation for cyber security using classical planning. In *Proc., 15th Int'l Conf. on Automated Planning and Scheduling*, 12–21. Palo Alto, CA: AAAI Press.

Bryce, D. 2014. Landmark-based plan distance measures for diverse planning. In *Proc., 24th Int'l Conf. on Automated Planning and Scheduling*, to appear.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 44:1–96.

Coman, A., and Muñoz-Avila, H. 2011. Generating diverse plans using quantitative and qualitative plan distance metrics. In *Proc., 25th AAAI Conf. on Artificial Intelligence*, 946–951. Palo Alto, CA.: AAAI Press.

Cushing, W.; Benton, J.; and Kambhampati, S. 2011. Cost based satisficing search considered harmful. In *Workshop on Heuristics for Domain-Independent Planning at 21st Int'l Conf. on Automated Planning and Scheduling*.

Do, M., and Kambhampati, S. 2003. SAPA: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20:155–194.

Fowler, J. W.; Kim, B.; Carlyle, W. M.; Gel, E. S.; and Horng, S.-M. 2005. Evaluating solution sets of 'a posteriori' solution techniques for bi-criteria combinatorial optimization problems. *Journal of Scheduling* 8:75–96.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proc., 16th Int'l Conf. on Automated Planning and Scheduling*, 212–221. Palo Alto, CA: AAAI Press.

Gerevini, A., and Serina, I. 2002. LPG: a planner based on local search for planning graphs. In *Proc., 6th Int'l Conf. on Artificial Intelligence Planning Systems*, 13–22. Palo Alto, CA: AAAI Press.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20:239–290.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predicatable exogenous events. *Journal of Artificial Intelligence Research* 25:187–231.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Khouadjia, M.-R.; Schoenauer, M.; Vidal, V.; Dréo, J.; and Savéant, P. 2013. Multi-objective AI planning: Evaluating DAEyahsp on a tunable benchmark. In *Proc. of the 7th International Conference on Evolutionary Multi-Criterion Optimization*, LNCS 7811, 36–50. Sheffield, UK: Springer.

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

Radzi, M. 2011. *Multi-objective planning using linear programming*. Ph.D. Dissertation, The Univ. of Strathclyde.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In *Proc., 20th Int'l Conf. on Automated Planning and Scheduling*, 137–144. Palo Alto, CA: AAAI Press.

Roberts, M.; Howe, A. E.; Ray, I.; and Urbanska, M. 2012. Using planning for a personalized security agent. In *Workshop on Problem Solving using Classical Planners at 26th AAAI Conf. on Artificial Intelligence*.

Roberts, M.; Howe, A.; and Flom, L. 2007. Learned models of performance for many planners. In *Workshop on Learning and Planning at 17th Int'l Conf. on Automated Planning and Scheduling*.

Roberts, M.; Howe, A.; and Ray, I. 2013. A tale of 'T' metrics. In *Workshop on Heuristic Search and Domain Independent Planning 23rd Int'l Conf. on Automated Planning and Scheduling*.

Roberts, M. 2013. *A Tale of 'T' Metrics: Choosing Tradeoffs in Multiobjective Planning*. Ph.D. Dissertation, Colorado State University.

Srivastava, B.; Kambhampati, S.; Nguyen, T.; Do, M.; Gerevini, A.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence*, 2016–2022.

Sroka, M., and Long, D. 2012. Exploring metric sensitivity of planners for generation of pareto frontiers. In *Starting AI Researchers*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, 306–317.

Stewart, B. S., and White, III, C. C. 1991. Multiobjective A*. *Journal of the ACM* 38(4):775–814.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *Proc., 14th Int'l Conf. on Automated Planning and Scheduling*, 150–159. Palo Alto, CA: AAAI Press.

Wilt, C., and Ruml, W. 2011. Cost-based heuristic search is sensitive to the ratio of operator costs. In *Proc., 4th An. Symp. on Combinatorial Search*, 172–179. Palo Alto, CA: AAAI Press.

Xie, F.; Nakhost, H.; and Müller, M. 2012. Planning via random walk-driven local search. In *Proc., 22nd Int'l Conf. on Automated Planning and Scheduling*, 315–322. Palo Alto, California: AAAI Press.