Addressing HTN Planning with Blind Depth First Search[*]

Juan Fernandez-Olivares, Ignacio Vellido, Luis Castillo
Department of Computer Science and Artificial Intelligence
University of Granada
faro@decsai.ugr.es, ignaciove@correo.ugr.es, l.castillo@decsai.ugr.es

Abstract

This paper briefly describes SIADEX, the HTN planner winner of the Partial Order track in the 2020 International Planning Competition. We also show a discussion of the results regarding run time and memory usage for the different problems configured in the competition.

Introduction

In this work we describe SIADEX, an HTN planner (Castillo et al. 2006) based on the same foundations than SHOP2 (Nau et al. 2003). It follows a progression search with a blind depth first search process and allows for partially ordered and recursive tasks networks. The planner was designed as a simple search process yet successfully applied to many real applications (Fdez-Olivares et al. 2006; González-Ferrer et al. 2013; Fdez-Olivares et al. 2019; Fernandez-Olivares and Perez 2020). Its power lies in that it is guided by the knowledge represented in the HTN planning domain, which is written in the hierarchical planning language HPDL (Castillo et al. 2006; González-Ferrer et al. 2013). This language is a hierarchical extension of PDDL 2.2 level 3 (Edelkamp and Hoffmann 2004), so the planner can reason about numeric and temporal information. The 2020 International Planning Competition (IPC2020) is concerned with only STRIPS-like domains, thus these features are obviated in this brief description. We refer to Castillo et al. (2006) and González-Ferrer et al. (2013) for a detailed description of the temporal and numeric capabilities of both the planner and the language.

SIADEX is the winner of the Partial Order track in the IPC held at ICAPS2020. A summary of the results obtained by the planner in this competition is shown in Table 1. SIADEX solved 95 out of 224 (42% overall coverage) problem instances configured in the competition. It shows 100% coverage (solved all the instances) and scores above 0.92 in 3 domains (Barman, Satellite, UMTranslog). For Barman and Satellite it scored 1, what means that it took less than 1 second to solve all the instances. In Rover domain the coverage and score are reasonable (70% and 0.70 respectively). But in domains for which it scores under 0.7, SIADEX shows poor coverage. This may be due to the fact that the task networks in Barman, Rover, Satellite and UM-Translog contain advice to solve the problems, and in other domains the planning knowledge embodies recursive tasks that cannot be adequately handled by the blind DFS search process implemented. For example, in Transport domain or Monroe (either full or partially observable alternatives) recursive tasks are defined as left recursive, while in UM-Translog are right recursive, with additional methods to appropriately guide to a hierarchical, recursive path planning process. In PCP domain SIADEX could not solve any instance. A detailed explanation of the results is shown below in the following sections. It is important to remark that in the verification tests SIADEX did not provide invalid plans for any instance.

Id	Domain	Instances	Solved	% Cov.	Score
4	PCP	17	0	0	0.00
7	Transport	40	1	2.5	0.03
3	Mon-PO	25	2	8	0.05
9	WoodW	30	3	10	0.10
2	Mon-FO	25	8	32	0.24
5	Rover	20	14	70	0.70
1	Barman	20	20	100	0.92
8	UM-Tra	22	22	100	1.00
6	Satellite	25	25	100	1.00

Table 1: Summary of the results of SIADEX in IPC2020. The columns describe the domain id for the Partial Order track, its name, number of instances configured, number of instances solved, coverage expressed as a percentage, and the score calculated as $min\{1, log(t)/log(T)\}$ for t the time to solve all the instances in each domain and $T = 1800$ seconds.

In the following sections we provide a brief, yet detailed description of the main features of SIADEX. Then we discuss the results of SIADEX analyzing its behavior in each domain, ending the paper with some concluding remarks.

