Multi-Objective AI Planning: Comparing Aggregation and Pareto Approaches

M. R. Khouadjia[1], M. Schoenauer[1], V. Vidal[2], J. Dréo[3], and P. Savéant[3]

[1] TAO Project, INRIA Saclay & LRI Paris-Sud University, Orsay, France
{mostepha-redouane.khouadjia, marc.schoenauer}@inria.fr,
[2] ONERA-DCSD, Toulouse, France
Vincent.Vidal@onera.fr
[3] THALES Research & Technology, Palaiseau, France
{johann.dreo, pierre.saveant}@thalesgroup.com

Abstract. Most real-world Planning problems are multi-objective, trying to minimize both the makespan of the solution plan, and some cost of the actions involved in the plan. But most, if not all existing approaches are based on single-objective planners, and use an aggregation of the objectives to remain in the single-objective context. *Divide-and-Evolve* is an evolutionary planner that won the temporal deterministic satisficing track at the last International Planning Competitions (IPC). Like all Evolutionary Algorithms (EA), it can easily be turned into a Pareto-based Multi-Objective EA. It is however important to validate the resulting algorithm by comparing it with the aggregation approach: this is the goal of this paper. The comparative experiments on a recently proposed benchmark set that are reported here demonstrate the usefulness of going Pareto-based in AI Planning.

1 Introduction

Most, if not all, classical AI planning solvers are single-objective. Given a planning domain (a set of predicates that describe the state of the system, and a set of actions with their pre-requisites and effects), and an instance of this domain (a set of objects on which the predicates are instantiated into boolean atoms, an initial state and a goal state), classical planners try to find, among the set of all feasible plans (sequences of actions such that, when applied to the initial state, the goal state becomes true), the one with the minimal number of actions (STRIP planning), or with the smallest cost (actions with costs) or with the smallest makespan (temporal planning, where actions have durations and can be applied in parallel). A detailed introduction to (single-objective) AI planning can be found in [1]. It is clear, however, that most planning problems are in fact multi-objective, as the optimal solution in real-world problems often involve some trade-off between makespan and cost [2]. A few trials have been made to turn some classical planners into multi-objective optimizers, either using some

This work was partially funded by DESCARWIN ANR project (ANR-09-COSI-002).

twist in PDDL 2.0[4] to account for both makespan and cost [3–5], or using the new hooks for several objectives offered by PDDL 3.0 [6]. However, all these approaches are based on a linear aggregation of the different objectives, and were not pursued, as witnessed by the new "net-benefit" IPC track, dedicated to aggregated multiple objectives, that took place in 2006 [7] and 2008 [8], ... but was canceled in 2011 due to a lack of entries.

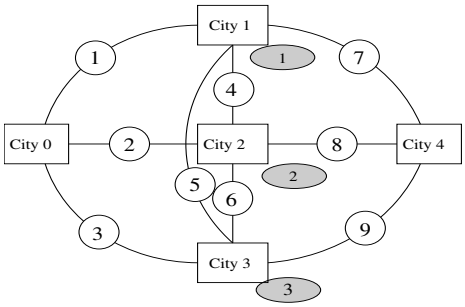
In the framework of Evolutionary Algorithms (EAs), Pareto multi-objective optimization has received a lot of attention [9], and any single-objective EA can "easily" be turned into a multi-objective EA, by modifying the selection step (and possible adding some archiving mechanism). Unfortunately, there exist very few evolutionary AI planners. Directly evolving plans, as in [10], obviously does not scale up, and was never extended to multi-objective setting. Hence, as far as we are aware of, the state-of-the-art in evolutionary AI planning is the previous work of some of the authors, *Divide-and-Evolve* (DAE). DAE evolves variables length sequences of states, that start with the problem initial state and end at the problem goal state. DAE relies on a classical embedded planner to sequentially reach each state of the sequence from the previous one. The concatenation of all plans given by the embedded planner is a solution plan of the original problem. DAE can thus solve all types of planning problems that the embedded planner can solve. Proof-of-concept for DAE was obtained with DAE_{CPT} [11], where the embedded planner was CPT, an exact planner [12] – and already included some small multi-objective experiments. Since then, the DAE paradigm has evolved, and YAHSP a sub-optimal lookahead strategy planning system [13] is now used as the embedded planner [14], and $\text{DAE}_{\text{YAHSP}}$ has reached state-of-the-art results in all planning domains [15], winning the temporal deterministic satisficing track at the last IPC in 2011[5].

