Generation of Near-Optimal Solutions Using ILP-Guided Sampling

Ashwin Srinivasan[1], Gautam Shroff[2], Lovekesh Vig[2], and Sarmimala Saikia[2]

[1] Department of Computer Science & Information Systems
BITS Pilani, Goa Campus, Goa 403726.
[2] TCS Research, New Delhi.

Abstract. Our interest in this paper is in optimisation problems that are intractable to solve by direct numerical optimisation, but nevertheless have significant amounts of relevant domain-specific knowledge. The category of heuristic search techniques known as estimation of distribution algorithms (EDAs) seek to incrementally sample from probability distributions in which optimal (or near-optimal) solutions have increasingly higher probabilities. Can we use domain knowledge to assist the estimation of these distributions? To answer this in the affirmative, we need: (a) a general-purpose technique for the incorporation of domain knowledge when constructing models for optimal values; and (b) a way of using these models to generate new data samples. Here we investigate a combination of the use of Inductive Logic Programming (ILP) for (a), and standard logic-programming machinery to generate new samples for (b). We demonstrate the approach on two optimisation problems (predicting optimal depth-of-win for the KRK endgame, and job-shop scheduling). Our results are promising and suggest that the use of ILP-constructed theories could be a useful technique for incorporating complex domain-knowledge into estimation distribution procedures.

1 Introduction

There are many real-world planning problems for which domain knowledge is qualitative, and not easily encoded in a form suitable for numerical optimisation. Here, for instance, are some guiding principles that are followed by the Australian Rail Track Corporation when scheduling trains: (1) If a "healthy" Train is running late, it should be given equal preference to other healthy Trains; (2) A higher priority train should be given preference to a lower priority train, provided the delay to the lower priority train is kept to a minimum; and so on. It is evident from this that train-scheduling may benefit from knowing if a train is "healthy", what a train's priority is, and so on. But are priorities and train-health fixed, irrespective of the context? What values constitute acceptable delays to a low-priority train? Generating good train-schedules will require a combination of quantitative knowledge of a train's running times and qualitative knowledge about the train in isolation, and in relation to other trains. In this paper, we propose a heuristic search method, that comes under the broad category of an

estimation distribution algorithm (EDA). EDAs iteratively generates better solutions to the optimisation problem using machine-constructed models. Usually EDA's have used generative probabilistic models, such as Bayesian Networks, where domain-knowledge needs to be translated into prior distributions and/or network topology. In this paper, we are concerned with problems for such a translation is not evident. Our interest in ILP is that it presents perhaps one the most flexible ways to use domain-knowledge when constructing models.

2 EDA for Optimisation

The basic EDA approach we use is the one proposed by the MIMIC algorithm [4]. Assuming that we are looking to minimise an objective function $F(\mathbf{x})$, where $\mathbf{x}$ is an instance from some instance-space $\mathcal{X}$, the approach first constructs an appropriate machine-learning model to discriminate between samples of lower and higher value, i.e., $F(\mathbf{x}) \leq \theta$ and $F(\mathbf{x}) > \theta$, and then generates samples using this model

Procedure EOMS: Evolutionary Optimisation using Model-Assisted Sampling

1. Initialize population $P := \{\mathbf{x}_i\}$; $\theta := \theta_0$
2. while not converged do
   (a) for all $\mathbf{x}_i$ in P $label(\mathbf{x}_i) := 1$ if $F(\mathbf{x}_i) \leq \theta$ else $label(\mathbf{x}_i) := 0$
   (b) train model M to discriminate between 1 and 0 labels i.e., $P(\mathbf{x} : label(\mathbf{x}) = 1|M) > P(\mathbf{x} : label(\mathbf{x}) = 0|M)$
   (c) regenerate P by repeated sampling using model M
   (d) reduce threshold θ
3. return P

Fig. 1. Evolutionary optimisation using machine-learning models to guide sampling.

