Stochastic Refinement

Alireza Tamaddoni-Nezhad and Stephen Muggleton

Department of Computing, Imperial College London
{atn,shm}@doc.ic.ac.uk

Abstract. The research presented in this paper is motivated by the following question. How can the generality order of clauses and the relevant concepts such as refinement be adapted to be used in a stochastic search? To address this question we introduce the concept of stochastic refinement operators and adapt a framework, called stochastic refinement search. In this paper we introduce stochastic refinements of a clause as a probability distribution over a set of clauses. This probability distribution can be viewed as a prior in a stochastic ILP search. We study the properties of a stochastic refinement search as two well known Markovian approaches: 1) Gibbs sampling algorithm and 2) random heuristic search. As a Gibbs sampling algorithm, a stochastic refinement search iteratively generates random samples from the hypothesis space according to a posterior distribution. We show that a minimum sample size can be set so that in each iteration a consistent clause is generated with a high probability. We study the stochastic refinement operators within the framework of random heuristic search and use this framework to characterise stochastic search methods in some ILP systems. We also study a special case of stochastic refinement search where refinement operators are defined with respect to subsumption order relative to a bottom clause. This paper also provided some insights to explain the relative advantages of using stochastic lgg-like operators as in the ILP systems Golem and ProGolem.

1 Introduction

Most ILP systems are traditionally based on clause refinement through a lattice defined by a generality order (e.g. subsumption). However, there is a long-standing and increasing interest in stochastic search methods in ILP for searching the space of candidate clauses (e.g. [11,15,12,18,10,8]). The research presented in this paper is motivated by the following question. How can the generality order of clauses and the relevant concepts such as refinement be adapted to be used in a stochastic search? To address this question we introduce the concept of stochastic refinement operators and adapt a framework, called stochastic refinement search.

Refinement of a clause is defined as a set of clauses. In this paper we introduce stochastic refinement of a clause as a probability distribution over a set of clauses. This probability distribution can be viewed as a prior in a stochastic ILP search. In this paper we define the concept of stochastic refinement search. In general a stochastic refinement search can be viewed as a Markov chain. We study the properties of a stochastic refinement search as two well known Markovian approaches: 1) Gibbs sampling algorithm and 2) random heuristic search. As a Gibbs sampling algorithm, a stochastic refinement search iteratively generates random samples from the hypothesis space according to a posterior distribution. We have shown that a minimum sample size can be set so that in each iteration a consistent clause is generated with a high probability.

We define a special case of random heuristic search [17] called monotonic random heuristic search and then we show that due to the generality order defined by the refinement operators, a stochastic refinement search can be viewed as a monotonic random heuristic search. The advantage of studying stochastic refinement search as a random heuristic search is that we can use the theoretical results from random heuristic search in order to analyse the behaviour and convergence of the search. We also study a special case of stochastic refinement search where refinement operators are defined with respect to subsumption order relative to a bottom clause [16]. This paper also provides some theoretical insights to explain the relative advantages of using stochastic lgg-like operators as in the ILP systems Golem and ProGolem.

This paper is organised as follows. In Section 2 we review some of the basic concepts from ILP which are used in the definitions and theorems in this paper. In Section 3 stochastic refinement operators are introduced and their properties are discussed. The framework of stochastic refinement search is discussed in Section 4 and this framework is used to characterise stochastic search methods in some ILP systems. In Section 5 we consider a special case where a stochastic search is used to explore a refinement graph bounded by a most specific (bottom) clause. Section 6 concludes the paper.

2 Preliminaries

We assume the reader to be familiar with the basic concepts from logic programming and inductive logic programming [9]. This section is intended as a brief reminder of some of the concepts used in definitions and theorems in this paper.

The general subsumption order on clauses, also known as θ-subsumption, is defined as follows.

