On the use of walkSAT based algorithms for MLN inference in some realistic applications

Romain Rincé, Romain Kervarc, Philippe Leray

On the Use of WalkSAT Based Algorithms for MLN Inference in Some Realistic Applications

Romain Rincé[1,2], Romain Kervarc[1], and Philippe Leray[2]

[1] ONERA – The French Aerospace Lab, Palaiseau
[2] LS2N – Laboratoire des Sciences du Numérique de Nantes, UMR CNRS 6004
Université de Nantes, France
romain.rince@onera.fr,romain.kervarc@onera.fr
philippe.leray@univ-nantes.fr

Abstract. WalkSAT is a local search algorithm conceived for solving SAT problems, which is also used for sampling possible worlds from a logical formula. This algorithm is used by Markov Logic Networks to perform slice sampling and give probabilities from a knowledge base defined with soft and hard constraints. In this paper, we will show that local search strategies, such as WalkSAT, may perform as poorly as a pure random walk on a category of problems that are quite common in industrial fields. We will also give some insights into the reasons that make random search algorithms intractable for these problems.

Keywords: Complex Event Processing, Local search, SAT solver, Markov logic network, WalkSAT, Chronicles, First-Order Logic, Temporal Logic

1 Introduction

In many industrial fields, it is necessary to manage large temporal streams of raw data that usually represent the first observable layer of a very complex system with a broad variety of interactions between agents. Being able to extract more valuable informations from this data is a major concern, especially in case of critical activities like air traffic safety, market surveillance, or cyber-attack detection. These needs have led to much research about efficient methods for analysing this kind of temporal data and recognising high level information.

In this paper, we focus on a specific formalism from the complex event processing domain known as *Chronicles* [8], more precisely on this latest version [9]. Chronicles are a powerful way to represent and recognise activities on temporal data flows. However, since chronicles are defined with logical formulae, using them on unreliable data may prove hard. Indeed, these streams are usually made of events detected by sensors, which can fatally miss some event or even produce them erroneously making the chronicles inefficient. Our first intent was to make chronicles able to deal with uncertainty by combining them with Markov Logic.

Markov Logic Networks (MLN), introduced by [10], are graphical models that try to combine the expressiveness of first-order logical formulae and the ability of graphical models to deal with uncertainty. During the last ten years, MLN have

been quite popular and have led to many practical and efficient experiments [2, 11,18], ranging over a wide spectrum of fields, including automatic image analysis [2, 7] or event recognition [16, 17]. The efficiency of MLN inference mainly relies on algorithm MC-SAT [10] which approximates probabilities of different worlds using a sampling technique. This sampling is performed by MaxWalkSAT for sampling plausible worlds. In this paper, we will mostly focus on this algorithm and issues that appear when it is applied to a specific category of problems.

MaxWalkSAT is a SAT solver, extended from WalkSAT, which handles weighted soft and hard constraints. SAT solvers are mainly divided into two families: complete and local search. Complete methods are extensions of the Davis-Putnam-Logemann-Loveland algorithm (DPLL) from [3] that assigns truth values to predicates of a first-order logic (FOL) formula until reaching a satisfactory assignment, known as world. This problem is NP-hard, but different techniques of pruning speed up the resolution.

Local search methods, as WalkSAT, on the other hand, try to find a satisfactory world by solving each inconsistent clause one by one. An advantage of local search strategies is that they are an approximately uniform sampler.

This paper is organised as follows. In Section 2, we will present briefly chronicles and their design with a FOL formula in Conjunctive Normal Form (CNF). In Section 3, we will introduce the Markov Logic and the WalkSAT algorithm. In Section 4, we will narrow chronicles design to a simple but common problem on FOL, for which we will show that trying and finding a satisfiable solution for this structure with WalkSAT strategies is exponential on time, resulting in MLN providing untrustworthy probabilities. In Section 5, we will give some insights into the reasons that lead WalkSAT strategies to be intractable in this case.