The very preliminary work in [11] regarding multi-objective optimization has also been recently revisited with $\text{DAE}_{\text{YAHSP}}$. The lack of existing benchmark suite for multi-objective planning led us to extend the small toy problem from [11] into a tunable benchmark domain, on which different multi-objectivization of $\text{DAE}_{\text{YAHSP}}$ (MO-$\text{DAE}_{\text{YAHSP}}$) were compared [16]. But because the only other approach in AI Planning is the aggregation of the objectives, there is a need to compare the multi-objective approach for $\text{DAE}_{\text{YAHSP}}$ with the single-objective approach based on the linear aggregation of the objectives: this is the purpose of the present work. Section 2 will briefly present planning problems and $\text{DAE}_{\text{YAHSP}}$ in the single-objective setting. In Section 3, the multi-objective context will be introduced. The multi-objective benchmark suite will be presented, and the multi-objectivization of $\text{DAE}_{\text{YAHSP}}$ will be detailed: because YAHSP is a single-objective planner[6], but can be asked to optimize either the makespan or the cost, specific strategies had to be designed regarding how it is called within MO-

[4] Planning Domain Definition Language, a dedicated language for the description of planning problems, set up for the International Planning Competitions (IPC).

[5] See `http://www.plg.inf.uc3m.es/ipc2011-deterministic`

[6] Note that it seems difficult, if at all opssible, to adapt it directly to multi-objective optimization, as it uses very different strategies for the makespan and the cost.



Flight durations are attached to the possible routes (white circles), costs/risks are attached to landing in the central cities (grey circles). Four sets of values given on the right. Default values in the first ("Lin.") column in the Table.

Dur./edge	Lin.	Cvx	Ccve
1	2	2	2
2	4	4	**3**
3	6	6	**4**
4	3	3	**1**
5	5	5	**2**
6	3	3	**1**
7	2	2	2
8	4	4	**3**
9	6	6	**4**
Cost/city			
1	30	30	30
2	20	**11**	**29**
3	10	10	10

Fig. 1: Schematic view, and 3 instances, of simple MultiZeno benchmark.

DaE$_{\text{YAHSP}}$. Section 4 describes the experimental settings, detailing in particular the implementation of the aggregation approach for DaE$_{\text{YAHSP}}$ and the intensive parameter tuning that was performed for all competing algorithms using the off-line problem-independent tuner ParamILS [17]. The results will be presented and discussed in Section 5, and as usual, conclusion and hints about on-going and further work will be given in Section 6.

2 Single-Objective Background

AI Planning Problems: A planning domain D is defined by a set of object types, a set of predicates, and a set of possible actions. An instance is defined by a set of objects of the domain types, an initial state, and a goal state. A predicate that is instantiated with objects is called an atom, and takes a boolean value. For a given instance, a state is defined by assigning values to all possible atoms. An action is defined by a set of *pre-conditions* (atoms) and a set of *effects* (changing some atom values): the action can be executed only if all pre-conditions are true in the current state, and after an action has been executed, the state is modified: the system enters a new state. The goal is to find a plan (sequence of actions) such that it leads from the initial state to the goal state, and minimizes either the number or costs of actions, or the makespan in the case of temporal planning where actions have durations and can be run in parallel.

A simple temporal planning problem in the domain of logistics (inspired by the well-known Zeno problem of IPC series) is given in Figure 1, and will be the basis of the benchmark used in this work: the problem involves cities, passengers, and planes (object types). Passengers can be transported from one city to another (action `fly`), following the links on the figure. One plane can only carry one passenger at a time from one city to another, and the flight duration (number on the link) is the same whether or not the plane carries a passenger

(this defines the *domain* of the problem). In the simplest non-trivial *instance* of such domain, there are 3 passengers and 2 planes. In the initial state, all passengers and planes are in `city 0`, and in the goal state, all passengers must be in `city 4`. In the default case labeled "Lin." in the table right (forget about the costs for now), the not-so-obvious makespan-optimal solution has a total makespan of 8 and is left as a teaser for the reader.

Divide-and-Evolve: Let $\mathcal{P}_D(I, G)$ denote the planning problem defined on domain D with initial state I and goal state G. In order to solve $\mathcal{P}_D(I, G)$, the basic idea of DAE$_X$ is to find a sequence of states $S_1, \ldots, S_n$, and to use some embedded planner X to solve the series of planning problems $\mathcal{P}_D(S_k, S_{k+1})$, for $k \in [0, n]$ (with the convention that $S_0 = I$ and $S_{n+1} = G$). The generation and optimization of the sequence of states $(S_i)_{i\in[1,n]}$ is driven by an evolutionary algorithm. The fitness of a sequence is computed using the embedded planner X, that is called in turn on each of the sub-problems $\mathcal{P}_D(S_k, S_{k+1})$. The concatenation of the corresponding plans (possibly compressed to take into account possible parallelism in the case of temporal planning) is a solution of the initial problem. In case one sub-problem cannot be solved by the embedded solver, the individual is said *unfeasible* and its fitness is highly penalized in order to ensure that unfeasible individuals are always selected after feasible ones. A thorough description of DAE$_X$ can be found in [15]. The rest of this section will briefly recall the evolutionary parts of DAE$_X$.