Definition 1 (Subsumption). *Let C and D be clauses. We say C subsumes D, denoted by $C \succeq D$, if there exists a substitution θ such that $C\theta$ is a subset of D. C properly subsumes D, denoted by $C \succ D$, if $C \succeq D$ and $D \not\succeq C$. C and D are subsume-equivalent, denoted by $C \sim D$, if $C \succeq D$ and $D \succeq C$.*

Remark 1. Let $\mathcal{C}$ be a set of first order clauses and $\succeq$ be the subsumption order as defined in Definition 1. Every finite subset of $\mathcal{C}$ has a *most general specialisation (mgs)* and a *least general generalisation (lgg)*. Thus $\langle \mathcal{C}, \succeq \rangle$ is a lattice.

The following definition is a reminder of the concept of unary refinement operators and related properties.

Definition 2 (Unary refinement operator). *Let $\langle G, \succeq \rangle$ be a quasi-ordered set. A (downward)* unary refinement operator *for $\langle G, \succeq \rangle$ is a function $\rho : G \rightarrow 2^G$, such that $\rho(C) \subseteq \{D | C \succeq D\}$, for every $C \in G$.*

- *The sets of* one-step refinements, n-step refinements *and* refinements *of some $C \in G$ are respectively: $\rho^1(C) = \rho(C)$, $\rho^n(C) = \{D|$ there is an $E \in \rho^{n-1}(C)$ such that $D \in \rho(E)\}, n \geq 2$ and $\rho^*(C) = \rho^1(C) \cup \rho^2(C) \cup ..$*
- *A ρ-*chain *from C to D is a sequence $C = C_0, C_1, \ldots, C_n = D$, such that $C_i \in \rho(C_{i-1})$ for every $1 \leq i \leq n$.*
- *ρ is* locally finite *if for every $C \in G$, $\rho(C)$ is finite and computable.*
- *ρ is* proper *if for every $C \in G$, $\rho(C) \subseteq \{D|C \succ D\}$.*
- *ρ is* complete *if for every $C, D \in G$ such that $C \succ D$, there is an $E \in \rho^*(C)$ such that $D \sim E$ (i.e. D and E are equivalent in the $\succeq$-order).*
- *ρ is* weakly complete *if $\rho^*(\Box) = G$, where $\Box$ is the top element of G.*
- *ρ is* non-redundant *if for every $C, D, E \in G$, $E \in \rho^*(C)$ and $E \in \rho^*(D)$ implies $C \in \rho^*(D)$ or $D \in \rho^*(C)$.*
- *ρ is* ideal *if it is locally finite, proper and complete.*
- *ρ is* optimal *if it is locally finite, non-redundant and weakly complete.*

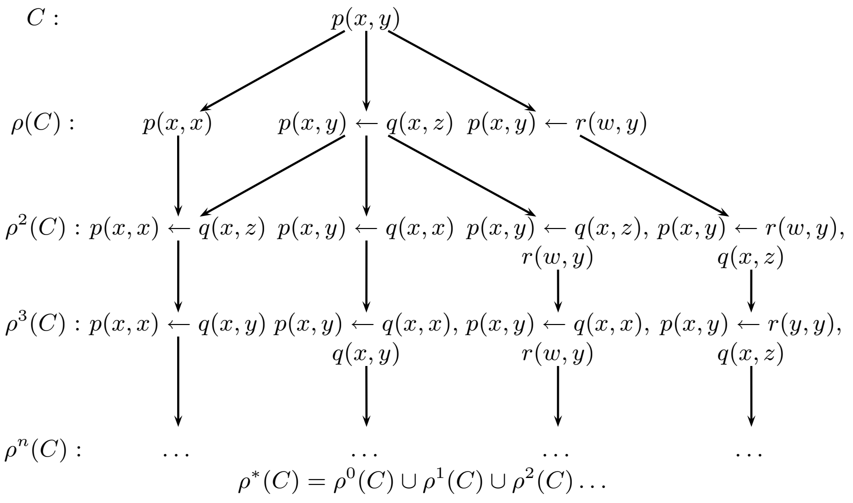
We can define analogous concepts for the dual case of an upward unary refinement operator.