2 Chronicles

The purpose of chronicles is the detection of meaningful information within temporal data flows. Data is composed of events, called Low Level Events (LLE), together with their detection time. Using these LLE and Allen's operators[3] [1], it is possible to define formulae describing the recognition of specific High Level Events (HLE). These HLE can be reused to compose more complex HLE. Figure 1 presents the recognitions of a chronicle where an event A precedes two events B, without any event C between the two B, which is denoted $A((BB) - [C])$. Here, $(BB) - [C]$ is a sub-chronicle used to define the final chronicle.

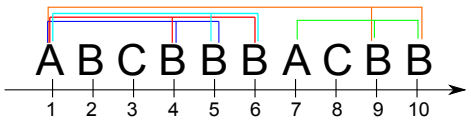

Fig. 1. All recognitions of chronicle $A((BB) - [C])$

[3] In fact, chronicles use more than 15 interval operators, including Allen's and some duration-related constraints. For further details, the reader may refer to [9].

As said before, this logical construction needs the analysed data to be lossless or errorless. To make chronicles robust to uncertainty, we modelled them into Markov Logic. In order to achieve this, we transformed each needed operator into a FOL formula. This step is almost straightforward.

For instance, the *sequence* operator OpSeq may be defined as follows:

$$\mathsf{Ch}(c_1,t_1,t_2) \wedge \mathsf{Ch}(c_2,t_3,t_4) \wedge (t_2 \leq t_3) \implies \mathsf{OpSeq}(c_1,c_2,t_1,t_2,t_3,t_4) \quad (1)$$

$$\neg\mathsf{Ch}(c_1,t_1,t_2) \implies \neg\mathsf{OpSeq}(c_1,c_2,t_1,t_2,t_3,t_4) \quad (2)$$

$$\neg\mathsf{Ch}(c_2,t_3,t_4) \implies \neg\mathsf{OpSeq}(c_1,c_2,t_1,t_2,t_3,t_4) \quad (3)$$

$$\neg(t_2 \leq t_3) \implies \neg\mathsf{OpSeq}(c_1,c_2,t_1,t_2,t_3,t_4) \quad (4)$$

where variables c_i represent the type of a chronicle and t_j instant of time. Ch and OpSeq are predicates that respectively define when a chronicle of a certain type has been recognised and when a sequence between two chronicles is satisfied. For instance, if the predicate $\mathsf{Ch}(c_1,t_1,t_2)$ is valued to true, a recognition of a chronicle of type c_1 has started at time t_1 and terminated at time t_2. OpSeq is valued to true if two chronicles of types c_1, c_2 have been recognized at the given times and if chronicle c_2 happens after the recognition of c_1. Notice that these rules are equivalent to formula (1) with $\iff$ but usually FOL problems are presented in CNF, so we adopt this presentation here.

Once the sequence has been defined, new types of chronicle may be constructed, such as the following (where A and B are chronicles types and AB the new type that defines the sequence of two chronicles A and B):

$$\mathsf{OpSeq}(A,B,t_1,t_2,t_3,t_4) \implies \mathsf{Ch}(AB,t_1,t_4) \quad (5)$$

$$\exists t_2,t_3 \quad \neg\mathsf{OpSeq}(A,B,t_1,t_2,t_3,t_4) \implies \neg\mathsf{Ch}(AB,t_1,t_4) \quad (6)$$

3 MLN and MaxWalkSAT

3.1 Markov Logic

A first-order knowledge base (KB) is a set of clauses, which may be seen as constraints on possible worlds of predicate values. If a rule is violated by a world x, this world does not satisfy the KB. With MLNs, the latter rule is weakened, letting clauses being wrong by associating them to a weight that reflects how strong the constraint is. The higher is the weight, the more probable is the clause to be satisfied. MLNs are Markov random fields where ground predicates are vertices and clauses are cliques on the graph. MLNs are designed using Markov Logic (ML). In ML, each formula F_i is associated to a weight w_i and the probability distribution over possible worlds x is given by

$$P(X=x) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(x)\right) \quad (7)$$

where n_i is the number of true groundings of F_i and $Z = \sum_{X=x}\exp\left(\sum_i w_i n_i(x)\right)$ is the partition function. On ML, an infinite weight on a formula is equivalent to defining a hard constraint as the probability will be zero if the formula is false.

