Research Statement

Sergio Jiménez Celorrio,

Universitat Pompeu Fabra. Barcelona. Spain.

Abstract

My research addresses complex decision making problems with innovative integrations of Machine Learning and Automated Planning. My current work on this topic has produced novel approaches to the automated generation of programs and controllers for autonomous behavior. My research agenda is exploiting the opportunities and applications of this promising research direction.

1 Scientific contributions

I introduce here my top 5 scientific contributions and the corresponding publications. Next I detail the addressed research problems and the main results I produced.

1. Generalization in automated planning. Effective computation of plans that generalize over different problems [**Jimenez Celorrio, Sergio** and Jonsson, 2015], [Segovia-Aguas et al., 2016], [Lotinac et al., 2016], [Segovia et al., 2016].

2. The *curse of dimensionality* in automated planning. Improving the scalability of planners using relational Machine Learning [De La Rosa et al., 2008], [De la Rosa et al., 2011], [**Jimenez Celorrio, Sergio** et al., 2012].

3. The *knowledge acquisition bottleneck* in automated planning. Learning performance models of actions for planning [**Jimenez Celorrio, Sergio** et al., 2006], [Lanchas et al., 2007], [**Jimenez Celorrio, Sergio** et al., 2008], [**Jimenez Celorrio, Sergio** et al., 2013].

4. The *expressiveness/performance* trade-off in automated planning. Development of a state-of-the-art temporal planner with a classical planning compilation [**Jimenez Celorrio, Sergio** et al., 2015].

5. Evaluation in automated planning. Co-organizer of the 7^{th} International Planning Competition (IPC) and the international workshop on the IPC evaluation. Invited speaker at the international summer school on Automated Planning to discuss about the evaluation of planners [Coles et al., 2012], [López et al., 2013], [López et al., 2015].

Other contributions include: *distinguished paper award* at the International Joint Conference on Artificial Intelligence (IJCAI[1]) 2016. *Co-organizer* of 3 editions of the international workshop on Planning and Learning. *Program committee* at the last 2 editions of the IJCAI and at the last edition of the Symposium on Combinatorial Search (SOCS[2]).

[1] IJCAI is ranked as $A+$ by the CORE rank and with *h5-index* 43 at Google Scholar
[2] SOCS has *h5-index* 18 at Google Scholar

1.1 Generalization in automated planning

While Machine Learning (ML) has difficulties with thorough deliberative tasks, current off-the-shelf planners fail to generalize and need to compute plans starting from scratch, even for similar tasks. The computation of *generalized plans* is a promising research direction to improve and integrate both AI paradigms.

Figure 1 shows a *generalized plan* for navigating to cell $(0, 0)$ in a grid starting from any cell and for any grid size. Variables x and y represent the position of the agent. Instructions `dec(x)` and `dec(y)` decrement x and y. Instructions `goto(0,!(x=0))` and `goto(2,!(y=0))` are conditional gotos that jump to line 0, if $x \neq 0$, and to line 2 if $y \neq 0$. Finally instruction `end` indicates termination.

```
0. dec(x)
1. goto(0,!(x=0))
2. dec(y)
3. goto(2,!(y=0))
4. end
```

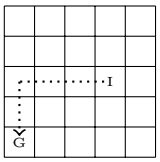


Fig. 1: Generalized plan for navigating to cell $(0, 0)$ in a grid.

Unlike classical sequential plans, *generalized plans* contain branching and repetition constructs which allows them to solve arbitrarily large problems as well as problems with partial observability and non-deterministic actions [Bonet et al., 2010], [Hu and Levesque, 2011], [Srivastava et al., 2011], [Hu and De Giacomo, 2013]. Techniques for generalized planning have then lots of potential applications at diverse areas of Computer Science such as automatic program synthesis, control of autonomous behavior, data wrangling or image classification [Alur et al., 2015, Gulwani et al., 2015, Torlak and Bodik, 2013].

My current research addresses the computation of *generalized plans* represented as *planning programs* [**Jimenez Celorrio, Sergio** and Jonsson, 2015, Segovia-Aguas et al., 2016] and as *finite state controllers* [Segovia et al., 2016]. In both cases generalized plans are computed following an hybrid approach that searches in the space of possible solutions by:

- Exploiting models of the agent and the environment. Generalized plans are computed using off-the-shelf classical planners which allows us to benefit from the model-based machinery of these solvers. Automated planning (AP) is nowadays a well formalized paradigm with powerful algorithms and heuristics that scale-up. Current planners are able to synthesize complex plans with hundreds of actions in seconds time [Geffner and Bonet, 2013].

