Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Better Time Constrained Search via Randomization and Postprocessing

Fan Xie and **Richard Valenzano** and **Martin Müller**
Computing Science, University of Alberta
Edmonton, Canada
{fxie2, valenzan, mmueller}@ualberta.ca

Abstract

Most of the satisficing planners which are based on heuristic search iteratively improve their solution quality through an anytime approach. Typically, the lowest-cost solution found so far is used to constrain the search. This avoids areas of the state space which cannot directly lead to lower cost solutions. However, in this paper we show that when used in conjunction with a post-processing plan improvement system such as ARAS, this bounding approach can harm a planner's performance since the bound may prevent the search from ever finding additional plans for the post-processor to improve.

The new anytime search framework of *Diverse Any-Time Search* addresses this issue through the use of restarts, randomization, and by not bounding as strictly as is done by previous approaches. Below, we will show that by using these techniques, the framework is able to generate a more diverse set of "raw" input plans for the post-processor to work on. We then show that when adding both Diverse Any-Time Search and the ARAS post-processor to LAMA-2011, the winner of the most recent IPC planning competition, the performance according to the IPC scoring metric improves from 511 points to over 570 points when tested on the 550 problems from IPC 2008 and IPC 2011. Performance gains are also seen when these techniques are added to Anytime Explicit Estimation Algorithm (AEES), as the performance improves from 440 points to over 513 points on the same problem set.

Introduction

Since IPC-2008, the satisficing planning community has been using the IPC scoring function to evaluate planners. This function emphasizes both plan quality and coverage simultaneously. Many satisficing planners such as LAMA (Richter and Westphal 2010) and Fast Downward (Helmert 2006) use an *anytime approach*: they attempt to quickly find an initial plan of possibly low quality, then use the remaining time to improve upon this plan. *Post-processing*, as implemented in the ARAS system (Nakhost and Müller 2010), is another recent plan quality improvement technique. This approach takes an existing valid plan as input and tries to improve it by removing unnecessary actions and by finding shortcuts with a local search. Another *post-processing* technique, discussed in (Chrpa, McCluskey, and Osborne 2012), analyzes action dependencies and independencies in order to identify redundant actions or non-optimal sub-plans.

Originally, the ARAS post-processor was applied as the final step of the planning process, after the planner had completed or was considered unlikely to find a better plan. However, since post-processing systems decouple plan improvement from plan discovery, post-processors like ARAS can also be used in an anytime fashion: by running them with a relatively tight time or memory limit on *every* plan produced by an anytime planner. This approach has been used by several recent planners (Nakhost et al. 2011; Valenzano et al. 2012; Xie, Nakhost, and Müller 2012).

Tests with these planners, described below, show that there is large variance in the amount of improvement achieved with post-processing. As such, a lower quality input plan can often yield a higher quality final plan through post-processing than an initially better quality input plan. This behaviour can be exploited by planning systems which use post-processing. Currently, most anytime satisficing planners use the cost of the best incumbent solution to bound their future search. This avoids wasting effort in areas of the state space that cannot *directly* lead to a better solution. However, such bounding greatly decreases the number and variety of plans that an anytime system finds and we will show that because of it, bounding can have a negative impact on performance, particularly when using post-processing.

The main contributions of this paper are as follows:

- We introduce the concept of *unproductive time*, which measures the amount of time after the best solution is found, to help explain the impact of bounding.
- We present evidence that bounding in an anytime system is detrimental when used in conjunction with a post-processing system.
- We develop the meta-algorithm Diverse Any-time Search (DAS), which uses restarting to generate a more diverse set of plans.
- We implement DAS in Fast Downward (Helmert 2006) and show that it leads to significant plan quality improvements for two recent planning algorithms: the state-of-the-art planner LAMA-2011 (Richter and Westphal 2010) and AEES (Thayer, Benton, and Helmert 2012).
- We show that the improvements from DAS and from post-processing are independent, and can even be syn-

ergetic: with LAMA, the improvement from using both techniques together is slightly larger than the sum of the improvements when applied individually.

The remainder of this paper is organized as follows: after introducing the concept of *unproductive time* and measuring it in LAMA for recent IPC planning problems, the new meta-algorithm DAS is introduced. DAS is tested both with and without the ARAS post-processor on LAMA-2011 and AEES. The experimental results show strong improvements in plan quality on IPC-2008 and IPC-2011 domains.

Unproductive Time in Any-time Satisficing Planning

As mentioned above, most state-of-the-art planners use an anytime strategy to improve solution quality over time. However, there has been little investigation into how much time is actually being used to find the final solution. Let *unproductive time* be defined as the amount of time remaining, out of the total time given, when the planner's best solution is found. For example, if an anytime planner A finds its best solution on a planning instance B at 13 minutes given a 30 minute time limit, and A does not improve upon this plan in the remaining 17 minutes, then the unproductive time for planner A on problem B is 17 minutes. The amount of unproductive time can be used to evaluate how efficiently an anytime planner is using the given search time, since that unproductive time could be spent doing something more useful, such as plan post-processing. Below, we will show that one of the consequences of using bounding in an anytime system such as LAMA-2011 is that it often leads to large amounts of unproductive time.