Example 1. Figure 1 shows part of a (downward) unary refinement graph for the subsumption order. In this graph clause $p(x, y)$ is refined either by unifying variables or by adding literals. The refinement operator presented by this graph is not complete as it does not include all possible refinements. It is proper as the graph does not contain cycles. It is redundant because it does not have a tree structure and there is more than one path from $p(x, y)$ to $p(x, x) \leftarrow q(x, z)$.

The following definition for binary refinement is adapted from [5].

Definition 3 (Binary refinement operator). *Let $\langle G, \succeq \rangle$ be a quasi-ordered set. A (downward)* binary refinement operator *for $\langle G, \succeq \rangle$ is a function $\rho : G^2 \rightarrow 2^G$, such that $\rho(C, D) \subseteq \{E|C \succeq E, D \succeq E\}$, for every $C \in G$.*

- *The sets of* one-step refinements, n-step refinements *and* refinements *of some $C \in G$ are respectively: $\rho^1(C, D) = \rho(C, D)$, $\rho^n(C, D) = \{E|$ there is an $F \in \rho^{n-1}(C, D)$ and an $H \in \rho^{n-1}(C, D)$ such that $E \in \rho(F, H)\}, n \geq 2$ and $\rho^*(C, D) = \rho^1(C, D) \cup \rho^2(C, D) \cup ..$*
- *A ρ-*chain *(C, D) to E is a sequence $(C, D) = (C_0, D_0), (C_1, D_1), \ldots, (C_m, D_m)$, such that $E = C_m$ or $E = D_m$ and $C_i, D_i \in \rho(C_{i-1}, D_{i-1})$ for every $1 \leq i \leq m$.*
- *ρ is* locally finite *if for every $C, D \in G$, $\rho(C, D)$ is finite and computable.*
- *ρ is* proper *if for every $C, D \in G$, $\rho(C, D) \subseteq \{E|C \succ E, D \succ E\}$.*
- *ρ is* complete *if for every $B, C, D \in G$ such that $C \succ B$, $D \succ B$ there is an $E \in \rho^*(C, D)$ such that $B \sim E$ (i.e. B and E are equivalent in the $\succeq$-order).*
- *ρ is* weakly complete *for $\langle G, \succeq \rangle$ if $\rho^*(\Box, \Box) = G$, where $\Box$ is the top element of G.*



$$\begin{tikzcd}
C: & & p(x,y) \arrow[dl] \arrow[d] \arrow[dr] & & \\
\rho(C): & p(x,x) \arrow[d] & p(x,y) \leftarrow q(x,z) \arrow[dl] \arrow[d] \arrow[dr] & p(x,y) \leftarrow r(w,y) \arrow[dr] & \\
\rho^2(C): & p(x,x) \leftarrow q(x,z) \arrow[d] & p(x,y) \leftarrow q(x,x) \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,z), \\ r(w,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow r(w,y), \\ q(x,z) \end{array} \arrow[d] \\
\rho^3(C): & p(x,x) \leftarrow q(x,y) \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,x), \\ q(x,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow q(x,x), \\ r(w,y) \end{array} \arrow[d] & \begin{array}{c} p(x,y) \leftarrow r(y,y), \\ q(x,z) \end{array} \arrow[d] \\
\rho^n(C): & \ldots & \ldots & \ldots & \ldots
\end{tikzcd}$$

$$\rho^*(C) = \rho^0(C) \cup \rho^1(C) \cup \rho^2(C) \ldots$$

Fig. 1. Part of a unary refinement graph representing (downward) refinements of a clause

- ρ *is* non-redundant *if for every* $B, C, D, E, F \in G$, $F \in \rho^*(B, C)$ *and* $F \in \rho^*(D, E)$ *implies* $B, C \in \rho^*(D, E)$ *or* $D, E \in \rho^*(B, C)$.
- ρ *is* ideal *if it is locally finite, proper and complete.*
- ρ *is* optimal *if it is locally finite, non-redundant and weakly complete.*

We can define analogous concepts for the dual case of an upward binary refinement operator.