An individual in DAE$_X$ is a variable-length list of partial states of the given domain (similar to the goal state), and a partial state is a variable-length list of atoms (instantiated predicates). The initialization procedure is based on a heuristic estimation, for each atom, of the earliest time from which it can become true [18]. Furthermore, most existing planners (and this is true for CPT and YAHSP, that have been used within DAE) start by computing some partial mutual exclusion between possible atoms: this information is also used to reduce the search space in DAE$_X$, whenever possible. An individual in DAE$_X$ is hence a variable-length time-consistent sequence of partial states, and each partial state is a variable-length list of atoms that are not pairwise mutually exclusive.

Crossover and mutation operators are applied with respective user-defined probabilities p_{Cross} and p_{Mut}. They are defined on the DAE$_X$ representation in a straightforward manner - though constrained by the heuristic chronology and the partial mutex relation between atoms. **One-point crossover** is adapted to variable-length representation: both crossover points are independently chosen, uniformly in both parents. Only one offspring is kept, the one that respects the approximate chronological constraint on the successive states. **Four different mutation operators** are included, and operate either at the individual level, by adding (`addState`) or removing (`delState`) a state, or at the state level by adding or modifying (`addChangeAtom`) or removing (`delAtom`) some atoms in a uniformly chose state. The choice among these operators is made according to user-defined relative weights (named w-mutationname - see Table 1).

3 Multi-Objective Background

3.1 Pareto-based Multi-Objective Divide-and-Evolve

Two modifications of DAE$_{\text{YAHSP}}$ are needed to turn it into an EMOA: use some multi-objective selection engine in lieu of the single-objective tournament selection that is used in the single-objective context; and compute the value of both objectives (makespan and cost) for both individuals. The former modification is straightforward, and several alternatives have been experimented within [16]. The conclusion is that the indicator-based selection using the hypervolume difference indicator [19] performs best – and only this one will be used in the following, denoted here MO-DAE$_{\text{YAHSP}}$. As explained above, the computation of the fitness is done by YAHSP– and YAHSP, like all known planners to-date, is a single-objective planner. It is nevertheless possible, since PDDL 3.0 [6], to specify other quantities of interest that are to be computed throughout the execution of the final plan, without interfering with the search. Within MO-DAE$_{\text{YAHSP}}$, two strategies are then possible for YAHSP: it can be asked to optimize either the makespan or the cost, and to simply compute the cost or the makespan when executing the solution plan (for feasible individuals).

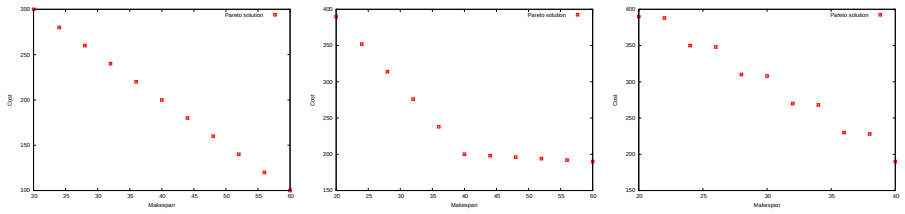
The choice between both strategies is governed by user-defined weights, named respectively W-makespan and W-cost (see table 1). For each individual, the actual strategy is randomly chosen according to those weights, and applied to all subproblems of the individual. Note that those weights are tuned using ParamILS (see Section 4), and it turned out that the optimal values for MO-DAE$_{\text{YAHSP}}$ have always been equal weights: something that was to be expected, as no objective should be preferred to the other.

3.2 Aggregation-based Multi-Objective Divide-and-Evolve

Aggregation is certainly the easiest and most common way to handle multi-objective problems with a single-objective optimization algorithm: a series of single-objective optimization problems are tackled in turn, the fitness of each of these problems is defined by a linear combination of the objectives. In the case of makespan and cost, both to be minimized, each linear combination can be defined by a single parameter α in $[0, 1]$. In the following, F_α will denote $\alpha * \text{makespan} + (1 - \alpha) * \text{cost}$, and DAE$_{\text{YAHSP}}$ run optimizing F_α will be called the α-run. One "run" of the aggregation method thus amounts to running several α-runs, and returns the set of non-dominated individuals among the union of all final populations[7]. Note that different *alpha*-runs might have different optimal values for their parameters: a complete parameter tuning run of PARAMILS must be performed for each α-run to ensure a fair comparison with other well-tuned approaches.