The inference with MLN relies on MC-SAT: a MCMC (Markov Chain Monte-Carlo) technique used to approximate the probability distribution. MC-SAT uses a slice sampling approach to represent the probability distribution over worlds. In this algorithm, it is necessary to be able to sample worlds from the KB. For this purpose, MC-SAT uses SampleSAT [20], to perform approximately uniform samples. SampleSAT consists on taking the solution given by any WalkSAT method, i.e a SAT solver using random search to find a solution for a FOL formula, smoothed by temperature annealing.

MLN uses MaxWalkSAT which is a weighted version of WalkSAT that allows both soft and hard constraints but the principle remains identical.

3.2 WalkSAT Strategies on Details

In their paper, [20] describe a way to sample approximately uniform solutions from a hard 3-SAT problem using a WalkSAT algorithm.

WalkSAT [14] (Algorithm 1) is an improved version of GSAT [15]: a simple procedure that tries and reaches a solution of a KB with a gradient technique that minimises the number of false clauses. WalkSAT adds a random step probability parameter that sets the chances to take a random step, i.e. flipping a random atom of the selected false clause[4], thus allowing escaping local optima.

Algorithm 1: WalkSAT(KB, m_t, m_f, p)

inputs : KB, a knowledge base in CNF
m_t, maximum number of tries
m_f, maximum number of flips
p, probability of random step
outputs: *bestSol*, the best solution found

```
for i ← 1 to m_t
    tmpSol ← a random assignment for the KB
    bestSol ← tmpSol
    for i ← 1 to m_f
        Choose a random unsatisfied clause c
        if Uniform(0, 1) < p
            flip a random atom in c
        else
            flip best atom in c
        if tmpSol better than bestSol
            bestSol ← tmpSol
return bestSol
```

4 A Simple Intractable Problem for WalkSAT

In this section, we show that using MLNs for solving a chronicle problem is intractable, and more generally, that this happens when the definition of a system with logical formulae contains too many biconditional statements.

[4] For MaxWalkSAT, the cost function is no longer the number of false clauses, but the sum of their weights.

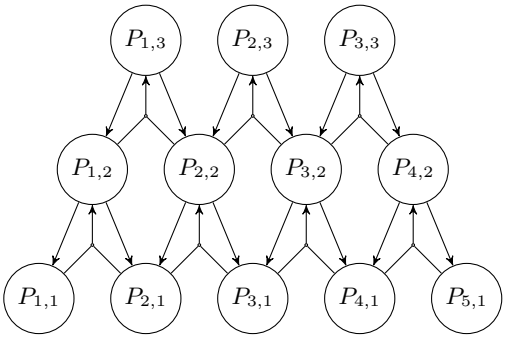

Fig. 2. A simplified chronicles representation with a height of 3 and a base length of 5

4.1 Problem Statement

For the sake of clarity, we show a simplified chronicle problem $\mathcal{F}$. As chronicles may be seen as a succession of deductions at different levels, like a truncated pyramid of deductions, we will choose the following representation:

Given a set of boolean variables $V = \{x_{1,1}, \ldots, x_{i-j+1,j}, \ldots, x_{1,k}\}$ for $j \in \{1, \ldots, k\}$ and $i \in \{1, \ldots, k-j+1\}$ and given the set of clauses $\mathcal{R} = \{c_1, \ldots, c_n\}$, the problem is designed as follows: $\mathcal{F} = \bigwedge_{c_i \in \mathcal{R}} c_i$ and

$$\begin{cases} (x_{i,j} \wedge x_{i+1,j} \implies x_{i,j+1}) \in \mathcal{R} \\ (x_{i,j+1} \implies x_{i,j}) \in \mathcal{R} \\ (x_{i,j+1} \implies x_{i+1,j}) \in \mathcal{R} \end{cases} \quad \text{with} \quad \begin{cases} 0 < j \leq k; \\ 0 < i \leq k - j + 1 \end{cases} \tag{8}$$

where j is the height of a predicates layer, i the position of a predicate on j^{th} layer and k the total number of layers. An example of this problem with a height of three and twelve nodes is shown on Figure 2. It is worth noting the similarity with the sequence operator definition provided previously on eq. 1.

To complete this problem, we add all predicates from the first layer as evidences. This simulates the recognition or non-detection of LLE on the data flow; note it will make the number of solutions drop from 2^k to 1.