[*]This work has been partially supported by Spanish Government Project MINECO RTI2018-098460-B-I00

SIADEX in a nutshell

As explained above, SIADEX is based on a non-informed search process, i.e., the planner does not use numeric heuristic information (the states are not evaluated in any way), thus the only heuristic used to guide the search is that represented in the HTN domain. That is to say, both the search process and the language are strongly headed to provide advice to the planning process. Search nodes are stored in a stack (fringe set) that is efficiently implemented and represents not only the basic information in standard HTN progression search (world state, current task network, solution plan prefix) but an agenda that stores the pending planning decisions. Since the planner follows a lifted planning approach, the main planning decisions (choice points in the search process) are related to tasks unification (either primitive or compound), precondition unification (either actions or methods), and alternative methods to be applied to a compound task (alternative methods for any task are stacked in a linear structure, therefore tasks's methods resemble if-then-else control structures).

HPDL incorporates almost all the features of HDDL (Höller et al. 2020), the language used as standard in this competition, and there is a specific translator from HDDL to HPDL based on the parser provided for the competition. HPDL primitive actions can be defined either as non temporal PDDL actions or PDDL durative actions, inheriting all the features that PDDL provides for preconditions and effects. Compound tasks are defined with a header (name and list of typed parameters) and have associated a set of methods to map compound tasks to predefined task networks. Every method inherits the parameters of the compound task it is associated with, but the language allows for the use of additional variables in a method if necessary. This feature allows for easily translating HDDL methods descriptions, which have their own list of typed parameters, into HPDL methods without affecting their semantics.

Furthermore, HPDL allows to restrict the type of a parameter inside a method, it is even possible to use variables in a method which are not defined in the paramaters of its associated task. That is, a method in HPDL can have a different set of variables than the parameters of the task its is attached to, just as happens in the methods of HDDL or SHOP. Anyway, we think that the syntax of HDDL is better than HPDL for this feature, in the sense that it directly provides a way to define the "local" parameters of a method. On the other hand, maybe HPDL provides more flexibility to manage variables, since one can use as many variables as needed in the "body" of a method without the need to define them in a "header" (HDDL forces to do it).

Method preconditions are described in the same way than PDDL actions preconditions (allowing for universal quantification), but the language extends preconditions expressions with additional features, like a special predicate to *bind* variables to symbolic or numerical expressions, and a *sort-by* structure, borrowed from SHOP2 (Nau et al. 2003), to provide an order between variable unifications according to a given criterion.

Task networks are represented as a partially ordered set of compound/primitive tasks, allowing for recursive definitions. The language also allows for the definition of *inline* tasks, i.e., tasks without name nor parameters which can be defined on-the-fly, which are interpreted as primitive actions and mostly used carry out *ad hoc* inference, by asserting or retracting facts in the world state. In real applications, this is a very welcome feature for knowledge engineers. Regarding the syntax to specify partially-ordered task networks, we borrowed from SHOP the two task ordering operators: the `(:ordered t1 .. tn)` of SHOP in HPDL is `(t1 .. tn)` and `(:unordered t1..tn)` is `[t1 .. tn]`, where `ti` does not refer to the label of a task, but to the header of the task itself. It is possible to combine them, for example, `(t1 [t2 t3] t4)`. We think that using labels for tasks (like in HDDL or in the former proposal of Erol, Hendler, and Nau (1994)) is more expressive since some ordering patterns cannot be expressed with () and []. For example, the following ordering constraint expressed in HDDL `:ordering ( (< t1 t4) (< t2 t4) (< t2 t5) (< t3 t5))` cannot be expressed with this syntax unless using ad-hoc predicates to force the ordering between `[t1, t2]` and `t4`, and `[t2, t3]` and `t5`. Nevertheless, HPDL embodies the capability to represent temporal constraints over start/end points of tasks (either primitive or compound) allowing to describe more expressive ordering patterns (in fact, all the relations of Allen's algebra can be represented (Castillo et al. 2006)). We think that it would be possible to define a compilation process that translates HDDL ordering constraints into HPDL decomposition methods including temporal constraints, but this feature is not used in the current competition.