2.1 ILP-assisted Evolutionary Optimisation

We propose the use ILP as the model construction technique in the EOMS procedure, since it provides an extremely flexible way to construct models using domain-knowledge. On the face of it, this would seem to pose a difficulty for the sampling step: how are we to generate new instances that are entailed by an ILP-constructed model? There are two straightforward options. First, if we have an enumerator of the instance space $\mathcal{X}$, then we could resort to a form of rejection-sampling. Second, we can restrict ILP-theories for any predicate to generative clauses, which allows the theories to be used generatively.[3], using standard logic-programming inference machinery to generate instances of the

[3] A syntactic way to do this is by adding constraints to the body of the clause that impose range (that is, type) restrictions on the variables in the head: see [5].

success-set of each predicate. Instances obtained in this manner are selected with some probability to achieve a non-uniform sampling of the success-set. Both these methods are viable for the purposes of this paper, but in general, we expect that more sophisticated ways of sampling would be needed: see for example [3]. The procedure EOIS in Fig. 2 is a refinement of the EOMS procedure above.

Procedure EOIS: Evolutionary Optimisation using ILP-Assisted Sampling

Given: (a) Background knowledge B; (b) an upper-bound θ^* on the cost of acceptable solutions; (c) a decreasing sequence of cost-values $\theta_1, \theta_2, \ldots, \theta_n$ s.t. $\theta_1 \geq \theta^* \geq \theta_n$; and (d) an upper-bound on the sample size n

1. Let $M_0 := \emptyset$ and $P_0 := sample(n, M_0, B)$
2. Let $k = 1$
3. while $(\theta_k \geq \theta^*)$ do
   (a) ${E_k}^+ := \{\mathbf{x}_i : \mathbf{x}_i \in P \text{ and } F(\mathbf{x}_i) \leq \theta_k\}$ and ${E_k}^- := \{\mathbf{x}_i : \mathbf{x}_i \in P \text{ and } F(\mathbf{x}_i) > \theta_k\}$
   (b) $M_k := ilp(B, {E_k}^+, {E_k}^-)$
   (c) $P_k := sample(n, M_k, B)$
   (d) increment k
4. return P_{k-1}

Fig. 2. Evolutionary optimisation using ILP models to guide sampling.

In EOIS, $ilp(B, E^+, E^-)$ is an ILP algorithm that returns a theory M s.t. $B \wedge M \models E^+$; $B \wedge M$ is inconsistent with the E^- only to the extent allowed by constraints in B; and $sample(n, M, B)$ returns a set of at most n instances entailed by $B \wedge M$, if $M \neq \emptyset$. If $M = \emptyset$, it returns a random selection of n instances from the instance-space. In general, we expect *sample* to be implemented as a probabilistic logic program (see [3]). Here, we make do by providing an initial sample to EOIS as input when $M = \emptyset$ (to prevent biasing future iterations: this sample is obtained by uniform random selection from the instance space). For subsequent steps, we assume the availability of a generator that returns a sample of the success-set of target-predicate. That is, when $M \neq \emptyset$, *sample* returns the set $S = \{e_i : 1 \leq i \leq n \text{ e}_i \in \mathcal{X} \text{ and } B \wedge M \vdash e \text{ and } Pr(e_i) \geq \delta\}$, where δ is some probability threshold (correctly therefore $sample(n, M, B)$ should be $sample(n, \delta, M, B)$). Thus if $\delta = 1$, the first n instances derived (or fewer if there are less) using SLD-resolution with $B \wedge M$ will be selected. Finally, we note that on iterations $k \geq 1$, we can use the data $P_0 \cup \cdots \cup P_{k-1}$ to obtain training examples ${E_k}^+$ and ${E_k}^-$ since the actual costs for the P's have have already been computed. For clarity, this detail has been omitted in EOIS.

3 Empirical Evaluation

For reasons of space, only bare details are presented here. For a complete description, see [7].

3.1 Aims

Our aims in the empirical evaluation are to investigate the following conjectures:

(1) On each iteration, the EOIS procedure will yield better samples than simple random sampling of the instance-space; and
(2) On termination, the EOIS procedure will yield more near-optimal instances than simple random sampling of the same number of instances as used for constructing the model.