- Learning from test cases. Test cases constrain the planner to generate solutions that satisfy them besides they are a natural specification for programming and controlling tasks. The *Test Driven Development* (TDD) is a good example where programming tests are created before the code itself is written so programmers can detect bugs at early stages and conceive effective decompositions of the task to solve [Cohen et al., 2003].

1.2 The curse of dimensionality in automated planning

Generating a plan is a PSpace-complete task [Bylander, 1994]. State-of-the-art planners cope with this complexity using reachability analysis and domain-independent heuristics [Helmert, 2006]. Unfortunately these heuristics are not always informed, e.g., they miss key knowledge when there are strong subgoals interactions. Domain-specific Control Knowledge (DCK) has shown to improve the scalability of planners in these situations [Bacchus and Kabanza, 2000, Nau et al., 2003] however, defining DCK is complex and requires expertise in the task to solve and in the used algorithm.

We showed that DCK can be automatically generated from examples using relational learning [De La Rosa et al., 2008, De la Rosa et al., 2011]. In addition we exploited the learned DCK as domain-dependent heuristics that improve the scalability of planners, even in problems with strong subgoals interactions like the BLOCKSWORLD. Table 1 shows the number of problems solved and the quality score achieved by different configurations of our planing system ROLLER, that learns DCK, and the state-of-the-art planner LAMA [Helmert, 2006], that cannot learn and serves as a baseline for comparison.

	ROLLER		ROLLER-BFS		ROLLER-BFS-HA		LAMA	
	solved	score	solved	score	solved	score	solved	score
Blocksworld [30]	**30**	**29.87**	8	2.47	8	2.40	17	0.17
Depots [22]	**21**	**19.86**	20	11.01	20	11.46	20	3.88
Gold-miner [30]	**30**	**26.00**	17	0.03	**30**	5.35	29	12.24
Matching-BW [30]	21	14.84	14	1.32	19	1.71	**25**	**20.02**
Parking [30]	**30**	**28.57**	**30**	22.72	**30**	23.60	23	1.69
Rovers [30]	28	**24.82**	26	13.41	**30**	16.24	**30**	18.59
Satellite [30]	**30**	**22.60**	25	14.61	**30**	18.33	28	15.66
Storage [30]	15	11.02	**19**	12.31	**19**	**16.17**	**19**	9.03
Thoughtful[30]	12	11.99	20	12.38	**23**	**13.09**	20	11.29
TPP [30]	**30**	**29.50**	16	14.83	19	6.27	**30**	9.66
Total	**247**	**219.07**	195	105.09	228	122.32	241	102.23

Tab. 1: Problems solved and quality score of our system ROLLER and the state-of-the-art planner LAMA. Total problems of each domain in brackets.

1.3 Knowledge acquisition bottleneck in automated planning

Off-the-shelf planners reason about correct and complete models of the environment but defining these models is often tricky [Kambhampati, 2007], e.g., the execution of an action may result in countless outcomes, states may not be fully observable, goals may not be completely specified, ...

In [Lanchas et al., 2007, **Jimenez Celorrio, Sergio** et al., 2008, **Jimenez Celorrio, Sergio** et al., 2013] we show how to learn models of the performance of actions in a given environment and how to complete a naive AP model with the learned knowledge. Table 2 shows the number of problems solved by our approach that learns the action performance models and compiles them into (1) a classical action model to feed *FF-Replan* or (2) a probabilistic action model to feed the probabilistic planner GPT [Bonet and Geffner, 2001]. The baseline is *FF-Replan* [Yoon et al., 2007] with the naive AP model and where no learning is performed.

	FF-Replan + learning classical model	*GPT + learning prob. model*	*FF-Replan* No learning
Blocksworld [450]	**450**	390	443
Slippery-Gripper [450]	**450**	**450**	369
Rovers [450]	421	270	**450**
OpenStacks [450]	90	**300**	0
Triangle-tireworld [450]	50	**373**	5
Satellite [450]	**300**	**300**	0

Fig. 2: Problems solved learning action performance models and without learning. Total problems of each domain in brackets.