4.2 Experimental Setup

For our experiments, we used MaxWalkSAT as a SAT solver implemented in Alchemy 2 [19] Our KB is designed with infinite weights on every clause, so it is supposed to produce equivalent results than WalkSAT [4]. This version uses a TABU implementation which will not let a predicate be flipped twice without n other predicates having been flipped in between[5].

We designed two experiments. In the first one, we want to set the number of nodes but keep control on the length of the first layer. So we ask MaxWalkSAT to solve, at the same time, parallel instances of $\mathcal{F}$ (eq. 8) with the first layer

[5] In our case, n was set at 10.

Height	2	3	4	5	6	7	8	9	10
Nodes per struct	3	6	10	15	21	28	36	45	55
Structures	666	333	200	133	95	71	55	44	36
Total flips	1505	3530	8265	25324	71567	406655	**980000**	**984068**	**984189**
Percentage solved	100%	100%	100%	100%	100%	100%	50%	10%	0%

Table 1. Number of flip before depending on the height and length of the structure with 2000 nodes.

length equal to the structure height. We set the nodes number to the closest value around 2000 considering previous constraints. This experiment could be seen as a chronicle problem where many distinct chronicles are being recognised on the same data flow.

In the second experiment, we wanted to study the impact of the length of first layer on the resolution time with a fixed height. We launch MaxWalkSAT on a single instance of a problem with height 4.

On both problems, MaxWalkSAT stops if it finds a solution or reaches one million flips[6].

4.3 Experimental Results

Table 1 shows the experiment results. The total number of needed flips quickly rises with the structure size growing, and MaxWalkSAT stops finding the optimal solution as soon as the height structure reaches 8. It could seem normal, given that, at constant nodes number, the number of clauses goes up, but keep in mind that this structure is not random and it is just made of repetitions of the same pattern. Usually, random strategies find all solutions for problems of this size. For instance [20], found all solutions for problems with more than 20 000 clauses and almost 5 000 variables.

We have indicated the percentage of substructures solved even if the whole problem is not, because random strategies can restart from the beginning (number of tries[7] in alg. 1). But we can see that, when reaching a height of ten, restart is not even useful since no structure is solved.

Unfortunately, the algorithm stops with a really low amount of nodes but, looking only at the steps where all the problem is fully solved, it is interesting to note that, per structure, the number of flips is worse in average than a pure random walk strategy on 3SAT known to be exponential regarding the number of nodes, namely $O(1.334^n)$ [12].

One could think height is the main problem, since the algorithm has then to propagate truth along longer deduction layers. So on our second experiment on

[6] In fact, the solver will stop before reaching the million flips because sometimes, when the algorithm has to flip the best atom, the flip does not occur if it leads to a worse solution than the current one, but this still counts as a flip. Our results just consider *efficient* flips, *i.e.* times when the value of a variable actually changes.

[7] With MLNs, the number of tries is usually set to 1 due to the cost on the algorithm.

Base length	10	20	25	30	40	50
Total nodes	34	74	94	114	154	194
Total clauses	92	192	232	292	392	592
Flips	6450	62394	386969	**864892**	**886892**	**898827**

Table 2. Number of flips needed to solve with the height fixed to 4.

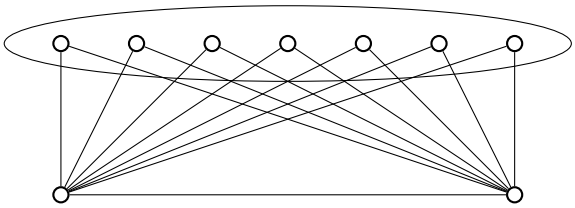

Fig. 3. An instance with strong difference on the node degree

tab. 2, we designed the same case than previously but with dependencies between the structures. Even if we showed that problems with numerous instances of height 4 were easily solved, we have evidenced that, at constant height, the problem becomes quickly intractable; especially if deductions shared a lot of common atoms, letting a predicate having an even small influence on all others.

In the light of the considerations above, it seems that the general structure has as much impact as the number of double implications in the formula.