3 Stochastic Refinement Operators

In this section we introduce the concept of stochastic refinement operators. According to Definition 2 (and Definition 3), refinement of a clause (or a pair of clauses) is a set of clauses. In the following we define stochastic refinement as a probability distribution over a set of clauses.

Definition 4 (Stochastic unary refinement operator). *Let* ρ *be a (downward) unary refinement operator for the quasi-ordered set* $\langle G, \succeq \rangle$ *and* $C \in G$. *A (downward)* stochastic unary refinement operator *is a function* $\sigma : G \rightarrow 2^{G \times [0,1]}$ *defined as follows:* $\sigma(C) = \{\langle D_i, p_i \rangle | D_i \in \rho(C),\ p_i \in [0,1]$ *and* $\sum p_i = 1$ *for* $1 \leq i \leq |\rho(C)|\}$.

- *A* ***σ-chain*** *from* C *to* D, *denoted by* $C \xrightarrow{\sigma} D$, *is a sequence* $C = C_0, C_1, \ldots, C_m = D$, *such that* $\langle C_i, p_i \rangle \in \sigma(C_{i-1})$ *for every* $1 \leq i \leq m$ *and the probability of this* σ-chain *is* $p(C \xrightarrow{\sigma} D) = \prod_{i=1}^{m} p_i$.
- ***n-step stochastic refinements*** *of* C *is defined as:* $\sigma^n(C) = \{\langle D, p \rangle | D \in \rho^n(C)$ *and* $p = \sum_{x \in X} p(x)$ *where* X *is the set of* σ*-chains from* C *to* $D\}$.

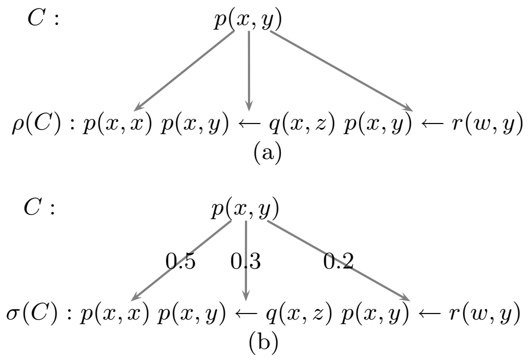

C :
$p(x,y)$
$\rho(C) : p(x,x)$ $p(x,y) \leftarrow q(x,z)$ $p(x,y) \leftarrow r(w,y)$
(a)
C :
$p(x,y)$
0.5
0.3
0.2
$\sigma(C) : p(x,x)$ $p(x,y) \leftarrow q(x,z)$ $p(x,y) \leftarrow r(w,y)$
(b)

Fig. 2. (a) Refinement of a clause is defined as a set of clauses (b) Stochastic refinement of a clause is defined as a probability distribution over a set of clauses

– ***stochastic refinements*** *of C is defined as:* $\sigma^*(C) = \{\langle D_i, p_i\rangle | D_i \in \rho^*(C)$, $p_i \in [0,1]$ *and* $\sum p_i = 1$ *for* $1 \leq i \leq |\rho^*(C)|\}$.
– *σ inherits the standard properties (i.e. local finiteness, properness and completeness) from ρ.*

Example 2. Figure 2.a shows refinement of a clause as a set of clauses. Stochastic refinement of a clause is defined as a probability distribution over a set of clauses (Figure 2.b).

Example 3. Figure 3 shows part of a stochastic refinement graph. This graph shows the probabilities of clauses in $\sigma^n(C)$ as defined in Definition 4. For example, there are two σ-chains from C to D_4. Hence, the probability of D_4 in $\sigma^2(C)$ is $0.5 \times 1.0 + 0.3 \times 0.4 = 0.62$.

According to Definition 4, $\sigma(C)$ and $\sigma^*(C)$ represent probability distributions. In the following we show that $\sigma^n(C)$ is also a probability distribution.