1.4 The expressiveness/performance trade-off in automated planning

Despite the expressiveness limitations of the classical planning model its use is very extended. A reason is that many expressive tasks can be compiled into classical planning and benefit straightforward from the power of classical planners. Examples are compilations of planning with soft-goals [Keyder and Geffner, 2009], planning with extended temporal goals [Patrizi et al., 2011], or compilations for contingent and conformant planning [Albore et al., 2009a, Palacios and Geffner, 2009] into classical planning.

An interesting example of expressive planning task is temporal planning. Temporal planning can model complex features such as deadlines, conditional effects, conditions during the application of actions, or effects occurring at arbitrary time points [Fox and Long, 2003]. We showed that temporal planning can also be compiled into classical planning so a classical planner can be used to solve the resulting task [**Jimenez Celorrio, Sergio** et al., 2015]. Table 2 reveals this approach is state-of-the-art in temporal planning since our planners TP* significantly outperform ITSAT, TFD and YAHSP3-MT the best temporal planners at the last international planning competition.

	TPSHE	TP(3)	TP(5)	TP(7)	ITSAT	TFD	YAHSP3
Driverlog[20]	19.28/20	1.47/5	0.97/3	0.72/2	1/1	-	1.61/4
Floortile[20]	4.59/5	-	-	-	19.17/20	-	1.69/2
Mapanalyser[20]	12.4/20	7.63/20	7.92/20	8.29/20	-	15.09/17	13.85/20
MatchCellar[20]	15.79/20	15.79/20	15.79/20	15.79/20	19/19	20/20	-
Parking[20]	9.87/20	6.74/20	6.42/19	5.82/17	1.29/6	13.01/20	19.14/20
RTAM[20]	11.64/15	2.92/12	2.6/10	2.02/6	-	-	17.09/20
Satellite[20]	16.22/18	5.72/14	5.49/14	5.49/14	-	14.31/17	13.56/20
Storage[20]	8.95/9	-	-	-	-	-	-
TMS[20]	0.09/18	-	-	-	18/18	-	-
Turn&Open[20]	14/19	5.89/11	5.62/10	5.35/10	6.57/7	12.97/18	-
DLS[20]	2.77/13	5.17/10	5.19/10	5.19/10	20/20	1/1	-
AIA[25]	3/3	7.5/9	8.5/10	8.5/10	-	9.98/10	3/3
Example[20]	-	12.07/17	15.35/16	8.55/9	-	0.62/1	-
Total	**118,6/180**	70,9/138	73,86/132	65,72/118	85,02/91	86,99/104	69,95/89

Tab. 2: Comparison of TP* with the best IPC planners. Quality score / instances solved per domain for each planner. Total problems in brackets.

1.5 Evaluation in automated planning

Research in AP is getting more and more based on empirical evaluation and hence, the demand of methodologies and benchmarks for solid evaluation is increasing. In 1998 the AP community made a move and initiated the International Planning Competition (IPC). The IPC is now considered the most important forum for evaluating AP systems and most of the new AP techniques are evaluated regarding the input language, benchmarks and metrics defined in this competition [Thimm et al., 2016].

In 2011, I co-organized the 7^{th} edition of the IPC, being also responsible for the planning and learning track [Coles et al., 2012], [López et al., 2015]. I also co-organized an international workshop on the evaluation of the IPC in 2012 and in 2013 I was invited speaker at the international summer school on automated planning to discuss about evaluation in planning[3]. All these events took part within the International Conference on Automated Planning and Scheduling (ICAPS), the main international conference on AP (ranked as $A+$ by the CORE conferences rank and with *h5-index* 25 at Google Scholar).

Reproducibility and open knowledge are essential to the advance of research in AP. With this regard I participated in the development of open source software for automatically running experiments in AP and inspecting the results [López et al., 2013].