Results discussion

Regarding runtime and memory usage, Tables 2 and 3 show that in general SIADEX uses around 0.3 seconds and 3.6 KB to solve almost every single instance, without practically no dispersion, except for the hardest problems in domains BarmanBDI and Monroe. Further, we have observed in the datasets of the competition that PyHiPOP (the other nondisqualified contestant) shows more runtime and memory usage than SIADEX (overall, at least an order of magnitude higher). This may partially be due to the fact that data structures in SIADEX are better handled. For example, the representation of nodes in the fringe set in SIADEX is optimized in such a way that, for each node, only the necessary information to revert the changes in case of backtracking is stored. This would allow SIADEX to be more efficient in both planning time and memory usage

In the following we briefly describe the results for each domain, aimed to analyse how complex are the problems proposed to SIADEX, as well as the diversity in the complexity of the instances.

PCP domain. SIADEX was unable to solve any instance in this domain. It is the only domain based on a propositional representation, and we think that this poor behaviour may be due to some failure in de parser from HDDL to HPDL, an issue that needs further study.

Wood Working domain. In this domain 3 out of 30 instances (10% coverage) were solved with a peak memory average around 3.6KB without almost variation among the

Id	Barman [1,20]		Monr-FO [21,45]		Monr-PO [46,70]		PCP [71,87]		Rover [88,107]		Satellite [108,132]		Transport [133,172]		UM-Tr [173,194]		WoodW [195,224]	
1	0.3	20	U	U	U	U	U	U	0.2	16	0.2	5	0.2	8.0	0.3	26	U	U
2	0.3	44	U	U	U	U	U	U	0.3	27	0.2	5	U	U	0.3	13	0.3	3
3	0.3	68	U	U	U	U	U	U	0.3	28	0.2	7	U	U	0.3	14	U	U
4	0.3	42	U	U	U	U	U	U	0.3	19	0.2	11	U	U	0.3	11	0.3	7
5	0.3	65	U	U	U	U	U	U	0.3	39	0.2	10	U	U	0.3	12	U	U
6	0.4	90	28.00	27	U	U	U	U	U	U	0.2	10	U	U	0.3	9	0.3	6
7	0.3	46	U	U	U	U	U	U	0.3	55	0.2	9	U	U	0.3	7	U	U
8	0.3	68	U	U	U	U	U	U	0.3	67	0.2	13	U	U	0.3	9	U	U
9	0.3	94	U	U	U	U	U	U	0.3	65	0.3	17	U	U	0.3	11	U	U
10	0.3	70	17.78	40	U	U	U	U	0.3	75	0.3	12	U	U	0.3	11	U	U
11	0.5	142	U	U	U	U	U	U	0.3	75	0.2	16	U	U	0.5	37	U	U
12	0.4	114	U	U	U	U	U	U	0.3	38	0.2	16	U	U	0.3	17	U	U
13	0.8	214	0.56	28	0.33	35	U	U	0.3	98	0.2	15	U	U	0.3	27	U	U
14	0.4	114	0.82	35	U	U	U	U	U	U	0.2	16	U	U	0.3	9	U	U
15	0.9	238	U	U	129	35	U	U	U	U	0.3	17	U	U	0.3	10	U	U
16	1.8	334	2.80	22	U	U	U	U	0.3	97	0.3	19	U	U	0.3	11	U	U
17	4.2	478	27.78	27	U	U	U	U	U	U	0.3	22	U	U	0.3	9	U	U
18	11.0	718	28.09	25	U	U	-	-	0.5	115	0.3	22	U	U	0.3	9	U	U
19	23.2	958	U	U	U	U	-	-	U	U	0.3	28	U	U	0.3	30	U	U
20	41.7	1198	1.69	24	U	U	-	-	U	U	0.3	27	U	U	0.3	17	U	U
21	-	-	U	U	U	U	-	-	-	-	0.2	18	U	U	0.3	26	U	U
22	-	-	U	U	U	U	-	-	-	-	0.3	25	U	U	0.3	18	U	U
23	-	-	U	U	U	U	-	-	-	-	0.2	5	U	U	-	-	U	U
24	-	-	U	U	U	U	-	-	-	-	0.2	5	U	U	-	-	U	U
25	-	-	U	U	U	U	-	-	-	-	0.2	7	U	U	-	-	U	U