Theorem 1. *n-step stochastic refinements of a clause represent a probability distribution.*

Proof. Let $\sigma^n(C)$ be n-step stochastic refinements of clause C as defined in Definition 4 such that $\sigma^n(C) = \{\langle D_1, p_1\rangle, \langle D_2, p_2\rangle, \ldots \langle D_m, p_m\rangle\}$. We will show that $\sum_{i=1}^{m} p_i = 1$. The proof is by induction on n. For $n = 1$ the theorem is true by definition of $\sigma(C)$. Assume that the theorem is true for k then the sum of probabilities $p_1, p_2, \ldots, p_s$ at level k is 1: $\sum_{i=1}^{s} p_i = 1$. Suppose that each node with probability p_i at level k is extended into t nodes with probabilities $q_{i1}, q_{i2}, \ldots, q_{it}$. Then the sum of probabilities at level $k+1$ will be: $\sum_{i=1}^{s}\sum_{j=1}^{t} p_i q_{ij} = \sum_{i=1}^{s} p_i(q_{i1} + q_{i2} + \cdots + q_{it})$. But $q_{i1} + q_{i2} + \cdots + q_{it} = 1$ and $\sum_{i=1}^{s} p_i = 1$ and therefore the sum of probabilities at level $k+1$ will be 1 and this completes the proof. $\square$

In the following we define analogous concepts for stochastic binary refinement operators.

$$\begin{tikzcd}
C: & & C \arrow[dl, "0.5"'] \arrow[d, "0.3"] \arrow[dr, "0.2"] & & \\
\sigma(C): & \langle D_1, 0.5\rangle \arrow[d, "1.0"] & \langle D_2, 0.3\rangle \arrow[dl, "0.4"'] \arrow[d, "0.4"] \arrow[dr, "0.2"] & \langle D_3, 0.2\rangle \arrow[dr, "1.0"] & \\
\sigma^2(C): & \langle D_4, 0.62\rangle \arrow[d] & \langle D_5, 0.12\rangle \arrow[d] & \langle D_6, 0.06\rangle \arrow[d] & \langle D_7, 0.2\rangle \arrow[d] \\
\sigma^n(C): & \dots & \dots & \dots & \dots
\end{tikzcd}$$

Fig. 3. Part of a stochastic refinement graph

Definition 5 (Stochastic binary refinement operator). *Let ρ be a (downward) binary refinement operator for the quasi-ordered set $\langle G, \succeq\rangle$ and $C, D \in G$. A (downward)* stochastic binary refinement operator *is a function $\sigma : G^2 \to 2^{G\times[0,1]}$ defined as: $\sigma(C, D) = \{\langle E_i, p_i\rangle | E_i \in \rho(C, D),\ p_i \in [0, 1]$ and $\sum p_i = 1$ for $1 \le i \le |\rho(C, D)|\}$.*

– *A **σ-chain** from (C, D) to E, denoted by $(C, D) \xrightarrow{\sigma} E$, is a sequence $(C, D) = (C_0, D_0), (C_1, D_1), \dots, (C_m, D_m)$, such that $E = C_m$ or $E = D_m$ and $\langle C_i, p_i\rangle, \langle D_i, q_i\rangle \in \sigma(C_{i-1}, D_{i-1})$ for every $1 \le i \le m$ and the probability of this σ*-chain *is $p((C, D) \xrightarrow{\sigma} E) = \prod_{i=1}^{m} p_i q_i$.*

– ***n-step stochastic binary refinements** of some $(C, D) \in G^2$ is defined as: $\sigma^n(C, D) = \{\langle E, p\rangle | E \in \rho^n(C, D)$ and $p = \sum_{x\in X} p(x)$ where X is the set of σ-chains from (C, D) to $E\}$.*

– ***stochastic binary refinements** of some $(C, D) \in G^2$ is defined as: $\sigma^*(C, D) = \{\langle E_i, p_i\rangle | E_i \in \rho^*(C, D),\ p_i \in [0, 1]$ and $\sum p_i = 1$ for $1 \le i \le |\rho^*(C, D)|\}$.*

– *σ inherits the standard properties (i.e. local finiteness, properness and completeness) from ρ.*

Example 4. Figure 4 shows stochastic binary refinement of a pair of clauses as a probability distribution over a set of clauses.