5 Discussion

5.1 General discussion

In this section, we discuss the reasons that make random walk strategies inefficient when dealing with this kind of logical structure. Structural problems have already been isolated like in [13] where GSAT has shown poor experimental results on specific situations. For instance, in a graph colouring problem using only three colours, if the graph presents nodes of a comparatively higher degree, GSAT tends to be stuck. An instance is shown on Figure 3 where GSAT will often be stuck with the two bottom nodes assigned to the same colour. WalkSAT has been introduced in this same article to solve this problem.

But we have shown that, even in balanced cases, a computational problem may arise, especially when a FOL formula is designed with clauses and their inverse. This substructure allows the solver to flip the same atom a huge number of times without being able to determine when it leads to an improvement.

In a local search, decision is completely correlated to the context around the atom to flip. For instance, in our problem, a node has 18 neighbours, which define the context. It is interesting to know that, on the 2^{18+1} contexts, the WalkSAT algorithm has a 46.2% chance of flipping the atom of interest from the good configuration to the wrong one, and a 36.6% chance the other way around.

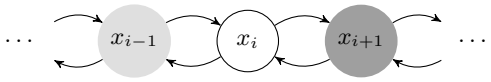

Fig. 4. Conflicting propagation of truth based on local search

If we choose a FOL formula mostly designed with rules of the following kind: $x_1 \wedge \cdots \wedge x_n \iff y$, it CNF will have one clause of size $n+1$ and n clauses of size 2. It is easy to see that in this configuration there are $2^{n+1}-1$ solutions where y is false and only one when y is true. And if y is used for higher deductions of this kind, putting y to false has a cost only if atoms on the above layer have been already set to true. This make usually WalkSAT more likely to put y to false. More simply, this sub-problem is almost equivalent to the highlighted structure by Selman and Kautz, which is easily solved by WalkSAT. But in our problem, an additional difficulty may account for the poor performance of WalkSAT. Indeed, the structure is repeated many times, with its high-degree nodes serving as low-degree nodes in several higher instances of the structure: this intrication is likely to cause a drop in performance. Assume for instance that such a node of degree n occurs also with a low degree in k structures. There will be $n + k$ clauses where it will occur negatively and $k + 1$ clauses where it will occur positively: consequently, the local search will show a structural tendency to set it to false.

Independently from the structure, double implications are intuitively problematic to solve with a local search. Usually, a deductive system tends to propagate truth values. But, with local search, tracks of this propagation are lost, and the algorithm is only guided by the number of truth values around the node[8]. Figure 4 is a simple but clear case of the inefficiency of WalkSAT on double implications. In this situation, where x_{i-1} is valued true (light gray) and x_{i+1} false (dark gray), determining the value of x_i will be random and meaningless.

5.2 Extension of the Problem to MLN

We expressed at the beginning of this article that local search methods lead MLN to give poor probabilities for deductions problems. The reason is straightforward: as the WalkSAT method cannot reach solutions that require a high level of deduction, using it as sampler would result in worlds with only few layers of the deduction consistent with the evidence. Therefore, the sample sets produced for MCSAT will not be reliable enough to produce trustworthy results.

6 Conclusion

Our first intent was to improve chronicles to let them handle uncertainties. But we have shown in this paper that MLNs can not perform well when applied on such problems. This is not due to the MCMC technique they use, but on a deeper

[8] Especially the number of nodes with a certain truth value.

problem correlated to the SAT domain. We have highlighted that specific logical problems designed with a lot of biconditional statements may lead local search techniques to be stuck, even on problems with small dimensions compared to other works where these methods have been used.

In the Max-SAT 2016 competition, many benchmarks — whether crafted or industrial — are designed that way. Considering the performances variations between complete and local methods, the highlighted problem might be a reason explaining these differences. We know that many parameters impact the efficiency of SAT-solvers, like the number of solutions or the diameter of the associated neighbourhood graph [6], so, obviously, inner structures might not be the only reason, but seem to be an interesting lead.

Even if complete methods have better results, they still have difficulties to solve large problems and local algorithms are still needed for many tasks. But, for future works, this inner structural problem should be taken in consideration when we design and use local search methods. To help with this design, it would also probably be useful to quantify somehow difficulties linked to structure repetitions evoked in 5.1. Some other candidates for local search would also have to be assessed in the context of MLN: some algorithms on the Max-SAT competition have sometimes really good results on benchmarks that seem designed as chronicles. It might be interesting to investigate them.