The choice of the number of values to choose for the different α depends on the available resources. But the choice of the actual values aims at exploring

[7] Some adaptive method has been proposed [20], where parameter α is adapted on-line, spanning all values within a single run: this is left for further work.



Linear Convex Concave

Fig. 2: Pareto Fronts for the MULTIZENO6$_{Cost}$ problems described in Figure 1.

the objective space as uniformly as possible, and some issues might arise if both objectives are not scaled similarly. We hence propose here to use some evenly spaced values for α (see Section 4), but only after both objectives have been scaled into [0,1]. However, for such scaling to be possible, some bounds must be known for each objective. When they are not known, these bounds can be approximated from single-objective runs on each of the objectives in turn.

3.3 Multi-Objective Benchmarks:

The reader will have by now solved the little puzzle set in Section 2, and found the solution with makespan 8, that manages to leave no plane idle (detailed solution in [16]). In order to turn this problem into a multi-objective one, costs (or risks) are added to the `fly` actions that land in one of the central cities, leading to two types of problem: In MULTIZENO$_{Cost}$, the second objective is the total costs, that is accumulated every time a plane lands in a central city; In MULTIZENO$_{Risk}$, the second objective is the maximal risk encountered during the complete execution of a plan; both are to be minimized. The complexity of the instances can be increased by adding more passengers: instances with 3, 6 and 9 passengers will be used here. Finally, by tuning the values of the flight durations and the costs/risks, different shapes of the Pareto front can be obtained: Figure 1 summarizes three possible instances for the MULTIZENO domain, and the corresponding Pareto fronts for the 6-passengers case are displayed in Figure 2.

4 Experimental Settings

Parameter Tuning: It is now widely acknowledged that the large number of parameters of most EAs, even though it is a source of flexibility, is also a weakness, in that a poor parameter setting can ruin the performances of the most promising algorithm. Whereas no generic approach exists for on-line control, there are today many available methods for off-line parameter tuning that should be used within any evolutionary experiment, in spite of their huge computational cost.

In this work, unless otherwise stated, the user-defined parameters of both MO-DAE$_{YAHSP}$ and DAE$_{YAHSP}$ shown in Table 1 have been tuned anew for each instance, using the PARAMILS framework [17]. PARAMILS performs an Iterated

Parameters	Range	Description
W-makespan	[0,5]	Weight for makespan strategy for YAHSP
W-cost	[0,5]	Weight for cost/risk strategy for YAHSP
Pop-size	[10,300]	Population size
Proba-cross	[0,1]	Probability to apply cross-over
Proba-mut	[0,1]	Probability to apply one of the mutation
w-addatom	[1,10]	Weight for addChangeAtom mutation
w-addgoal	[1,10]	Weight for addGoal mutation
w-delatom	[1,10]	Weight for delAtom mutation
w-delgoal	[1,10]	Weight for delGoal mutation
Proba-change	[0,1]	Probability to change each atom in the addChangeAtom mutation
Proba-delatom	[0,1]	Probability to delete each atom in the delAtom mutation
Radius	[1,10]	Number of neighbour goals to consider for the addGoal mutation

Table 1: Set of parameters off-line tuned using ParamILS.

Local Search in the space of possible parameter configurations, evaluating each configuration by running the algorithm to be optimized with this configuration on the given instance.

Stopping Criteria: Due to the variable number of calls to YAHSP the number of function evaluation is not representative of the CPU effort of runs of DaE$_{\text{YAHSP}}$. Hence the stopping criterion of all DaE$_{\text{YAHSP}}$ run was set to a given wall-clock time (300, 600 and 1800 seconds for MultiZeno3, 6 and 9 respectively (on an Intel(R) Xeon(R) @ 2.67GHz or equivalent). That of MO-DaE$_{\text{YAHSP}}$was set accordingly: for the sake of a fair comparison, because one run of the aggregated approach requires n runs of the single-objective version of DaE$_{\text{YAHSP}}$, MO-DaE$_{\text{YAHSP}}$was run for n times the time of each of the DaE$_{\text{YAHSP}}$ runs. In the following, n will vary from 3 to 8 (see Section 5). The stopping criterion for ParamILS was likewise set to a fixed wall-clock time: 48h (resp. 72h) for MultiZeno3 and 6 (resp. MultiZeno9), corresponding to 576, 288, and 144 parameter configuration evaluations for MultiZeno3, 6 and 9 respectively.