As for $\sigma^n(C)$, it can be shown that $\sigma^n(C, D)$ is a probability distribution.

Theorem 2. *n-step stochastic binary refinements of a pair of clauses represent a probability distribution.*

Proof. Let $\sigma^n(C, D)$ be n-step stochastic binary refinements of clause C and D as defined in Definition 5 such that $\sigma^n(C, D) = \{\langle D_1, p_1\rangle, \langle D_2, p_2\rangle, \dots \langle D_m, p_m\rangle\}$. We will show that $\sum_{i=1}^{m} p_i = 1$. The proof is similar to the proof of Theorem 1 except that we need to show that the sum of probabilities at level $k + 1$ will be: $\sum_{i=1}^{s}\sum_{j=1}^{t}\sum_{l=1}^{u} p_i q_j r_{ijl} = \sum_{i=1}^{s}\sum_{j=1}^{t} p_i q_j (r_{ij1} + r_{ij2} + \cdots + r_{iju})$ where the binary refinement of two nodes with probabilities p_i and q_j at level k is extended into u nodes with probabilities $r_{ij1}, r_{ij2}, \dots, r_{iju}$. But $r_{ij1} + r_{ij2} + \cdots + r_{iju} = 1$ and $\sum_{i=1}^{s}\sum_{j=1}^{t} p_i q_j = \sum_{i=1}^{s} p_i (q_1, q_2, \dots, p_t) = \sum_{i=1}^{s} p_i = 1$ and therefore the sum of probabilities at level $k + 1$ will be 1 and this completes the proof. $\square$

$$\begin{tikzcd}
(C,D): & & C \arrow[dl] \arrow[dr] \arrow[drrr] & & D \arrow[dllll] \arrow[dl] \arrow[dr] & \\
\sigma(C,D): & \langle E_1, 0.4\rangle & & \langle E_2, 0.4\rangle & & \langle E_3, 0.2\rangle
\end{tikzcd}$$

Fig. 4. Stochastic binary refinement of a pair of clauses is defined as a probability distribution over a set of clauses

4 Stochastic Refinement Search

Different stochastic search methods have been used to explore the space of candidate clauses in ILP. Examples of these search methods are simulated annealing (e.g.[11,13]), genetic algorithms (e.g. [15]) and randomised restarted search (e.g. [18]). In this section we define a general framework which can be used to characterise some of stochastic search methods in ILP.

Definition 6 (Stochastic refinement search). *Let G and σ be defined as in Definition 4 and S_0 be a sample from G with size s. A stochastic refinement search involves a sequence $S_0 \xrightarrow{\sigma} S_1 \xrightarrow{\sigma} S_2 \xrightarrow{\sigma} \ldots$ where S_{i+1} is generated from S_i such that for each $C_{i+1} \in S_{i+1}$ there exists $C_i \in S_i$ such that $\langle C_{i+1}, p\rangle \in \sigma(C_i)$. Similarly a stochastic refinement search can be defined for a stochastic binary refinement operator σ (Definition 5) where for each $C_{i+1} \in S_{i+1}$ there exist C_i and $D_i \in S_i$ such that $\langle C_{i+1}, p\rangle \in \sigma(C_i, D_i)$.*

Figure 5 shows stochastic refinement search with unary and binary stochastic refinement operators. In Definition 6, the initial sample S_0 corresponds to the starting point of the search as in simulated annealing or an initial population as in genetic algorithms. These are usually generated randomly. The sample size s is equal to one in simulated annealing and it is the population size in a genetic algorithm. Stochastic refinement operator σ corresponds to task-specific operators or search transition rules in an stochastic ILP search algorithm. These operators generate new clauses from the clauses in the previous search state or population. For example, in the simulated annealing search used in ILP (e.g.[11,13]), the transition operators can be viewed as downward stochastic unary refinement operators which stochastically choose the next literal to be added to the body of a given clause. Crossover operators of a genetic algorithm search used in ILP (e.g. [15]) can be viewed as stochastic binary refinement operators. Note that some stochastic refinement searches (e.g. genetic algorithms) use both stochastic unary and binary refinement operators.