Another possibility will be to look at complete methods; latest improvements on the field made algorithms almost as fast as local search. However, in the context of MLN where it is necessary to make MCMC calculations to compute probabilities, local search methods have a big asset, as they provide approximately uniform samples [20]. On the other hand, to our knowledge, the question whether complete methods also provide such samples has not been studied yet, probably as complete methods used to be too computation time-demanding to be even considered in this context, and would have to be assessed beforehand.

Finally, there are other approaches like the probabilistic logic programming that mix probabilities and logical formulae, Problog [5] for instance. WalkSAT algorithms are sometimes used there too, but many probabilistic logic programming approaches make use of complete methods instead. Hence, investigating these approaches would be an interesting lead to solve the problem arisen here, and we plan to tackle them in future work.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
2. Biswas, R., Thrun, S., Fujimura, K.: Recognizing activities with multiple cues. In: Human Motion–Understanding, Modeling, Capture and Animation, pp. 255–270. Springer (2007)
3. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM 5(7), 394–397 (1962)
4. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for Artificial Intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning 3(1), 1–155 (2009)

5. Fierens, D., Broeck, G.V.d., Thon, I., Gutmann, B., De Raedt, L.: Inference in probabilistic logic programs using weighted CNF's. Theory and Practice of Logic Programming 15(03), 258–401 (2012)
6. Hoos, H.H., Stützle, T.: Stochastic local search: foundations and applications. Morgan Kaufmann Publishers (2005)
7. Kembhavi, A., Yeh, T., Davis, L.S.: Why did the person cross the road (there)? Scene understanding using probabilistic logic models and common sense reasoning. In: Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II. LNCS, vol. 6312, pp. 693–706. Springer (2010)
8. Ornato, M., Carle, P.: Reconnaissance d'intentions sans reconnaissance de plan. 2es Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents p. 29 (1994)
9. Piel, A.: Reconnaissance de comportements complexes par traitement en ligne de flux d'evenements. Ph.D. thesis, U. Paris 13 (2014)
10. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
11. Sadilek, A.: Modeling human behavior at a large scale. Ph.D. thesis, Rochester University (2012)
12. Schoning, T.: A probabilistic algorithm for k-SAT and constraint satisfaction problems. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science. pp. 410–414. IEEE (1999)
13. Selman, B., Kautz, H.: Domain-independent Extensions to GSAT: Solving Large Structured Satisfiability Problems. In: Proceedings of the 13th International Joint Conference on Artifical Intelligence. pp. 290–295. Morgan Kaufmann Publishers Inc. (1993)
14. Selman, B., Kautz, H., Cohen, B., others: Local search strategies for satisfiability testing. In: DIMACS Series in Discrete Mathematics. vol. 26, pp. 521–532 (1993)
15. Selman, B., Levesque, H.J., Mitchell, D.G., others: A New Method for Solving Hard Satisfiability Problems. In: Proceedings of the 10th National Conference on Artificial Intelligence. vol. 92, pp. 440–446 (1992)
16. Skarlatidis, A., Artikis, A., Filippou, J., Paliouras, G.: A probabilistic logic programming event calculus. Theory and Practice of Logic Programming 15(02), 213–245 (2015)
17. Skarlatidis, A., Paliouras, G., Vouros, G.A., Artikis, A.: Probabilistic event calculus based on markov logic networks. In: Proceedings of Rule-Based Modeling and Computing on the Semantic Web. LNCS, vol. 7018, pp. 155–170. Springer (2011)
18. Snidaro, L., Visentini, I., Bryan, K.: Fusing uncertain knowledge and evidence for maritime situational awareness via Markov Logic Networks. Information Fusion 21, 159–172 (2015)
19. Sumner, M., Domingos, P.: The alchemy tutorial (2010)
20. Wei, W., Erenrich, J., Selman, B.: Towards efficient sampling: Exploiting random walk strategies. In: Proceedings of the 19th National Conference on Artificial Intelligence. pp. 670–676 (2004)