Performance Metrics and Results Visualization: The quality measure used by ParamILS to optimize the parameters of both MO-DaE$_{\text{YAHSP}}$ and each of the α-runs of DaE$_{\text{YAHSP}}$ is the unary hypervolume I_{H^-} [19] of the set of non-dominated points output by the algorithm with respect to the complete true Pareto front (only instances where the true Pareto front is fully known have been experimented with). The lower the better (a value of 0 indicates that the exact Pareto front has been reached). All reported differences in hypervolume have been tested using Wilcoxon signed rank test at 95% confidence level, unless otherwise stated.

However, and because the true front is made of a few scattered points (at most 17 for MultiZeno9 in this paper), it is also possible to visually monitor the empirical Cumulative Distribution Function of the probability to discover each point, as well as the whole front. This allows some deeper comparison between algorithms even when none has found the whole front. Such *hitting plots* will be used in the following, together with more classical plots of hypervolume vs time. Finally, because hitting plots only tell if a given point was reached and do not

provide any information regarding how far from the other points the different runs ended, more details on the approximated Pareto fronts will be given by visualizing the merged final populations of all runs of given settings.

Implementation: For all experiments, 11 independent runs have been performed, implemented within the ParadisEO-MOEO framework[8]. All performance assessment procedures (hypervolume calculations, statistical tests), have been achieved using the PISA performance assessment tool[9].

5 Experimental Results

This section will compare the Pareto-based MO-DaE$_{\text{YAHSP}}$ and the aggregation approach Agg-DaE$_{\text{YAHSP}}$ on MultiZeno3, 6 and 9. Unless otherwise stated, the default domain definition leading to a linear Pareto front (see Figure 1 and 2-left) will be used, and one Agg-DaE$_{\text{YAHSP}}$ run will be made of 7 different α-runs, with α taking the values $0, 0.1, 0.3, 0.5, 0.7, 0.9$, and 1.

The MultiZeno3 Problem proved to be too easy: both MO-DaE$_{\text{YAHSP}}$ and Agg-DaE$_{\text{YAHSP}}$ find the complete Pareto fronts, and the hitting plots reach 100% in less than 80s (resp. 90s) for the Cost (resp. Risk) version of the instance (not shown here). MO-DaE$_{\text{YAHSP}}$ is slightly slower (resp. faster) than Agg-DaE$_{\text{YAHSP}}$ in the Cost (resp. Risk) instance, but no significant difference is to be reported. Only instances -6 and -9 will be looked at in the following.

The Risk Objective: On these instances, however, the Risk objective proved to be almost too difficult to be of interest here, even though there are only 3 points on the Pareto Front, whatever the number of passengers: as can be seen on Figure 6, no algorithm could identify the complete Pareto front for the MultiZeno9 instance (line 4); for MultiZeno6 (line 2), MO-DaE$_{\text{YAHSP}}$ could reliably identify the whole front (in 9 runs out of 11), while only a single run of Agg-DaE$_{\text{YAHSP}}$ could identify the middle point (40,20). MO-DaE$_{\text{YAHSP}}$ is hence a clear winner here - however, too little information is brought by the risk value, as one single stop in a risky station will completely hide the possibly low-risk remaining of the plan. Further work will aim at designing a smoother fitness for such situations.

The rest of the paper will hence concentrate on the Cost versions of MultiZeno6 and 9 (simply denoted MultiZeno{6,9}), where significant differences between both approaches can be highlighted.

Results on the Default Instance: From the plots of the evolution of the average hypervolumes (Figure 3), MO-DaE$_{\text{YAHSP}}$ is the winner for MultiZeno6, and Agg-DaE$_{\text{YAHSP}}$ is the winner for MultiZeno9. Taking a closer look at the hitting plots (Figure 6), we can see for MultiZeno6 (line 1) that all runs of

[8] `http://paradiseo.gforge.inria.fr/`
[9] `http://www.tik.ee.ethz.ch/pisa/`

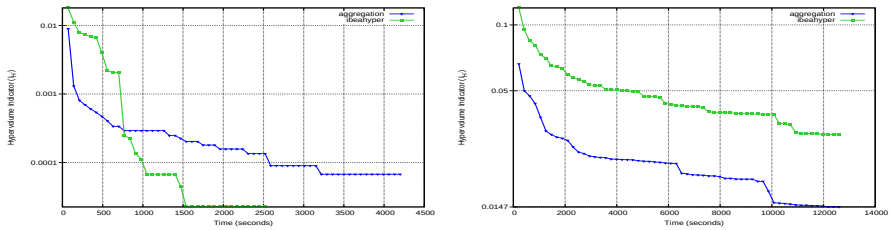
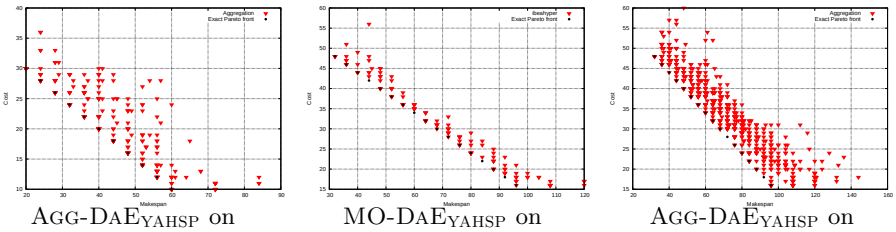


Fig. 3: Evolution of hypervolume for DAE$_{YAHSP}$ (green squares) and AGG-DAE$_{YAHSP}$ (blue triangles) for MULTIZENO6 (left) and MULTIZENO9 (right).