According to Definition 6 a stochastic refinement search is a sequence of random samples with the property that the current state depends only on the previous state. Hence, in general a stochastic refinement search can be viewed as a Markov chain. In this section we study the properties of a stochastic refinement search as two well known Markovian approaches: 1) a Gibbs sampling algorithm and 2) a random heuristic search.

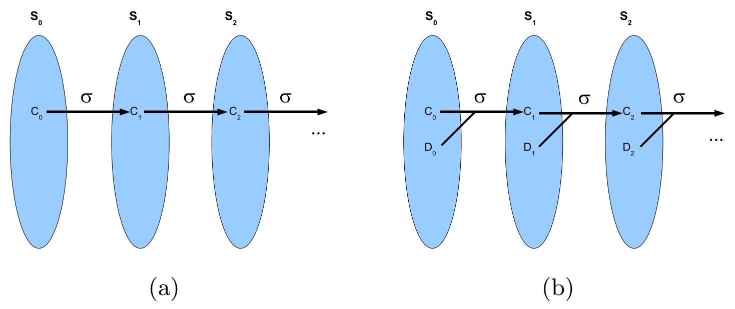


Fig. 5. A stochastic refinement search with (a) unary stochastic refinement operator (b) binary stochastic refinement operator

4.1 Stochastic Refinement Search as a Gibbs Sampling Algorithm

As noted in [8], some stochastic machine learning algorithms can be viewed as Gibbs-MAP algorithms. According to [3], most machine learning algorithms can be viewed as approximations to the following general Bayesian statistical inference algorithms.

MAP - returns the hypothesis having maximum posterior probability.
Gibbs - randomly samples hypotheses according to the posterior distribution.
Bayes Prediction - classifies unseen instances based on weighted joint prediction of the entire hypothesis space.

All of the above assume a Bayes' prior distribution over the hypothesis space. In the case of Gibbs this is used for sampling. Stochastic refinement search introduced in this paper assumes a prior distribution over the hypothesis space which is defined by stochastic refinement operators. A stochastic refinement search can be viewed as a Gibbs-MAP algorithm. A Gibbs-MAP algorithm is a Gibbs-like approximations to MAP based on sampling. It has been shown (e.g. [1]) that the sequence of samples in a Gibbs sampling algorithm constitutes a Markov chain. It has also been shown [2] that average-case error bounds for Gibbs algorithms are comparable to other Bayesian algorithms. A stochastic refinement search is a Gibbs-MAP algorithm which is aimed at maximising posterior probability by iteratively generating new samples from the hypothesis space. In a stochastic refinement search each new sample generated from the previous sample using stochastic refinement operators. There is a trade-off between the efficiency and accuracy which can be controlled by the sample size. In this section we show how with a proper sample size one could guarantee that with a high probability a consistent and compressive clause can be generated at each generation. We give examples of stochastic refinement search methods with unary and binary refinement operators.

The Quick Generalisation (QG) algorithm described in [8] is a Gibbs-MAP algorithm, which by construction, generates consistent clauses (by stochastically pruning Progol bottom clauses). A sampling algorithm based on the QG algorithm returns the clause with highest positive compression from a sample of s calls to QG. In the QG sampling algorithm described in [8], the sample size s is set by the user so that at least one of the clauses has positive compression. In this setting, the algorithm simply returns a consistent clause with the highest positive compression. The QG sampling algorithm can be viewed as a stochastic refinement search with unary refinement operator which randomly samples from "fringe" clauses (i.e. maximally general consistent clauses in the hypothesis space). As shown in [8], a minimum sample size for a QG sampling algorithm can be estimated based on the percentage of consistent clauses which have positive compression.