LAMA-2011 is a state-of-the-art planner that has been shown to achieve both high coverage and strong solution quality. It won the sequential satisficing track of the International Planning Competition in 2011 (IPC 2011) after, in its previous incarnation as LAMA-2008, it won the same track at IPC 2008. LAMA's high coverage is achieved through the use of multiple heuristics (Richter, Helmert, and Westphal 2008), preferred operators (Richter and Helmert 2009), and greedy best-first search (Bonet and Geffner 2001). LAMA-2011 starts its search with two runs of greedy best-first search: first with a distance-to-go heuristic and then with a cost-to-go heuristic. Next, LAMA improves the quality of its solutions through the anytime procedure of Restarting Weighted A* (RWA*) (Richter, Thayer, and Ruml 2010). This procedure starts a new WA* search with a lower weight w whenever a new best solution is found. Only cost-to-go heuristics are used in this phase.

Whenever a new best solution with cost C is found, this cost is used to *bound* the rest of the search. This means that only nodes with g-cost (cost of best known path to the node) less than C are added to the open list. This prunes states that cannot lead directly to a better solution than before. Figure 1 shows that this approach also leads to a very large fraction of unproductive time on IPC benchmarks. Among the total of 244 problems solved in IPC-2011 with an 1800 second (30 minute) time limit, in more than 45% (111) of the problems, LAMA-2011 is unproductive for more than 1700 seconds. Table 1 shows the amount of unproductive time separately for each IPC-2011 domain. In the four domains of 2011-barman, 2011-elevators, 2011-parcprinter and 2011-woodworking, unproductive time exceeds 90%. In these domains, the planner is able to quickly find an initial solution, but fails to improve upon it.

As a typical example, Table 2 shows the number of solutions and the amount of unproductive time for the 20 instances of 2011-elevators. With the exception of instance 04, LAMA-2011 finds only a single solution to each problem. This does not at all imply that the first solution found by LAMA-2011 is optimal. Subsequent postprocessing with ARAS yields improved solutions for these problems. In these cases, it is much more difficult to find a second solution when using cost-to-go heuristics and the bound from the first solution, than to generate the initial solution using distance-to-go heuristics with no bound.

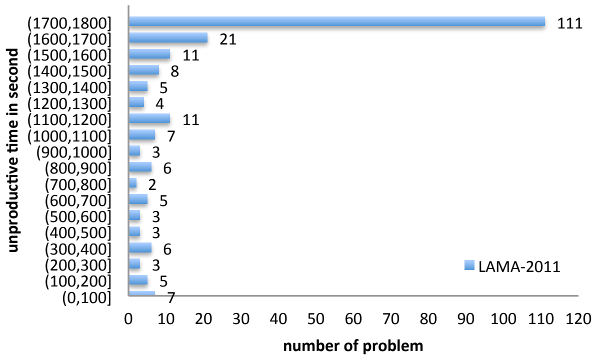

Figure 1: *Unproductive Time* of LAMA-2011 on IPC-2011

Domain	solved	total time	UT	percentage
parcprinter	20	36000	35997	99.99%
barman	20	36000	35901	99.73%
woodworking	20	36000	35357	98.21%
elevators	20	36000	33890	94.14%
visitall	20	36000	31280	86.89%
pegsol	20	36000	31107	86.41%
scanalyzer	20	36000	29844	82.90%
nomystery	10	18000	14682	81.57%
transport	15	27000	20554	76.13%
openstacks	20	36000	24902	69.17%
floortile	6	10800	7032	65.11%
tidybot	16	28800	18099	62.84%
sokoban	19	34200	20377	59.58%
parking	18	32400	13948	43.05%

Table 1: *Unproductive Time* (UT) of LAMA-2011 on different IPC-2011 domains.

Post-Processing with ARAS

Since even a state-of-the-art planner such as LAMA-2011 spends the large majority of its execution time being unproductive, a natural question becomes: How to use this time more effectively?

Instance	#s	UT	Instance	#s	UT
01	1	1799	11	1	1760
02	1	1798	12	1	1704
03	1	1797	13	1	1679
04	3	1794	14	1	1725
05	1	1799	15	1	1659
06	1	1796	16	1	1649
07	1	1798	17	1	1576
08	1	1791	18	1	1510
09	1	1789	19	1	1152
10	1	1783	20	1	1531

Table 2: Number of solutions (#s) and *Unproductive Time* (UT) of LAMA-2011 in the 20 instances of 2011-elevators.

One possibility is to feed the solutions found by a planner into a post-processing plan improvement system such as ARAS (Nakhost and Müller 2010). ARAS consists of two components. The first, Action Elimination (AE), examines the plan for unnecessary actions. This involves scanning the plan from front to back, removing each action and its dependent actions in turn, and testing if the resulting plan is still valid and reaches a goal. If so, the unnecessary actions are discarded and the scan continues on the resulting shorter plan. The second and main component of ARAS is Plan Neighbourhood Graph Search (PNGS). This technique builds a neighbourhood graph in the state space around the trajectory of the input plan by performing a breadth-first search to some depth d. Once this plan neighbourhood is constructed, a lowest-cost plan from the start to a goal state is extracted from this graph. Intuitively, the algorithm identifies local shortcuts along the path.

Typically, ARAS is run with a time and memory limit. Until the first of these limits is reached, AE and PNGS are alternated (starting with AE). PNGS is run iteratively, with an increasing depth bound. The best plan found within the limits is returned.