3.2 Materials

Data We use two synthetic datasets, one arising from the KRK chess endgame, and the other a restricted, but nevertheless hard 5×5 job-shop scheduling (scheduling 5 jobs taking varying lengths of time onto 5 machines, each capable of processing just one task at a time). The optimisation problem we examine for the KRK endgame is to predict the depth-of-win with optimal play [1].

The job-shop scheduling problem is less controlled than the chess endgame, but is nevertheless representative of many real-life applications (like scheduling trains), which are, in general, known to be computationally hard. We use a job-shop problem with five jobs, each consisting of five tasks that need to be executed in order. These 25 tasks are to be performed using 5 machines, each capable of performing a particular task, albeit for any of the jobs. A 5×5 matrix defines how long task j of job i takes to execute on machine j.

3.3 Background Knowledge

For Chess, background predicates encode the following (WK denotes the While King, WR the White Rook, and BK the Black King): (a) Distance between pieces WK-BK, WK-BK, WK-WR; (b) File and distance patterns: WR-BK, WK-WR, WK-BK; (c) "Alignment distance": WR-BK; (d) Adjacency patterns: WK-WR, WK-BK, WR-BK; (e) "Between" patterns: WR between WK and BK, WK between WR and BK, BK between WK and WR; (f) Distance to closest edge: BK; (g) Distance to closest corner: BK; (h) Distance to centre: WK; and (i) Inter-piece patterns: Kings in opposition, Kings almost-in-opposition, L-shaped pattern. We direct the reader to [2] for the history of using these concepts, and their definitions.

For Job-Shop, background predicates encode: (a) schedule job J "early" on machine M (early means first or second); (b) schedule job J "late" on machine M (late means last or second-last); (c) job J has the fastest task for machine M; (d) job J has the slowest task for machine M; (e) job J has a fast task for machine M (fast means the fastest or second-fastest); (f) Job J has a slow task for machine M (slow means slowest or second-slowest); (g) Waiting time for machine M; (h) Total waiting time; (i) Time taken before executing a task on a machine. Correctly, the predicates for (g)–(i) encode upper and lower bounds on times, using the standard inequality predicates $\leq$ and $\geq$.

3.4 Results

Results relevant to conjectures (1) and (2) are tabulated in Fig. 3 and Fig. 4. The principal conclusions that can drawn from the results are these: (1) For both problems, and every threshold value θ_k, the probabilty of obtaining instances with cost at most θ_k with model-guided sampling is substantially higher than without a model. This provides evidence that model-guided sampling results in better samples than simple random sampling (Conjecture 1); and (2) For both problems and every threshold value θ_k, samples obtained with model-guided sampling contain a substantially number of near-optimal instances than samples obtained with a model (Conjecture 2)
We note also that all results have been obtained by sampling a small portion of the instance space (about 10 % for Chess, and about 3 % for Job-Shop). We now

Model	$Pr(F(\mathbf{x}) \leq \theta_k \mid M_k)$		
	$k=1$	$k=2$	$k=3$
None	0.136	0.022	0.001
ILP	0.816	0.462	0.409
	(6.0)	(21.0)	(409.0)

(a) Chess

Model	$Pr(F(\mathbf{x}) \leq \theta_k \mid M_k)$		
	$k=1$	$k=2$	$k=3$
None	0.507	0.025	0.003
ILP	0.647	0.171	0.080
	(1.3)	(6.8)	(26.7)

(b) Job-Shop

Fig. 3. Probabilities of obtaining good instances $\mathbf{x}$ for each iteration k of the EOIS procedure. In effect, this is an estimate of the precision when predicting $F(\mathbf{x}) \leq \theta_k$. "None" corresponds to simple random sampling ($M_k = \emptyset$). The number in parentheses below each ILP entry denotes the ratio of that entry against the corresponding entry for "None". This represents the gain in precision of using the ILP model over simple random sampling.

