Random Searching the ILP Lattice

Nicola Di Mauro, Floriana Esposito, Teresa M.A. Basile and Stefano Ferilli

Università degli Studi di Bari, Dipartimento di Informatica, 70125 Bari, Italy

Abstract. In recent we years, significant progress on stochastic local search (SLS) algorithms for solving hard combinatorial (NP-complete) problems, such as the propositional satisfiability problem (SAT) and the travelling salesman problem (TSP), has been made, demonstrating the benefits of randomizing and restarting the search. It has been showed that these methods significantly outperform deterministic search.
A challenging issue in current Inductive Logic Programming (ILP) algorithms is the use of efficient methods for searching very large spaces of hypothesis, in which classical complete deterministic algorithms fail. In this paper, we propose a randomization strategy for ILP, similar to a restart strategy, able to randomly jump in the clause lattice to skip some portion of it.

1 Introduction

Recently, a lot of works demonstrated the benefits of randomizing and restarting the search for solving hard combinatorial problems such as SAT and TSP, and proved that for some challenging AI tasks stochastic search may significantly outperform deterministic search.

It is well known that, in ILP, managing the size of the hypothesis space represents a main problem and in many cases algorithms that perform an exhaustive search are intractable. Most ILP algorithms, such as FOIL and PROGOL, search the lattice of clauses ordered by subsumption in a deterministic manner, based on *refinement* (*top-down* algorithms) or on *least general generalization* (*bottom-up* algorithms) of clauses. This paper proposes a stochastic search method for ILP, based on a random backtracking search in the clause subsumption lattice.

The first stochastic search algorithms applied to ILP clause lattices were Genetic Algorithms (GAs), starting from GA-SMART [2] up to the last works by Tamaddoni-Nezhad and Muggleton [5] and Divina [1]. In [4] the author proposes two simple probabilistic schemes that restrict the search space for an ILP system, namely *stochastic clause selection* that randomly selects a sample of clauses from the search space which contains a good clause, and *iterative beam search*. In [6] authors report a study of randomised restarted search in ILP, proposing a randomised general-to-specific search, a rapid random restart search, and two randomised searches using the stochastic local search algorithms GSAT and WalkGSAT. In [3] a Simulated Annealing framework for ILP was presented, by describing two non deterministic algorithms that independently use generalization and specialization operations, and showing that they partially overcome the difficulties of greedy methods.

2 The Algorithm

We are interested in performing a random search in the clause subsumption lattice bounded at one end by a finite most specific (*bottom*) clause derived using an example seed. The algorithm proposed in this paper (RBK) performs a Random Backtracking Depth First Search in this lattice.

The space explored by backtrack search procedures can be represented by a tree, where an internal node represents a backtrack point, and a leaf node represents either a solution or a failure. In our case, leaf nodes are those clauses that are at least cons% consistent (do not cover at least cons% negative examples) and comp% complete (cover at least comp% positive examples). Randomization is introduced as follows: the algorithm starts the depth search and can backtrack at most $\mathsf{bks} * r$ times, where bks is an input parameter and r is a random number in $]0, 1]$. After that, if it has not found a solution to the problem, then it jumps forward to node j (see below), in order to skip some unexplored space and avoid *traps*, and restarts the depth search from the node j.

The algorithm can perform a maximun number of jumps, jmps, and for each jump it uses an estimate of the search space size to calculate how many nodes have to be skipped. In particular, given the parameter bks, the number, rjmps, of remaining jumps, $\mathsf{bks} * 1,5 * \mathsf{rjmps}$ is an estimate of the maximum number of nodes still to be explored by RBK. Estimating the search space size S, using a Monte Carlo approach, and given the size T of the tree's portion already explored, $(S - T - \mathsf{bks} * 1,5 * \mathsf{rjmps})/\mathsf{rjmps}$ represents the number of nodes to be skipped at each jump. During the search, RBK evaluates each node in order to check for completeness and consistency of the clause it represents. It can prune the subtree of a node n if the completeness (consistency) value is less (greater) than the comp% (cons%) input parameter.

The proposed strategy allows to jump out of a *trap* when it explores a region far from a solution (like a restart strategy), never explores repeated nodes, and may be complete (it can explore the complete search space, bks=+inf).