AGG-DAE$_{YAHSP}$ on MULTIZENO6 | MO-DAE$_{YAHSP}$ on MULTIZENO9 | AGG-DAE$_{YAHSP}$ on MULTIZENO9

Fig. 4: Pareto fronts approximations (union of all final populations).

MO-DAE$_{YAHSP}$ reach the complete Pareto front in around 2500s, while only 9 runs out of 11 do reach it. On the other hand, for MULTIZENO9, and though the figures of line 3 are more difficult to read because they contain the CDF for 17 points, slightly more points seem to be reached by AGG-DAE$_{YAHSP}$ than by MO-DAE$_{YAHSP}$. Looking now at the approximations of the Pareto fronts (Figure 4), the fronts returned by AGG-DAE$_{YAHSP}$ for MULTIZENO6 show a large dispersion away from the true front, whereas the same figure for MO-DAE$_{YAHSP}$ (not shown) only contains the true front. Regarding MULTIZENO9, even though it reaches less points from the true front, MO-DAE$_{YAHSP}$ demonstrates a much more robust behavior than AGG-DAE$_{YAHSP}$, for which the approximate fronts are, again, quite dispersed, sometimes far from the true front.

Results on other MultiZeno6 instances: Further experiments have been conducted on different variants of MULTIZENO6 instance, described in Figure 1. The corresponding hitting plots can be seen on Figure 5. As in the LINEAR default case, MO-DAE$_{YAHSP}$ is a clear winner – and this is confirmed by the plots of the approximate Pareto fronts (not shown), for which AGG-DAE$_{YAHSP}$ again shows a much larger dispersion away from the true front than MO-DAE$_{YAHSP}$.

All results presented until now have been obtained by first optimizing the parameters of all algorithms with PARAMILS. Interestingly, when using the parameters optimized by PARAMILS for the Linear instance on these other instances, the results are only slightly worse: this observation will motivate further work dedicated to the generalization of the parameter tuning across instances.

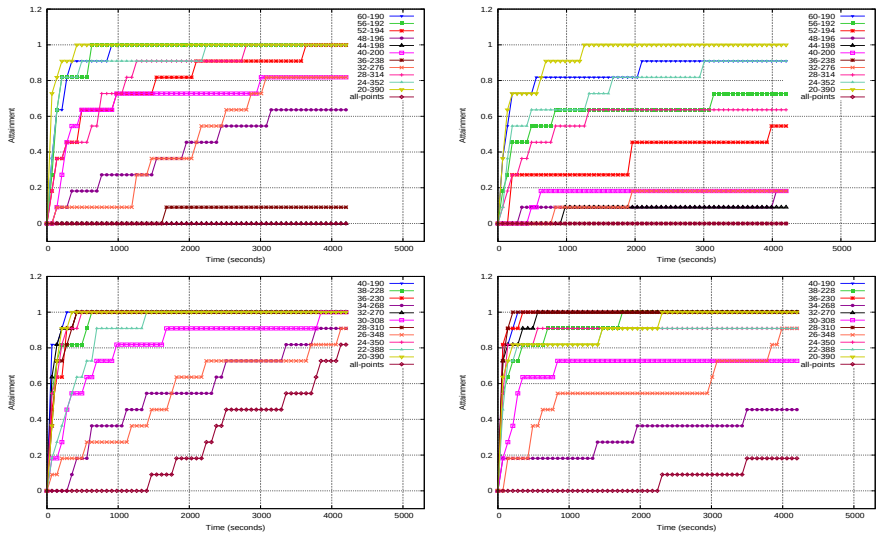


Fig. 5: Hitting plots for MO-DAE$_{\text{YAHSP}}$ (left) and AGG-DAE$_{\text{YAHSP}}$ (right), for instances 2, 3, and 4 of MULTIZENO6 from Figure 1 (from top to bottom).