When ARAS was first introduced, it was used in a *process-once* fashion. This involved splitting the available planning time into two phases: plan finding and plan post-processing. In the first phase, a planner is used to find some plan (or a series of plans). When the plan finding phase is up, the best plan found so far is handed to the post-processor which is then allotted all of the remaining time. Several IPC-2011 planners (Nakhost et al. 2011; Valenzano et al. 2012) used ARAS in an anytime fashion. This involves interleaving the plan finding and post-processing phase by using the post-processor in between iterations of an anytime planner. Each new plan found is immediately post-processed by ARAS. When ARAS hits one of its limits, it returns the best solution it has found and allows the anytime planner to continue looking for more plans.

There are two motivations for using ARAS in this way. First, for most anytime satisficing planners, it is difficult to determine if any new plans will be found during the remaining time or if the planner will be unproductive. Second, even if the anytime planner can find a better plan with more search, there is no guarantee that post-processing the best plan found by an anytime planner will have a lower cost than the post-processed version of a weaker plan found earlier. The following experiment illustrates this. There are 151 problems from IPC-2011 for which LAMA-2011 generates more than one plan (without any assistance from a post-processor). If ARAS is run with a 2 GB memory limit on all these plans, then on 44 (29%) of these problems, the lowest cost solution is generated by post-processing an earlier plan, not by post-processing the final and best input plan found by LAMA-2011. The instance 2011-transport07 is typical: LAMA-2011 generates two plans of cost 8396 and 7222. Post-processing with ARAS improves the weaker input plan to 6379, and the stronger one to 6402. The weaker initial plan leads to the better final result.

This suggests that generating a greater diversity of input plans can be important for post-processing. If the anytime planner is only able to generate a small number of similar plans, then ARAS can only search within a very restricted set of neighbourhoods. This likely means that all its output plans will be similar, and of similar quality. When ARAS is handed a larger and more diverse set of input plans, it has a greater chance of finding significant improvements for at least one of them.

The Diverse Any-time Search Meta-Algorithm

The *Diverse Any-time Search (DAS)* meta-algorithm, shown in Algorithm 1, uses restarts with no bounds and post-processing in order to improve a given anytime planner. DAS divides the given total planning time into N equal time segments, where N is a user-supplied constant. In the first segment, the anytime planner P is run normally, except that ARAS is used to post-process each plan generated by P. At any time during this first segment, P can use its best plan found so far, *excluding* post-processing, to bound its search.

When time runs out on the first time segment, the best plan found so far, *including* post-processing, is saved. The anytime planner is restarted without any knowledge of previous solutions, and with a planner-specific randomization which will vary the planner's search. At the end of each search segment, the best overall plan is updated, and P restarts from scratch with a new random seed.

As the early, greedier iterations of P typically find plans much more quickly and frequently, by restarting from scratch (and thus performing these greedy iterations again), DAS increases the number of plans available to the post-processor. This increases the number of opportunities for finding a strong improvement. In our experiments, if P cannot find any solution by the end of the first time segment, then it is does not restart. This ensures that the coverage of the planners using DAS is the same as those that do not use it, and so we can directly focus on the impact that DAS and other anytime approaches have on plan quality. These details are included in Algorithm 2. The algorithm uses two time limits: a soft time limit t for each segment, used for restarting if at least one solution was found, and a hard time limit T for the whole search. If in the first segment, P cannot find a solution within the soft time limit t, then restarts are disabled. In Algorithm 2, in the assignment $\langle p1, t1\rangle \leftarrow P.search(conf, bound, rand, time)$, *conf* is the current search configuration (such as the weight for RWA*), *bound* is the plan quality bound, and *rand* is the random seed. If the search finds a solution within time limit $time$, it returns the plan $p1$ and time used $t1$. Otherwise, the search terminates with $p1 = NULL$ and $t1 = time.$

The expression $\langle p2, t2\rangle \leftarrow PP.process(p1, time)$ indicates a call to the post-processor *PP* with $p1$ as the input plan. The post-processor *PP* returns when it either reaches the $time$ limit or a pre-set memory limit. When PP terminates because of the time limit, it returns the best plan found and $t2 = time$. When it terminates because of memory, it returns the best plan found and the time used. In each case, the best plan returned could still be the input plan $p1$ if no improvements are found.

Algorithm 1 Diverse Any-time Search

Input Initial State I, goal condition G given search time T, planner P, post-processor PP
Parameter N
Output A solution plan

```
$t \leftarrow T/N$
$\langle plan_{best}, cost_{best}\rangle \leftarrow \langle [], \infty\rangle$
for $i$ from 1 to $N$ do
    $rand \leftarrow generate\_random\_seed()$
    $isSolved \leftarrow i == 1$
    $plan \leftarrow AnytimeSearchWithPostprocessing$
    $(I, G, T, t, rand, P, PP, isSolved)$
    if $cost(plan) < cost_{best}$ then
        $\langle plan_{best}, cost_{best}\rangle \leftarrow \langle plan, cost(plan)\rangle$
    end if
end for
return $plan_{best}$
```

Experiments

In this section, we describe five sets of experiments which show the utility of DAS and help enhance our understanding of this meta-algorithm. We begin with experiments which show that bounding can be harmful when used in conjunction with a post-processor. This is followed by a look at the performance of DAS without postprocessing in LAMA. We then experiment with combining DAS in LAMA with the ARAS system. The fourth set of experiments then looks at the impact of parameterization on DAS. Finally, we show that DAS is also effective when added to AEES.