3 Experiments

Experiments were run using an artificial dataset containing in total 9000 facts. A problem $\mathcal{P}_{xy}$ of the dataset represents a learning problem in which the target clause is made up of 4 variables and y binary predicate symbols, $p_1, \ldots, p_y$, and 50 positive and 50 negative randomly generated examples, containing 4 literals for each predicate symbol, whose arguments are selected, uniformly and without replacement, from the set of all possible pairs of x constants.

Table 1 reports statistics on the algorithm execution. Column 2 reports, for each problem, the estimated search space size. The deterministic variant of RBK, obtained by setting bks=+inf (that corresponds to standard Depth First Search, or DFS), has been executed, on each problem, 10 times[1] with (columns 3 and

[1] Each time the algorithm selects at random a seed from which to start the search.

Table 1. Deterministic Search. Space: estimated search space nodes; ALL: Depth First Search (DFS, deterministic) without pruning; PRUNING: DFS with pruning; Snodes: skipped nodes. Time in seconds.

		ALL		PRUNING			RBK			FOIL
Prob	Space	Time	Nodes	Time	Nodes	Pruned nodes	Nodes	Snodes	Time	
$\mathcal{P}_{44}$	2524,2	2,28	1707,1	1,4	922,3	784,8 (46,0%)	779,9	530,8	1,23	0,52
$\mathcal{P}_{45}$	19608,9	8,53	8014,9	3,48	2411,4	5603,5 (69,9%)	1376,8	649,0	2,04	7,53
$\mathcal{P}_{46}$	181225,6	83,48	77155,5	7,05	4554,5	72601,0 (94,1%)	2026,5	5799,4	3,07	107,54
$\mathcal{P}_{54}$	2689,1	1,04	1001,4	0,4	282,6	718,8 (71,8%)	266,2	134,9	0,38	0,63
$\mathcal{P}_{55}$	20640,8	11,92	10212,7	1,23	884,0	9328,7 (91,3%)	469,1	184,6	0,81	51,40
$\mathcal{P}_{56}$	187724,1	180,84	88698	1,93	1184,1	87513,9 (98,7%)	798,3	1835,1	1,30	**708,61**
P_{64}	2082,5	1,19	1102	0,26	144,9	957,1 (86,8%)	104,6	41,2	0,20	0,62
P_{65}	21295,5	13,08	8620,2	1,11	602,3	8017,9 (93,0%)	527,0	140,2	0,91	**69,07**
P_{66}	209124,7	147,86	50362,5	2,78	1168	49194,5 (97,7%)	627,2	7800,5	1,47	**231,84**

4) and without (columns 5 and 6) pruning the search tree; column 7 reports the number of nodes that have been skipped. It is important to note that a high percentage of the nodes are skipped just because they do not represent good clauses, and hence RBK will have a low probability to skip nodes representing good clauses. As we can see RBK outperforms DFS on all problems by evaluating less nodes. Finally, from the last column it is possible to see that both DFS and RBK are better than FOIL, that in some cases (boldfaced values) is not able to find the target concept.

Concluding, we proposed a random backtracking algorithm for searching the clause subsumption lattice that outperforms both its deterministic variant and a classical ILP algorithm.

References

1. Federico Divina. Relational inductive learning with a hybrid evolutionary algorithm. *AI Communications*, 18(1):67–69, 2005.
2. A. Giordana and C. Sale. Learning structured concepts using genetic algorithms. In D. Sleeman and P. Edwards, editors, *Prooceedings of the 9th Internation Workshop on Machine Learning*, pages 50–55. Morgan Kaufmann, 1992.
3. M. Serrurier, H. Prade, and G. Richard. A simulated annealing framework for ILP. In R. Camacho, R. King, and A. Srinivasan, editors, *ILP 2004*, volume 3194 of *LNAI*, pages 288–304, 2004.
4. A. Srinivasan. A study of two sampling methods for analysing large datasets with ILP. *Data Mining and Knowledge Discovery*, 3(1):95–123, 1999.
5. A. Tamaddoni-Nezhad and S.H. Muggleton. A genetic approach to ILP. In S. Matwin and C. Sammut, editors, *ILP02*, volume 2583 of *LNAI*, pages 285–300. Springer, 2002.
6. F. Železný, A. Srinivasan, and D. Page. A monte carlo study of randomised restarted search in ILP. In R. Camacho, R. King, and A. Srinivasan, editors, *ILP04*, volume 3194 of *LNAI*, pages 341–358. Springer, 2004.