Table 2: Run time average (in seconds) and plan length for the instances solved by Siadex in the domains of the competition. "U" stands for unsolved and "-" for no configured. The instance index interval for each domain is shown under its name. The table only shows 25 rows for each domain, since this is the maximum number of instances that SIADEX reached to solve for all the domains.

instances. This is the only domain where PyHiPOP superseded SIADEX. Without a deeper understanding of the planning domain we are unable to provide an explanation of this fact, nevertheless, we think that PyHiPOP solved more instances (concretely instances 197 and 199) because of the use of unbound variables in the definition of the HTN problem.

Transport domain. 1 out of 40 instances (instance id 133) were solved, with a memory average of 0.00346KB and a runtime of 0.2 seconds, showing a similar memory usage than in other domains. On the other hand, PyHiPOP solved 2 out of 40 for the same domain with an average memory of 1.5 MBytes and an average runtime of 18.6 seconds, both almost without dispersion. The behaviour of SIADEX in this domain may be explained by the way in which the tasks used to recursively solve path planning problems are represented. The order in which methods have to be applied matters, and in the Transport domain the task *get-to* is represented as a left recursive task in which the first method to be used contains the recursive decomposition. If SIADEX firstly addresses the task decomposition with that recursive method, it easily falls into an infinitely recursive loop. This could have been fixed with an improvement of the parsing process, by identifying the appropriate order in which methods should be applied.

Monroe domain. In domain Monroe-Partially-Observable 2 out of 25 instances (instances 59 and 61) were solved, with a memory peak average of 3604.8 KB and 7773394.4 KB, and a runtime of 0.3 and 128.9 seconds, respectively. In Monroe-Fully-Observable 8 out of 25 (32% coverage) instances were solved with an important dispersion among the runtime and memory usage. In fact there is a coefficient of variation[1] of 0.85 in runtime values and 0.87 in memory usage, while in other domains (except BarmanBDI which amounts to 1) this coefficient is less than 0.17. In both domains run time and memory usage is clearly superior to that of other domains. The values found in that domain are an indication that the problems configured in the Monroe domain (in both versions) are diverse in complexity and harder to solve by SIADEX.

Rover domain. SIADEX solved 14 out of 20 problem instances (70% of coverage) with a score of 0.72. It was unable to solve the instances 93, 101, 102, 104, 106 and 107 of this domain. In this domain there is a 17% of variation with respect to the mean value in runtime and a 6% of variation for peak memory. This means that SIADEX solved all the problems in almost the same run time showing a similar use of memory in all of them. We can conclude that the 70% instances in this domain are of similar computational complexity and that the problems are not diverse.

[1]The coefficient of variation measures the significance of the standard deviation with respect to the average.

Id	Bar	M-FO	Rov	Sat	UM	Wwo
1	3.5	U	3.6	3.5	3.5	U
2	3.4	U	3.5	3.5	3.6	3.5
3	3.5	U	3.5	3.5	3.6	U
4	3.5	U	3.6	3.6	3.6	3.5
5	3.4	U	3.5	3.5	3.5	U
6	3.4	2286.6	0	3.5	3.4	3.7
7	3.5	U	3.6	3.5	3.4	U
8	3.4	U	3.5	3.7	3.5	U
9	3.5	U	3.4	3.5	3.5	U
10	3.5	1581.3	3.5	3.6	3.5	U
11	3.4	U	3.4	3.4	13	U
12	3.4	U	3.6	3.5	3.5	U
13	3.5	20.1	3.6	3.7	3.4	U
14	3.6	43.8	0	3.5	3.5	U
15	3.6	U	0	3.4	3.7	U
16	3.6	246.9	3.7	3.5	3.6	U
17	3.9	2281.6	0	3.6	3.5	U
18	11.1	2558.3	3.5	3.5	3.5	U
19	22.6	U	0	3.5	3.5	U
20	31.2	136.1	0	3.5	3.6	U
21	-	U	-	3.5	3.4	U
22	-	U	-	3.5	3.6	U
23	-	U	-	3.5	-	U
24	-	U	-	3.5	-	U
25	-	U	-	3.4	-	U