2 Vision for the future

My research agenda is to develop innovative techniques for the **automatic synthesize of programs and controllers for autonomous behavior** [Torlak and Bodik, 2013, Gulwani et al., 2015] and to explore the opportunities and applications of this promising research direction. More precisely my aim is to advance the state-of-the-art working in the following innovative ideas:

- **Learning high-level state features**. In many tasks a solution that generalizes is only computable if certain high-level state features are available. These features allow to capture key concepts and to accurately distinguish between states for taking good decisions. Examples are the *well-placed*, *above* or *good-tower* concepts for the Blocksworld [Martín and Geffner, 2004]. We have recently studied how to generate such features in programs when features are restricted to *conjunctive queries* [Lotinac et al., 2016]. Open questions that however remain as future work are scaling-up this mechanism, generating such features as arbitrary formulae or computing features for other forms of general solutions such as controllers.

- **Active learning**. Using off-the-shelf planners to synthesize programs or controllers from a large number of test cases is hard because planners degrade their performance as problem sizes increase. Selecting significant test cases is then key to achieve solutions that generalize. Nowadays there is no method to identify small sets of significant test cases for a given task and, in general, the selection of relevant learning examples is an open problem for systems that integrate planning and learning [Fern et al., 2011, De la Rosa et al., 2011, **Jimenez Celorrio, Sergio** et al., 2012].

[3] http://icaps13.icaps-conference.org/student-program/summer-school/

- **Automatic problem decomposition**. The space of possible programs (or controllers) grows exponentially with the number of program lines (or controller states). The development of effective problem decomposition techniques will allow to address more challenging tasks. Work on the automatic generation of planning hierarchies [Hogg et al., 2008, Lotinac and Jonsson, 2016] is a good starting point to address this research question.

- **Validation of generalized plans**. While generalization in ML is assessed by means of statistics and validation sets, it is an open issue for generalized planning. Moreover the validation of programs or controllers allows to guarantee that they are bug-free over a given family of tests cases. Existing works on computing generalized plans only guarantee that a plan generalizes over the test cases comprised in the generalized planning task. Interestingly [Bonet and Geffner, 2015] establish conditions under which FSCs that solve one partially observable non-deterministic problem generalize to other problems.

2.1 Method

Generalized plans can have diverse forms that range from *generalized polices* [Khardon, 1999, Martín and Geffner, 2004], *DS-planners* [Winner and Veloso, 2003], *contingent trees* [Albore et al., 2009b] or *POMDP policies* [Bonet and Geffner, 2009] to *Finite State Controllers* [Bonet et al., 2010]. Likewise algorithms for generalized planning range from pure *top-down* approaches, that search in the space of generalized plans a solution that covers a given set of instances, to *bottom-up* approaches that look at a single instance, compute a solution to it, generalize the solution and merge it with the previous ones, incrementally increasing the domain coverage of the generalized plan.

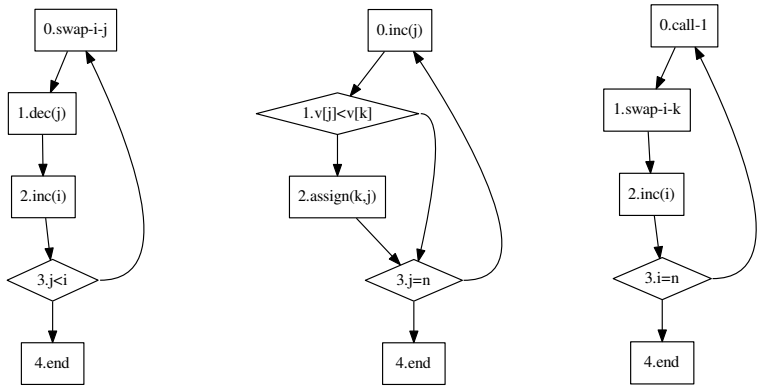
My plan is to address program and controller synthesize starting from our current formalisms and methods [**Jimenez Celorrio, Sergio** and Jonsson, 2015, Segovia-Aguas et al., 2016]. We model programming instructions as planning actions and programming test cases as initial states and goals of a planning task. We showed that is possible to generate plans with control-flow and calls to procedures with a novel compilation that enhances a given classical planning task with mechanisms from programming. Moreover our compilation allows to incorporate prior knowledge in the form of existing procedures, and to automatically complete the remaining program. The compilation uses the standard planning language PDDL [Fox and Long, 2003] so the resulting tasks are solved with off-the-shelf classical planners.