All experiments in this section were run on an 8 core 2.8 GHz machine with a time limit of 30 minutes and memory limit of 4 GB per problem. Unless otherwise noted, the test set is made of all 550 problems from IPC-2008 and IPC-2011. Results for planners which use randomization are averaged over 5 runs (unless otherwise noted). All planners are implemented on the same version of the Fast Downward code base (Helmert 2006) and so the translation from PDDL to SAS+ is not included against the time limit since it is the same for all planners. For the first four experiments, the scores shown use the IPC metric with LAMA-2011 as the baseline. This means that if L is the plan found

Algorithm 2 Anytime Search with Post-processing

Input Initial State I, goal condition G, hard timelimit T, soft timelimit t, random seed $rand$, planner P, post-processor PP, first solution found flag $solved$
Parameter N
Output A solution plan

```
$conf \leftarrow P.GetFirstConf()$
$bound \leftarrow \infty$
$plan_{best} \leftarrow []$
$isSolved \leftarrow solved$
$totalTime \leftarrow 0$
$restart \leftarrow true$
while have time do
    if not $isSolved$ or not $restart$ then
        $time \leftarrow T - totalTime$
        $\langle p1, t1\rangle \leftarrow P.search(conf, bound, rand, time)$
        if not $isSolved$ and $t1 > t$ then
            $restart \leftarrow false$
        end if
    else
        $time \leftarrow t - totalTime$
        $\langle p1, t1\rangle \leftarrow P.search(conf, bound, rand, time)$
    end if
    if $p1 == []$ then
        return []
    end if
    $isSolved \leftarrow true$
    $totalTime \leftarrow totalTime + t1$
    $conf \leftarrow P.nextConf(conf)$
    $bound \leftarrow cost(p1)$
    if $restart$ then
        $time \leftarrow t - totalTime$
    else
        $time \leftarrow T - totalTime$
    end if
    $\langle p2, t2\rangle \leftarrow PP.process(p1, time)$
    $totalTime \leftarrow totalTime + t2$
    if $cost(p2) < cost(plan_{best})$ then
        $plan_{best} \leftarrow p2$
    end if
end while
return $plan_{best}$
```

by LAMA-2011, then the score of a given plan P is given by $cost(L)/cost(P)$. For the post-processor, we use the ARAS system with a 2 GB memory limit.

Experiment 1: Using Post-Processing and Bounding in LAMA-2011

Table 3 compares three planners on IPC-2011 domains:

- **LAMA-2011** is the IPC-2011 version of LAMA.
- **LAMA-Aras** is an implementation of *Diverse Any-time Search (DAS)* with the input planner being *LAMA-2011*, the input post-processor being ARAS, and *N=1*. This means that there are no restarts, and that the improved

Domain	LAMA-2011	LAMA-Aras	LAMA-Aras-B
barman	20	**23.99**	**23.99**
elevators	20	**26.01**	25.87
floortile	6	**6.77**	**6.77**
nomystery	**10**	**10.00**	9.83
openstacks	**20**	19.98	19.89
parcprinter	20	**20.10**	19.78
parking	18	**18.93**	17.46
pegsol	**20**	**20.00**	**20.00**
scanalyzer	20	**23.35**	21.26
sokoban	19	**20.23**	19.05
tidybot	16	**16.77**	**16.77**
transport	15	**17.70**	16.38
visitall	20	**20.45**	20.37
woodworking	20	**20.96**	20.85
Total	244	**265.24**	258.27

Table 3: Plan Quality of LAMA-2011, LAMA-Aras and LAMA-Aras-B on IPC-2011.

plans found by ARAS are *not* used for bounding, but LAMA-2011 *still* does its own bounding internally.

- **LAMA-Aras-B** is like LAMA-Aras except all plans, including the improved plans found by Any-time ARAS, are used to bound the subsequent iterations of WA*.

Table 3 shows that combining bounding with post-processor can be harmful, since LAMA-Aras dominates LAMA-Aras-B in almost all domains. Among the three planners, LAMA-Aras almost always gets the best score, except by a small margin in openstacks and tidybot. ARAS is known to be ineffective on openstacks problems (Nakhost and Müller 2010), and the time wasted running it causes a slight decrease in plan quality in that domain.

Experiment 2: DAS without Postprocessing

This section examines the impact of using Diverse Any-time Search in terms of unproductive time and the number of solutions it generates when used with LAMA-2011. These tests do not use any post-processing. They show that the new meta-algorithm increases the number of plans found and even improves solution quality in a number of domains.

When DAS is added to LAMA-2011, RWA* is being used within each time segment (as is bounding). When a time segment ends, RWA* starts again from scratch with a greedy best-first search iteration, though it does not bound using information from previous time segments. The source of diversity is *random operator ordering* (Valenzano et al. 2012). This involves randomly shuffling the order of the generated set of successors of an expanded node before they are added to the open list. Random operator ordering affects the search by changing how ties are broken. However, to ensure that competing algorithms have the same coverage, we use the default operator ordering during the first time segment.

We refer to the new planner as **Diverse-LAMA(N)**. **N** is the parameter which affects the length of the time segments. Table 4 compares this planner, setting $N = 4$, with LAMA-2011 on the 2011-elevators domain. The new planner exhibits much less unproductive time[1]. In particular:

- The average number of plans increases from 1.1 to 4.3. On 17 of the problems, we see an increase from 1 plan with standard LAMA-2011 to 4 plans with Diverse-LAMA(4), since Diverse-LAMA(4) finds one plan per segment. In the cases of elev02 and elev03, Diverse-LAMA(4) sometimes finds more than 1 plan per segment, depending on the random seed, whereas there is a segment in which no plan is found on elev05.
- The amount of unproductive time decreases in all but one instance (elev11), often drastically. The problem elev11 is the only exception as **Diverse-LAMA(4)** is unable to improve the plan it finds during the first segment when it finds the same plan as LAMA-2011. However, in all other problems there was at least one plan found in a later segment that was better than the first plan found.