6 Discussion and Conclusion

The experiments presented in this paper have somehow demonstrated the greater efficiency of the Pareto-based approach to multi-objective AI Planning MO-DAE$_{\text{YAHSP}}$ compared to the more traditional approach by aggregation of the objectives AGG-DAE$_{\text{YAHSP}}$. The case is clear on MULTIZENO6, and on the different instances that have been experimented with, where MO-DAE$_{\text{YAHSP}}$ robustly finds the whole Pareto front (except for the CONVEX instance), whereas AGG-DAE$_{\text{YAHSP}}$ performs much worse in all aspects. This is also true on the MULTIZENO9 instance, in spite of the better hypervolume indicator: indeed, a few more points on the Pareto front are found a little more often, but the global picture remains a poor approximation of the Pareto front. Other experiments on more instances are needed to confirm these first results, and on-going work is concerned with solving instances generated from IPC benchmarks by merging the cost and the temporal domains when the same instances exist in both.

Regarding the computational cost, one AGG-DAE$_{\text{YAHSP}}$ run requires several single-objective runs – and as many parameter tuning procedures. We have chosen here to use 7 different values for α, and it was clear from results not shown here that taking away a few of these resulted in a decrease of quality of the results. The computational cost of the parameter tuning could be reduced, too: first, a complete tuning anew for each instance is unrealistic, and was only done here for the sake of a fair comparison between both approaches; second, even on a single instance, it should be possible to tune all parameters (except those of YAHSP strategy) for all α-runs together. Finally, one of the most promising directions for future research is the on-line tuning of YAHSP strategy, e.g., using a self-adaptive approach, where the strategies are attached to the individual.