Table 3 shows the CPU time to compute programs that solve well-known programming tasks like computing the n^{th} term of the Fibonacci or the summatory series, reversing a list, sorting a list with the selection sort algorithm or DFS traversing a binary tree of variable size. Figure 3 shows the flow charts of two synthesized programs.

A similar compilation can be defined to the automatic generation of controllers. Finite State Controllers (FSCs) are an effective way to represent autonomous behavior, prominent examples include robotics [Brooks, 1989] and video-games [Buckland, 2004]. My current research explores the notion of hierarchical FSCs and the computation of controllers that can call other controllers

Domain	Time (s)
DFS-Binary-Tree	165
Fibonacci	3570
Reverse-list	22
Sorting-list	30
Summatory	1

Tab. 3: CPU time to generate the programs for these tasks. All programs obtained using the classical planner Fast Downward [Helmert, 2006] on a processor Intel Core i5 3.10GHz x 4 with a 4GB memory bound.



(a) **Reverse** (b) **Sorting** (procedure 1 and main)

Fig. 3: Flow diagrams of synthesized programs: (a) Reversing a list of n elements, (b) Sorting a list of n numbers.

to generate hierarchical FSCs in a modular fashion, or even to apply recursion [Segovia et al., 2016].

Figure 4 shows a recursive FSC that implements the Depth First Search (DFS) algorithm for traversing a binary tree of any size. Here, n is the lone parameter of the controller. Action $\mathsf{visit}(n)$ visits node n, while $\mathsf{copyL}(n, m)$ and $\mathsf{copyR}(n, m)$ assign the left and right child of n to m, respectively. Action $\mathsf{call}(m)$ refers to a recursive call to the FSC itself, assigning argument m to the parameter n and restarting execution from Q_0. Condition $\mathsf{leaf}(n)$ tests whether n is a leaf node, while a hyphen '-' indicates that the transition fires no matter what. The controller state Q_4 is a terminal state, and the action $\mathsf{visit}(child)$ on the transition to Q_4 is in fact not needed and could be removed. However, the entire FSC is automatically generated by our approach, so we present conditions and actions exactly as they appear.

2.2 Impact

The automatic synthesis and validation of programs and controllers is a hot topic on Artificial Intelligence with lots of potential applications. Some evidences

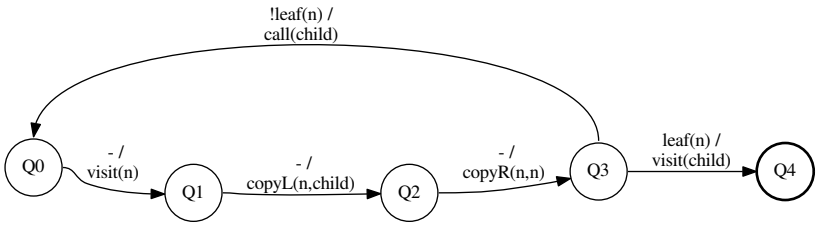


Fig. 4: Synthesized controller that implements the DFS traversal of a binary tree.

are these ideas being part of the FLASH FILL feature in Excel, Microsoft Office 2013 [Gulwani, 2011], the recent *Science* publication on program synthesis [Lake et al., 2015], the funding of a specific research group on this topic at MIT[4], recent groundbreaking results for image classification [Alur et al., 2015] or our work on *Hierarchical Finite State Controllers for Generalized Planning* receiving the IJCAI 2016 distinguished paper award.

References

[Albore et al., 2009a] Albore, A., Palacios, H., and Geffner, H. (2009a). A translation-based approach to contingent planning. In *IJCAI*, pages 1623–1628.

[Albore et al., 2009b] Albore, A., Palacios, H., and Geffner, H. (2009b). A translation-based approach to contingent planning. In *International Joint Conference on Artificial Intelligence*.

[Alur et al., 2015] Alur, R., Bodik, R., Juniwal, G., Martin, M. M., Raghothaman, M., Seshia, S. A., Singh, R., Solar-Lezama, A., Torlak, E., and Udupa, A. (2015). Syntax-guided synthesis. *Dependable Software Systems Engineering*, 40:1–25.

[Bacchus and Kabanza, 2000] Bacchus, F. and Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1):123–191.