Due to LAMA-2011's high amount of unproductive time in this domain, Diverse-LAMA(4) is also often able to find better solutions. This is because LAMA-2011 rarely finds a new solution after the first $1800/4 = 450$ seconds. In contrast, Diverse-LAMA(4) continues to find solutions by restarting and returning to a greedier search, some of which are better than the solutions found in the first 450 seconds.

Table 5 compares the plan quality of LAMA-2011 and Diverse-LAMA(4) on IPC-2011 domains and shows that this behaviour is also not limited to the elevators domain. In total, Diverse-LAMA(4) improves by a score of 6.1 though this improvement is not uniform over all domains. Instead, Diverse-LAMA(4) improves its solution quality over LAMA-2011 in 8 domains, while it is worse in 5 domains. These improvements are mainly made in domains in which LAMA-2011 has a high percentage of *unproductive time* for the same reasons as was the case in the elevators domain. However, in those domains in which LAMA-2011 is more productive later in the search, the restarts prevent Diverse-LAMA(4) from following through on one search long enough to find the best solutions. This is more apparent in Table 6, which shows the number of problems on which each of LAMA-2011 and Diverse-LAMA(4) found the best plan. In those domains in which LAMA-2011 is mostly unproductive, Diverse-LAMA(4) rarely generates worse final solutions, while for those domains in which LAMA-2011 is more productive later on — such as Floortile, Tidybot, Sokoban and Parking — Diverse-LAMA(4) will occasionally find weaker plans.

Experiment 3: Combining DAS with ARAS

This section tests the DAS when used with LAMA and ARAS. The system is denoted as Diverse-LAMA-Aras(N). The four planners **LAMA-2011**, **LAMA-2011-Aras**, **Diverse-LAMA(4)**, and **Diverse-LAMA-Aras(4)** are tested on all 550 problems from IPC-2008 and IPC-2011.

Table 7 shows a comparison of the plan quality of these planners in each of the domains tested. Diverse-LAMA-Aras(4) is the best (or tied for best) in 18 of the 23 domains

[1]Here, we show only one run instead of the average over 5 runs.

Instance	#s1	UT1	#s2	UT2	Instance	#s1	UT1	#s2	UT2
elev01	1	1799	4	1350	elev11	1	1760	4	1762
elev02	1	1798	6	330	elev12	1	1704	4	863
elev03	1	1797	4	1349	elev13	1	1679	4	354
elev04	3	1794	9	835	elev14	1	1725	4	793
elev05	1	1799	3	900	elev15	1	1659	4	713
elev06	1	1796	4	446	elev16	1	1649	4	795
elev07	1	1798	4	448	elev17	1	1576	4	1262
elev08	1	1791	4	895	elev18	1	1510	4	294
elev09	1	1789	4	1342	elev19	1	1152	4	1009
elev10	1	1783	4	440	elev20	1	1531	4	296

Table 4: Number of solutions and *Unproductive Time* of LAMA-2011 (#s1 and UT1) and Diverse-LAMA(4) (#s2 and UT2) in the 20 instances of 2011-elevators.

domain	UT	LAMA-2011	Diverse-LAMA(4)
2011-parcprinter	99.99%	20	**20.08**
2011-barman	99.73%	20	**21.76**
2011-woodworking	98.21%	20	**20.48**
2011-elevators	94.14%	20	**25.20**
2011-visitall	86.89%	20	**20.10**
2011-pegsol	86.41%	**20**	19.79
2011-scanalyzer	82.90%	20	**20.75**
2011-nomystery	81.57%	**10**	**10**
2011-transport	76.13%	15	**15.69**
2011-openstacks	69.17%	20	**20.22**
2011-floortile	65.11%	**6**	5.05
2011-tidybot	62.84%	**16**	15.30
2011-sokoban	59.58%	**19**	18.59
2011-parking	43.05%	**18**	17.21
total		244	**250.10**

Table 5: Plan Quality of LAMA-2011, Diverse-LAMA(4) on IPC-2011. Domains are sorted by decreasing fraction of *Unproductive time* (**UT**) shown in Table 1.

and achieves the highest overall score, improving over the baseline planner LAMA-2011 by **59** units.

Figure 2 shows the normalized score curve over 30 minutes of the 4 tested planners over all test domains. The three vertical lines indicate the restart points for Diverse-LAMA(4) and Diverse-LAMA-Aras(4) of 450, 900 and 1350 seconds. Notice that the time axis is in log scale. Before the first restart, DAS and non-DAS versions of the same planner are nearly the same[2]. Immediately after the first restart, the DAS planners show a quick jump in solution quality. This is because for many problems, restarting allows the planner to find new, sometimes better solutions. A similar but less pronounced jump is also visible after the second restart.

By producing more plans, Diverse-LAMA-Aras(4) also provides more input plans for ARAS. Compared to Diverse-LAMA-Aras(4), the improvement from the first restart is more pronounced in Diverse-LAMA(4). Using ARAS smoothes out some of the variance in solution quality between different runs.