Table 3: Memory usage of SIADEX (in KBytes). PCP, Transport and Monroe-PO are not shown, but described in the text. “U” stands for unsolved and “-” for no configured.

Barman-BDI domain. In this domain all the instances were solved with a score of 0.92. This domain shows the greater dispersion among the run time and memory usage values, therefore the problems configured are diverse (from the SIADEX perspective) and it seems that the problem configuration could be considered as a good testbed to evaluate the performance of the planner.

Satellite and UM-Translog domains. In both domains SIADEX achieved a 100% coverage and all the problems were solved in less than one second. Similar values in average run time and memory usage are shown, with very little variation, though some variation in the length of plans. These are the easier problems to SIADEX, and we think that for these domains there is more than only domain dynamics encoded in the decomposition methods.

Conclusions

In summary we are presenting an HTN planning algorithm based on DFS that handles almost all the features of HDDL. The planner shows a reasonable behaviour with respect to the domains an problems provided for the competition. All the domains and problems have been successfully translated from HDDL to HPDL, and the planner provides valid solutions in all the problems for which it finds one. However the planner does not provide solutions to several problems in concrete domains. We think that this is due to recursion problems which are difficult to overcome with a blind DFS process, and which in real applications are avoided by knowledge engineering, injecting specific knowledge to the planner, after a thorough analysis of the domain. In the domains without domain-specific advice for how to solve a problem, this planner is bound to lack behind those that apply heuristic guidance. Since SHOP also relies on expert knowledge, having a comparison with it would be very interesting. Anyway, our aim in this competition is to be aware of the situation of our planner with respect current developments, and to deeply analyse the limitations of our planning techniques in order to establish research directions for the improvement of both, the planning process and the planning language.

Acknowledgments

We are very grateful to the organizers of the competition, and specially to Gregor Behnke for his invaluable help and effort to develop the HDDL to HPDL parser. Many of their tests led us to discover both some limitations of our language, and ways to improve them in future versions of our planner.

References

Castillo, L.; Fernández-Olivares, J.; García-Pérez, Ó.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *ICAPS*, 63–72.

Edelkamp, S., and Hoffmann, J. 2004. Pddl 2.2: The language for the classical part of the 4th international planning competition, albert ludwigs universität institüt fur informatik. Technical report, Germany, Technical Report.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, 1123–1128.

Fdez-Olivares, J.; Castillo, L.; Garcıa-Pérez, O.; and Palao, F. 2006. Bringing users and planning technology together. experiences in siadex. In *Proc. ICAPS*, 11–20.

Fdez-Olivares, J.; Onaindia, E.; Castillo, L.; Jordán, J.; and Cózar, J. 2019. Personalized conciliation of clinical guidelines for comorbid patients through multi-agent planning. *Artificial intelligence in medicine* 96:167–186.

Fernandez-Olivares, J., and Perez, R. 2020. Driver activity recognition by means of temporal htn planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, 375–383.

González-Ferrer, A.; Ten Teije, A.; Fdez-Olivares, J.; and Milian, K. 2013. Automated generation of patient-tailored electronic care pathways by translating computer-interpretable guidelines into hierarchical task networks. *Artificial intelligence in medicine* 57(2):91–109.

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. Hddl: An extension to pddl for expressing hierarchical planning problems. In *AAAI*, 9883–9891.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. Shop2: An htn planning system. *Journal of artificial intelligence research* 20:379–404.