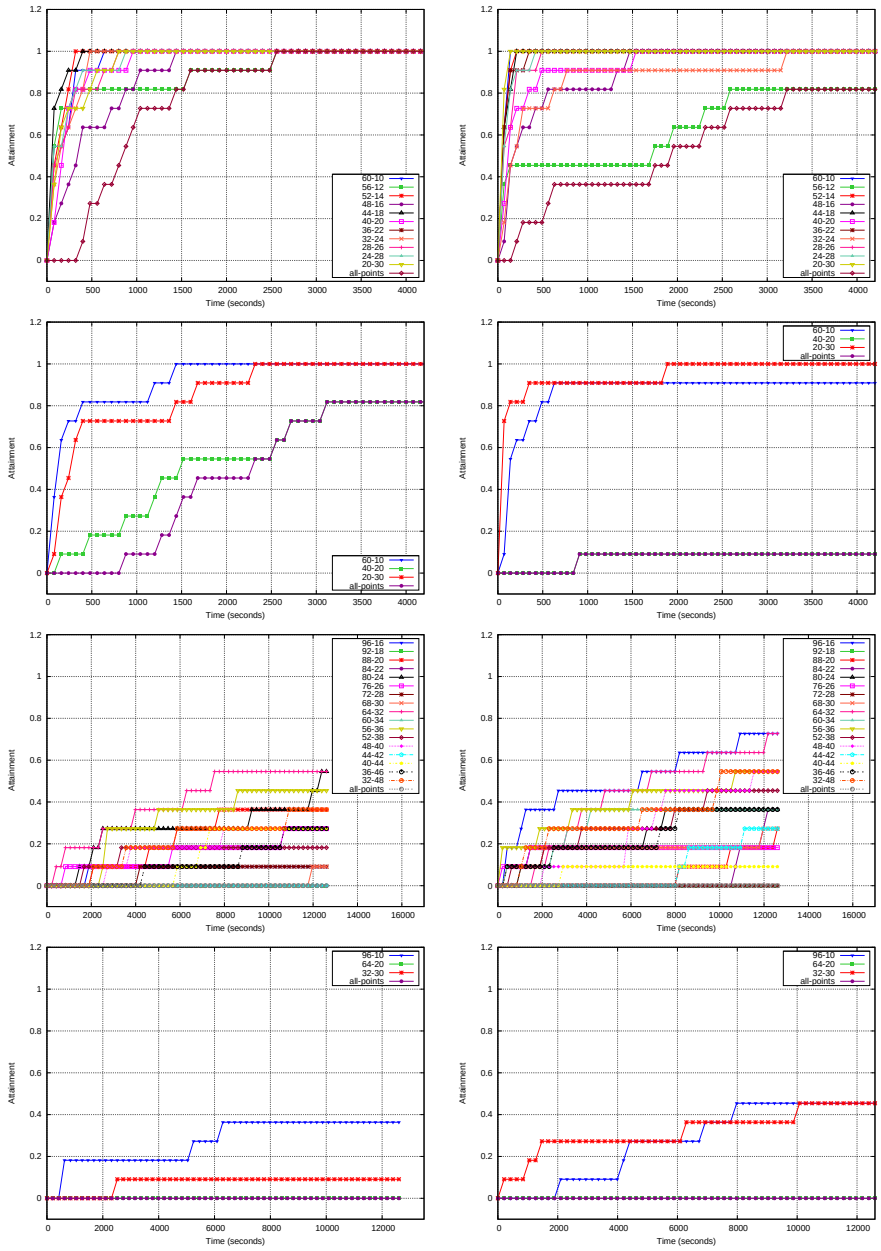


Fig. 6: Hitting plots for MO-DAE$_{\text{YAHSP}}$ (left) and AGG-DAE$_{\text{YAHSP}}$ (right), for instances (from top to bottom) MULTIZENO6$_{Cost}$, MULTIZENO6$_{Risk}$, MULTIZENO9$_{Cost}$, and MULTIZENO9$_{Risk}$. The lower line on each plots is the experimental CDF for the probability to reach the whole Pareto front.

References

1. Ghallab, M., Nau, D., Traverso, P.: Automated Planning, Theory and Practice. Morgan Kaufmann (2004)
2. Kambhampati, S.: 1001 ways to skin a planning graph for heuristic fun and profit. Invited talk at ICAPS'03 (2003)
3. Do, M., Kambhampati, S.: SAPA: A Multi-Objective Metric Temporal Planner. J. Artif. Intell. Res. (JAIR) **20** (2003) 155–194
4. Refanidis, I., Vlahavas, I.: Multiobjective Heuristic State-Space Planning. Artificial Intelligence **145**(1) (2003) 1–32
5. Gerevini, A., Saetti, A., Serina, I.: An Approach to Efficient Planning with Numerical Fluents and Multi-Criteria Plan Quality. Artificial Intelligence **172(8-9)** (2008) 899–944
6. Gerevini, A., Long, D.: Preferences and Soft Constraints in PDDL3. In: ICAPS Workshop on Planning with Preferences and Soft Constraints. (2006) 46–53
7. Chen, Y., Wah, B., Hsu, C.: Temporal Planning using Subgoal Partitioning and Resolution in SGPlan. J. of Artificial Intelligence Research **26**(1) (2006) 323–369
8. Edelkamp, S., Kissmann, P.: Optimal Symbolic Planning with Action Costs and Preferences. In: Proc. 21^{st} IJCAI. (2009) 1690–1695
9. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley (2001)
10. Brie, A.H., Morignot, P.: Genetic Planning Using Variable Length Chromosomes. In Biundo, S., Myers, K.L., Rajan, K., eds.: 15th Intl Conf. on Automated Planning and Scheduling, AAAI Press (2005) 320–329
11. Schoenauer, M., Savéant, P., Vidal, V.: Divide-and-Evolve: a New Memetic Scheme for Domain-Independent Temporal Planning. In Gottlieb, J., Raidl, G., eds.: Proc. 6^{th} EvoCOP, LNCS 3906, Springer (2006) 247–260
12. Vidal, V., Geffner, H.: Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming. In: Proc. AAAI-2004. (2004) 570–577
13. Vidal, V.: A Lookahead Strategy for Heuristic Search Planning. In: Proceedings of the 14^{th} ICAPS, AAAI Press (2004) 150–159
14. Bibaï, J., Savéant, P., Schoenauer, M., Vidal, V.: On the Benefit of Sub-Optimality within the Divide-and-Evolve Scheme. In Cowling, P., Merz, P., eds.: Proc. 10^{th} EvoCOP, LNCS 6022, Springer Verlag (2010) 23–34
15. Bibaï, J., Savéant, P., Schoenauer, M., Vidal, V.: An Evolutionary Metaheuristic Based on State Decomposition for Domain-Independent Satisficing Planning. In R. Brafman et al., ed.: Proc. 20^{th} ICAPS, AAAI Press (2010) 18–25
16. Khouadjia, M.R., Schoenauer, M., Vidal, V., Dréo, J., Savéant, P.: Multi-Objective AI Planning: Evaluating DAE-YAHSP on a Tunable Benchmark. In R. C. Purshouse et al., ed.: Proc. EMO'13. LNCS, Springer Verlag (2013) To appear.
17. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. J. Artif. Intell. Res. (JAIR) **36** (2009) 267–306
18. Haslum, P., Geffner, H.: Admissible Heuristics for Optimal Planning. In: Proc. AIPS-2000. (2000) 70–82
19. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In Xin Yao et al., ed.: Proc. PPSN VIII, LNCS 3242, Springer Verlag (2004) 832–842
20. Jin, Y., Okabe, T., Sendhoff, B.: Adapting weighted aggregation for multiobjective evolution strategies. In E. Zitzler et al., ed.: Proc. EMO 2001, LNCS 1993, Springer Verlag (2001) 96–110