[2]To fully utilize our computational resources, we run several processes simultaneously on a multi-core machine. While all versions use the same memory limit, the restarts cause DAS to use less memory. The resulting decrease in memory contention accounts for the small differences in planner performance.

domain	UP	better	worse	total
2011-parcprinter	99.99%	**2**	0	20
2011-barman	99.73%	**19**	0	20
2011-woodworking	98.21%	**8**	0	20
2011-elevators	94.14%	**19**	0	20
2011-visitall	86.89%	**6**	3	20
2011-pegsol	86.41%	0	**1**	20
2011-scanalyzer	82.90%	**6**	2	20
2011-nomystery	81.57%	**0**	**0**	10
2011-transport	76.13%	**6**	0	15
2011-openstacks	69.17%	**4**	**4**	20
2011-floortile	65.11%	0	**2**	6
2011-tidybot	62.84%	2	**7**	16
2011-sokoban	59.58%	1	**4**	19
2011-parking	43.05%	**7**	4	18

Table 6: Plan Comparison between Diverse-LAMA(4) and LAMA-2011 on different domains. The columns **better** indicates in how many problems Diverse-LAMA(4) generates better plans than LAMA-2011 (**worse** means how many worse). Domains are ordered according to the percentages of *Unproductive time* (**UP**) shown in Table 1.

As shown by the previous two experiments, using either ARAS or DAS improves plan quality. In Table 8, we show that the performance improvements from these techniques are independent, and sometimes even synergetic. The score for each planning system is split into two components: the raw scores of the best plans produced by the planning system ignoring the impact of ARAS (*ie.* ARAS is being run, but the plans it outputs are not counted towards the score of the planner), and the independent contribution of ARAS when it is used in the planning system. For comparison, the scores of the baseline planners (*ie.* those that do not run ARAS at all) are also shown. The raw scores are slightly worse than the baseline scores, since the planner producing the raw scores uses less time for the main search because of using ARAS (though it is not counted in the score). The improvement of Aras over Diverse-LAMA(4) is slightly larger than the improvement of Aras over LAMA-2011. This demonstrates that the improvements from Diverse-LAMA(4) finding substantially different overall plans seems to be largely independent from the local plan improvements found by Aras. The following two examples help to explain this behaviour:

- In 2011-floortile 05, the best raw plan generated by Diverse-LAMA(4) has cost 132, while LAMA-2011 can find a cost 63 plan. ARAS can improve the cost 132 plan to a cost of 63 as well. This suggests that ARAS can help DAS in cases where restarting prevents the search from running long enough to find good plans.
- In 2011-woodworking 01, plans of cost 1600, 1630 and 1620 are produced in time segments 1, 3 and 4, while LAMA-2011 only finds one solution of cost 1600. ARAS improves the three solutions as follows: 1600 $\rightarrow$ 1460,

1630 → 1290 and 1620 → 1380. The worst input plan is easiest to improve, while the best input plan becomes the worst after post-processing. Out of the 244 problem instances solved by LAMA-2011, in 44 cases the best final plan produced from Anytime Aras does not come from LAMA's best plan. In Diverse-LAMA-Aras(4), this ratio increases dramatically, to 86/244 **problems**. This demonstrates how increased plan diversity from DAS can improve overall performance.

domain	LAMA	DL(4)	LAMA-Aras	DL-Aras(4)
08-cybersec	**30**	**30.00**	**30.00**	**30.00**
08-elevators	30	35.82	38.50	**43.34**
08-openstacks	30	30.25	29.97	**30.35**
08-parcprinter	30	30.00	30.09	**30.10**
08-pegsol	**30**	29.78	**30.00**	**30.00**
08-scanalyzer	30	31.35	34.00	**34.18**
08-sokoban	**28**	27.15	27.66	27.48
08-transport	29	31.51	35.14	**36.73**
08-woodworking	30	30.92	33.10	**34.28**
11-barman	20	21.76	23.99	**24.20**
11-elevators	20	25.20	26.01	**31.16**
11-floortile	6	5.01	**6.77**	**6.77**
11-nomystery	**10**	**10.00**	**10.00**	9.89
11-openstacks	20	**20.22**	19.98	20.11
11-parcprinter	20	20.08	**20.10**	20.05
11-parking	18	17.21	18.93	**19.54**
11-pegsol	20	19.79	20.00	**20.01**
11-scanalyzer	20	20.75	23.35	**23.46**
11-sokoban	19	18.59	20.23	**20.54**
11-tidybot	16	15.21	**16.77**	16.17
11-transport	15	15.69	17.70	**19.68**
11-visitall	20	20.10	20.45	**20.53**
11-woodworking	20	20.48	20.96	**21.78**
total	511	526.89	553.69	**570.35**

Table 7: Plan Quality of LAMA-2011 (LAMA), LAMA-2011-Aras (**LAMA-Aras**), Diverse-LAMA(4) (**DL(4)**) and Diverse-LAMA-Aras(4) (**DL-Aras(4)**) on all 550 problems from IPC-2008 and IPC-2011. Extra experiments data can be found: *www.cs.ualberta.ca/research/theses-publications/technical-reports/2013/TR13-02*.