[Bonet and Geffner, 2001] Bonet, B. and Geffner, H. (2001). Gpt: a tool for planning with uncertainty and partial information. In *Proc. IJCAI-01 Workshop on Planning with Uncertainty and Partial Information*, pages 82–87.

[Bonet and Geffner, 2009] Bonet, B. and Geffner, H. (2009). Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, pages 1641–1646. Citeseer.

[Bonet and Geffner, 2015] Bonet, B. and Geffner, H. (2015). Policies that generalize: Solving many planning problems with the same policy. *Proc IJCAI 2015*.

[Bonet et al., 2010] Bonet, B., Palacios, H., and Geffner, H. (2010). Automatic derivation of finite-state machines for behavior control. In *AAAI Conference on Artificial Intelligence*.

[Brooks, 1989] Brooks, R. (1989). A robot that walks; emergent behaviours from a carefully evolved network. *Neural Computation*, 1:253–262.

[Buckland, 2004] Buckland, M. (2004). *Programming Game AI by Example*. Wordware Publishing, Inc.

[Bylander, 1994] Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1):165–204.

[Cohen et al., 2003] Cohen, D., Lindvall, M., and Costa, P. (2003). Agile software development. *DACS SOAR Report*.

[4] http://groups.csail.mit.edu/cap/

[Coles et al., 2012] Coles, A., Coles, A., Olaya, A. G., **Jimenez Celorrio, Sergio**, López, C. L., Sanner, S., and Yoon, S. (2012). A survey of the seventh international planning competition. *AI Magazine*, 33(1):83–88.

[De La Rosa et al., 2008] De La Rosa, T., **Jimenez Celorrio, Sergio**, and Borrajo, D. (2008). Learning relational decision trees for guiding heuristic planning. *International Conference on Automated Planning and Scheduling (ICAPS)*.

[De la Rosa et al., 2011] De la Rosa, T., **Jimenez Celorrio, Sergio**, Fuentetaja, R., and Borrajo, D. (2011). Scaling up heuristic planning with relational decision trees. *Journal of Artificial Intelligence Research*, pages 767–813.

[Fern et al., 2011] Fern, A., Khardon, R., and Tadepalli, P. (2011). The first learning track of the international planning competition. *Machine Learning*, 84(1-2):81–107.

[Fox and Long, 2003] Fox, M. and Long, D. (2003). Pddl2. 1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res.(JAIR)*, 20:61–124.

[Geffner and Bonet, 2013] Geffner, H. and Bonet, B. (2013). A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141.

[Gulwani, 2011] Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, volume 46, pages 317–330. ACM.

[Gulwani et al., 2015] Gulwani, S., Hernandez-Orallo, J., Kitzelmann, E., Muggleton, S. H., Schmid, U., and Zorn, B. (2015). Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99.

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

[Hogg et al., 2008] Hogg, C., Munoz-Avila, H., and Kuter, U. (2008). Htn-maker: Learning htns with minimal additional knowledge engineering required. In *AAAI*, pages 950–956.

[Hu and De Giacomo, 2013] Hu, Y. and De Giacomo, G. (2013). A generic technique for synthesizing bounded finite-state controllers. In *International Conference on Automated Planning and Scheduling*.

[Hu and Levesque, 2011] Hu, Y. and Levesque, H. J. (2011). A correctness result for reasoning about one-dimensional planning problems. In *International Joint Conference on Artificial Intelligence*, pages 2638–2643.

[Kambhampati, 2007] Kambhampati, S. (2007). Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In *National Conference on Artificial Intelligence*, page 1601.

[Keyder and Geffner, 2009] Keyder, E. and Geffner, H. (2009). Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, pages 547–556.

[Khardon, 1999] Khardon, R. (1999). Learning action strategies for planning domains. *Artificial Intelligence*, 113(1):125–148.

[Lake et al., 2015] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

[Lanchas et al., 2007] Lanchas, J., **Jimenez Celorrio, Sergio**, Fernández, F., and Borrajo, D. (2007). Learning action durations from executions. *Workshop on Planning and Learning ICAPS 2007*.

[López et al., 2013] López, C. L., **Jimenez Celorrio, Sergio**, and Helmert, M. (2013). Automating the evaluation of planning systems. *AI Communications*, 26(4):331–354.