Model	Near-Optimal Instances		
	$k=1$	$k=2$	$k=3$
None	1/27	2/27	3/27
ILP	11/27	22/27	27/27
	(1000)	(1964)	(2549)

(a) Chess

Model	Near-Optimal Instances		
	$k=1$	$k=2$	$k=3$
None	3/304	6/304	9/304
ILP	6/304	28/304	36/304
	(1000)	(1987)	(2895)

(b) Job-Shop

Fig. 4. Fraction of near-optimal instances ($F(\mathbf{x}) \leq \theta^*$) generated on each iteration of EOIS. The fraction a/b denotes that a instances of b are generated. The numbers in parentheses denote the number of training instances used by the ILP engine. The values with "None" are the numbers expected by sampling the same number of training instances as used for training the ILP engine.

examine the result in more detail. It is evident that the performance on the Job-Shop domain is not as good as on Chess. The natural question that arises is: Why is this so? We conjecture that this is a consequence of the background knowledge for Job-Shop not being as relevant to low cost values, as was the case for Chess. Some evidence for this was already apparent when we had to increase the lengths of clauses allowed for the ILP engine (this is usually a sign that the background knowledge is somewhat low-level). In contrast, with Chess, some of the concepts refer specifically to "cornering" the Black King, with a view of ending the game as soon as possible. We would expect these predicates to be especially useful for positions at depths-of-win near 0. Evidence of the unreliable performance of the EOIS procedure in Chess, with irrelevant background knowledge is in Fig. 5. These results suggest a refinement to the conclusions we can draw from the use of EOIS, namely: we expect the EOIS procedure to be less effective if the background knowledge is not very relevant to low-cost solutions.

Background	$Pr(F(\mathbf{x}) \leq \theta_k \mid M_{k,B})$		
B	$k=1$	$k=2$	$k=3$
B_{low}	0.658	0.417	0.0
B_{high}	0.816	0.462	0.409

(a)

Background	Near-Optimal Instances		
B	$k=1$	$k=2$	$k=3$
B_{low}	17/27 (1000)	4/27 (1959)	0/27 (2581)
B_{high}	11/27 (1000)	17/27 (1964)	27/27 (2549)

(b)

Fig. 5. Precision (a), and recall of near-optimal instances (b) of the EOIS procedure with background knowledge of low relevance to near-optimal solutions. The results are for Chess, with B_{low} denoting background predicates that simply define the geometry of the board, using the predicates *less_than* and *adjacent*. B_{high} denotes background used previously, with high relevance to low depths-of-win. The numbers in parentheses in (b) are the number of training instances as before.

4 Concluding Remarks

It is uncommon to see ILP applied to optimisation problems. The use of ILP-constructed theories within an evolutionary optimisation procedure is one answer to the question of how to use ILP for optimisation (but not the only answer: recent work in [6], for a different approach). This requires the ILP model to be used generatively, which is not common practice in ILP. In this paper we have been able to do this by a combination of careful definition of the background knowledge, and adding range-restrictions to clauses constructed by ILP. Our results provide evidence that this combination of ILP and logic-programming provides one way of incorporating complex, but relevant domain-knowledge into evolutionary optimisation.

1. M. Bain and S. Muggleton. Learning optimal chess strategies. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 13*, pages 291–309. Oxford University Press, Inc., New York, NY, USA, 1995.
2. Gabriel Breda. KRK Chess Endgame Database Knowledge Extraction and Compression, 2006. Diploma Thesis, Technische Universität, Darmstadt.
3. James Cussens. Stochastic logic programs: Sampling, inference and applications. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 115–122, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
4. Jeremy S De Bonet, Charles L Isbell, Paul Viola, et al. Mimic: Finding optima by estimating probability densities. *Advances in neural information processing systems*, pages 424–430, 1997.
5. S. Muggleton and C. Feng. Efficient Induction of Logic Programs. In S. Muggleton, editor, *Inductive Logic Programming*, pages 281–298. Academic Press, London, 1992.
6. Y. Shiina and H. Ohwada. Using Machine-Generated Soft Constraints for Roster Problems. In S. Muggleton and H. Watanabe, editors, *Latest Advances in Inductive Logic Programming*, pages 227–234. World Scientific Press, 2014.
7. A. Srinivasan, G. Shroff, L. Vig, and S. Saikia. Generation of Near-Optimal Solutions Using ILP-Assisted Sampling, 2016. Technical Report TR-EOIS-2016-1, TCS Research. Available at https://arxiv.org/pdf/1608.01093.pdf.