Experiment 4: Testing DAS with Different Numbers of Segments

Recall that DAS is parameterized by the number of segments, N, for which it runs, with $N = 1$ corresponding to LAMA-ARAS, and $N = 4$ to the algorithm used in the previous experiments. Figure 3 shows how the behaviour of this meta-algorithm changes when varying N in the range from 1 to 120 on IPC 2011 domains. Overall, the score differences are small, with the best results for N from 3 to 6. For $N \leq 3$, plan quality increases with N, taking advantage of the diversity and number of plans generated. For $N > 6$, the solution quality slowly decreases as the runtime for both LAMA and ARAS becomes ever shorter.[3] The trade-off is that a too large N does not leave the planner enough time to find high quality input plans, while small N hurt diversity.

[3]The time of each time segment is given by T/N, where T is the total time (30 minutes in our experiments).

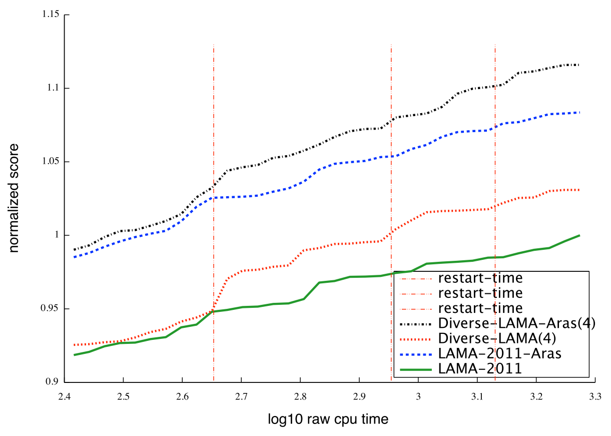

Figure 2: Normalized Score Curve of the 4 tested planners.

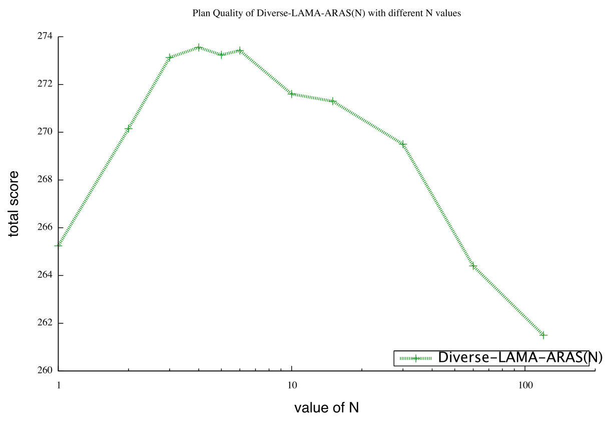

Figure 3: Plan Quality of Diverse-LAMA-Aras(N) with different N values.

Experiment 5: Combining DAS with Anytime Explicit Estimation Search

Anytime explicit estimation search (AEES) is an anytime search algorithm introduced by Thayer, Benton, and Helmert (2012). AEES uses explicit estimation search (EES) (Thayer and Ruml 2011) as its main search component. EES is a sub-optimal search algorithm which is able to use both inadmissible and admissible heuristics while still satisfying a given solution cost bound. It does so by focusing its search on nodes that the inadmissible heuristic estimates will lead to solutions that are within the bound. It has been shown to be particularly effective in domains with non-unit action costs due to its ability to use both distance-to-go and cost-to-go heuristics (Thayer and Ruml 2011). AEES is the anytime version of EES, which lowers the sub-optimality bound whenever a new best solution is found, by using the latest solution as the new bound. The AEES algorithm's goal is

domain	LAMA	DL(4)	LAMA-Aras			DL-ARAS(4)		
			raw$_{LA}$	Δ_{ARAS}	Final	raw$_{DL}$	Δ_{ARAS}	Final
barman	20.00	21.76	20.00	3.99	23.99	21.70	2.51	24.20
elevators	20.00	25.20	20.21	5.80	26.01	24.82	6.35	31.16
floortile	6.00	5.01	5.21	1.55	6.77	4.67	2.10	6.77
nomystery	10.00	10.00	10.00	0.00	10.00	9.86	0.03	9.89
openstacks	20.00	20.22	19.98	0.00	19.98	19.94	0.17	20.11
parcprinter	20.00	20.08	20.00	0.10	20.10	20.00	0.05	20.05
parking	18.00	17.21	16.84	2.09	18.93	16.33	3.21	19.54
pegsol	20.00	19.79	19.57	0.43	20.00	17.65	2.36	20.01
scanalyzer	20.00	20.75	18.89	4.46	23.35	20.39	3.07	23.46
sokoban	19.00	18.59	16.58	3.65	20.23	16.10	4.44	20.54
tidybot	16.00	15.21	15.15	1.62	16.77	14.75	1.42	16.17
transport	15.00	15.69	14.00	3.70	17.70	16.32	3.37	19.68
visitall	20.00	20.10	20.00	0.45	20.45	20.06	0.47	20.53
woodwork	20.00	20.48	20.00	0.96	20.96	20.48	1.30	21.78
Total	244.0	250.1	236.44	28.80	265.24	243.06	30.84	273.90

Table 8: Combined effect of DAS and post-processing in IPC-2011 domains.