[López et al., 2015] López, C. L., **Jimenez Celorrio, Sergio**, and Olaya, Á. G. (2015). The deterministic part of the seventh international planning competition. *Artificial Intelligence*, 223:82–119.

[Lotinac and Jonsson, 2016] Lotinac, D. and Jonsson, A. (2016). Constructing hierarchical task models using invariance analysis. In *European Conference on Artificial Intelligence*.

[Lotinac et al., 2016] Lotinac, D., Segovia, J., **Jimenez Celorrio, Sergio**, and Jonsson, A. (2016). Automatic generation of high-level state features for generalized planning. In *International Joint Conference on Artificial Intelligence*.

[Martín and Geffner, 2004] Martín, M. and Geffner, H. (2004). Learning generalized policies from planning examples using concept languages. *Applied Intelligence*, 20:9–19.

[Nau et al., 2003] Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *J. Artif. Intell. Res.(JAIR)*, 20:379–404.

[Palacios and Geffner, 2009] Palacios, H. and Geffner, H. (2009). Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, pages 623–675.

[Patrizi et al., 2011] Patrizi, F., Lipoveztky, N., De Giacomo, G., and Geffner, H. (2011). Computing infinite plans for ltl goals using a classical planner. In *International Joint Conference on Artificial Intelligence*.

[Segovia et al., 2016] Segovia, J., **Jimenez Celorrio, Sergio**, and Jonsson, A. (2016). Hierarchical finite state controllers for generalized planning. In *International Joint Conference on Artificial Intelligence*.

[Segovia-Aguas et al., 2016] Segovia-Aguas, J., **Jimenez Celorrio, Sergio**, and Jonsson, A. (2016). Generalized planning with procedural domain control knowledge. In *International Conference on Automated Planning and Scheduling*.

[Srivastava et al., 2011] Srivastava, S., Immerman, N., Zilberstein, S., and Zhang, T. (2011). Directed search for generalized plans using classical planners. In *International Conference on Automated Planning and Scheduling*, pages 226–233.

[Thimm et al., 2016] Thimm, M., Villata, S., Cerutti, F., Oren, N., Strass, H., and Vallati, M. (2016). Summary report of the first international competition on computational models of argumentation. *AI Magazine*.

[Torlak and Bodik, 2013] Torlak, E. and Bodik, R. (2013). Growing solver-aided languages with rosette. In *Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software*, pages 135–152. ACM.

[Winner and Veloso, 2003] Winner, E. and Veloso, M. (2003). Distill: Learning domain-specific planners by example. In *ICML*, pages 800–807.

[**Jimenez Celorrio, Sergio** et al., 2006] **Jimenez Celorrio, Sergio**, Coles, A., and Smith, A. (2006). Planning in probabilistic domains using a deterministic numeric planner. *25th Workshop of the UK Planning and Scheduling Special Interest Group*.

[**Jimenez Celorrio, Sergio** et al., 2012] **Jimenez Celorrio, Sergio**, De La Rosa, T., Fernández, S., Fernández, F., and Borrajo, D. (2012). A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(04):433–467.

[**Jimenez Celorrio, Sergio** et al., 2008] **Jimenez Celorrio, Sergio**, Fernández, F., and Borrajo, D. (2008). The pela architecture: integrating planning and learning to improve execution. *23rd AAAI Conference on Artificial Intelligence*.

[**Jimenez Celorrio, Sergio** et al., 2013] **Jimenez Celorrio, Sergio**, Fernández, F., and Borrajo, D. (2013). Integrating planning, execution and learning to improve plan execution. *Computational Intelligence*, 29(1):1–36.

[**Jimenez Celorrio, Sergio** and Jonsson, 2015] **Jimenez Celorrio, Sergio** and Jonsson, A. (2015). Computing plans with control flow and procedures using a classical planner. In *Eighth Annual Symposium on Combinatorial Search*.

[**Jimenez Celorrio, Sergio** et al., 2015] **Jimenez Celorrio, Sergio**, Jonsson, A., and Palacios, H. (2015). Temporal planning with required concurrency using classical planning. In *International Conference on Automated Planning and Scheduling*.

[Yoon et al., 2007] Yoon, S. W., Fern, A., and Givan, R. (2007). Ff-replan: A baseline for probabilistic planning. In *ICAPS*, volume 7, pages 352–359.