domain	AEES	DE(4)	AEES-Aras	DE-Aras(4)
08-cybersec	29	31.20	29.00	**32.36**
08-elevators	30	33.80	41.67	**45.20**
08-openstacks	30	**31.35**	30.00	31.15
08-parcprinter	25	25.16	25.70	**25.82**
08-pegsol	30	29.96	**30.11**	30.05
08-scanalyzer	30	30.53	**34.35**	34.16
08-sokoban	**27**	26.47	26.71	26.73
08-transport	28	31.09	38.43	**40.86**
08-woodworking	20	20.25	21.14	**21.32**
11-barman	20	20.88	22.74	**22.85**
11-elevators	19	23.03	25.03	**28.06**
11-floortile	**6**	5.50	**6.00**	**6.00**
11-nomystery	**10**	9.94	**10.00**	9.91
11-openstacks	20	20.92	20.00	**21.00**
11-parcprinter	11	11.14	11.88	**11.92**
11-parking	15	15.15	16.61	**18.40**
11-pegsol	20	19.92	20.11	20.04
11-scanalyzer	20	21.00	**24.73**	24.43
11-sokoban	**17**	16.73	16.60	16.96
11-tidybot	13	13.56	**16.21**	15.41
11-transport	13	14.60	18.22	**19.51**
11-visitall	3	3.32	7.30	**7.39**
11-woodworking	4	3.98	4.12	**4.13**
total	440	459.50	496.67	**513.67**

Table 9: Plan Quality of AEES, Diverse-AEES(4) (**DE(4)**), AEES-Aras, and Diverse-AEES-Aras(4) (**DE-Aras(4)**) on all 550 problems from IPC-2008 and IPC-2011.

to minimize the time between solutions, and generate more solutions. This makes it a good test case for DAS.

We repeat the same set of IPC experiments, running DAS with AEES configured as follows: it uses the two planning-specific enhancements of deferred evaluation and preferred operators, and the three heuristics Landmark-cut (admissible cost-to-go heuristic) (Helmert and Domshlak 2009), FF-cost (inadmissible cost-to-go heuristic) and FF-distance (distance-to-go heuristic) (Hoffmann and Nebel 2001). The scores shown use the IPC metric with AEES as a baseline. If L is the plan computed by AEES, then the score of a given plan P is calculated by $cost(L)/cost(P)$. The experimental results are shown in Table 9. Similar to the LAMA-2011 experiments, Diverse-AEES-ARAS(4) gets the highest score, improving the baseline planner AEES by 73.7 units from 440 to 513.7, and achieving the best score in 14 of 23 domains.

Conclusions and Future Work

In this paper, we have shown that the search performance of the current state-of-the-art planner LAMA-2011 suffers from a large amount of *unproductive time*, time which can be used in other ways such as post-processing. The new meta-algorithm of Diverse Any-time Search tries to utilize this unproductive time with randomized restarts so as to generate a larger and more diverse set of plans for a post-processing system such as ARAS to improve upon. Experimental results show that the new framework leads to significant improvements on IPC-2008 and IPC-2011 domains, for both LAMA-2011 and the AEES algorithm.

The best parameter N for DAS depends on factors such as the planning domain, randomizing method and search algorithm. However, in the experiments the performance was robust for small values of N between 3 and 6. One interesting future work is to automatically tune N using information on the search so far.

Acknowledgements

The authors wish to thank Dr. Jordan Tyler Thayer for providing the source code for AEES, and the anonymous referees for their valuable advice. This research was supported by GRAND NCE, a Canadian federally funded Network of Centres of Excellence, and by NSERC, the Natural Sciences and Engineering Research Council of Canada.

References

Bonet, B., and Geffner, H. 2001. Heuristic search planner 2.0. *AI Magazine* 22(3):77–80.

Brafman, R. I.; Geffner, H.; Hoffmann, J.; and Kautz, H. A., eds. 2010. *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*. AAAI.

Chrpa, L.; McCluskey, T.; and Osborne, H. 2012. Optimizing plans through analysis of action dependencies and independencies. In McCluskey et al. (2012), 338–342.

Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds. 2009. *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini et al. (2009), 162–169.

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

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds. 2012. *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Nakhost, H., and Müller, M. 2010. Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In Brafman et al. (2010), 121–128.

Nakhost, H.; Müller, M.; Valenzano, R.; and Xie, F. 2011. Arvand: the art of random walks. In García-Olaya, A.; Jiménez, S.; and Linares López, C., eds., *The 2011 International Planning Competition*, 15–16. Universidad Carlos III de Madrid.

Richter, S., and Helmert, M. 2009. Preferred operators and deferred evaluation in satisficing planning. In Gerevini et al. (2009), 273–280.

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.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In Fox, D., and Gomes, C. P., eds., *AAAI*, 975–982. AAAI Press.

Richter, S.; Thayer, J.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In Brafman et al. (2010), 137–144.

Thayer, J., and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In Walsh, T., ed., *IJCAI*, 674–679. IJCAI/AAAI.

Thayer, J.; Benton, J.; and Helmert, M. 2012. Better parameter-free anytime search by minimizing time between solutions. In Borrajo, D.; Felner, A.; Korf, R. E.; Likhachev, M.; López, C. L.; Ruml, W.; and Sturtevant, N. R., eds., *SOCS*, 120–128. AAAI Press.

Valenzano, R.; Nakhost, H.; Müller, M.; Schaeffer, J.; and Sturtevant, N. 2012. Arvandherd: Parallel planning with a portfolio. In Raedt, L. D.; Bessière, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P. J. F., eds., *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 786–791. IOS Press.

Xie, F.; Nakhost, H.; and Müller, M. 2012. Planning via random walk-driven local search. In McCluskey et al. (2012), 315–322.