In the following we show that a minimum sample size can be also estimated for a stochastic refinement search with a binary operator. Golem [4] and ProGolem [6] can be viewed as stochastic refinement search methods which use lgg-like binary operators. Unlike in QG, the stochastic refinements in Golem and ProGolem do not guarantee to generate consistent clauses. However, the following theorem shows how the sample size and the probability of generating a consistent clause are related. This can be used to set a minimum sample size such that with a high probability at least one consistent *lgg* is generated.

Theorem 3. *Let k be an upper bound on the number of clauses in a target theory, c be a natural number and E^+ and E^- be the set of positive and negative examples respectively. Suppose that $s = ck$ pairs of clauses are randomly sampled from E^+. Then the probability that there is a pair of clauses C_1 and C_2, such that $lgg(C_1, C_2)$ is consistent with E^-, is at least $1 - e^{-c}$.*

Proof. First suppose that the k clauses in the target theory have disjoint coverage and each covers the same number of clauses. In this case, there are k partitions each covered by one clause in the target theory. The probability that a randomly selected pair of clauses belong to the same partition is $\frac{1}{k}$. If we randomly sample $s = ck$ pairs of clauses, the probability that there is no pair of clauses belonging to the same partition is $(1 - \frac{1}{k})^{ck}$. If the coverages are of different sizes then the probability that there is not a pair of clauses covered by the same clause in the target theory is less than $(1 - \frac{1}{k})^{ck}$. This is because the product of the probabilities are maximum if the coverages have the same size. Similarly, if the coverages are not disjoint then the probability that there is not a pair of clauses covered by the same clause in the target theory is less than $(1 - \frac{1}{k})^{ck}$. Also note that the maximum value for $(1 - \frac{1}{k})^{ck}$ is $\lim_{k \to \infty}(1 - \frac{1}{k})^{ck} = e^{-c}$. Then the probability that a randomly selected pair of clauses C_1 and C_2 are both covered by the same target clause is at least $1 - e^{-c}$. But if C_1 and C_2 are both covered by the same target clause C then $lgg(C_1, C_2)$ is consistent because by definition $lgg(C_1, C_2)$ is more specific than C which is consistent. Hence, the probability that there is a pair of clauses C_1 and C_2 such that $lgg(C_1, C_2)$ is consistent is at least $1 - e^{-c}$. □

Example 5. Consider the Golem algorithm as described in [4]. Suppose that the upper bound on the number of clauses in the target theory is k. Then according to Theorem 3, by selecting a minimum sample size $s = 2k$, the probability that a consistent lgg is generated is at least $1 - e^{-2} = 0.86$.

4.2 Stochastic Refinement Search as a Random Heuristic Search

It has been shown [17] that many stochastic search methods including simulated annealing, stochastic beam search and genetic algorithms are instances of a general framework called random heuristic search. The basic conditions are that the search space (Ω) be finite and that the search transition rules (τ) be Markovian and expressible as the result of s independent identically distributed random choices. The finiteness condition is not a serious limitation in most cases, including ILP, since in practice it is common to use a depth bound which leads to a finite search space. In this section we show that a stochastic refinement search is a special case of random heuristic search. First we define a random heuristic search. The following definition is adapted from [17].

Definition 7 (Random heuristic search). *Let $\Omega = \{x_0, x_1, \ldots, x_{n-1}\}$ be a search space and P_0 be a sample from Ω with size s. A random heuristic search involves a sequence $P_0 \to P_1 \to P_2 \to \ldots$ where each sample P_{i+1} is generated from previous sample P_i. Each sample P_i can be represented by a probability vector $p_i = \langle t_0, t_1, \ldots, t_{n-1}\rangle$ such that t_j is the proportion of x_j in P_i. Hence, a random heuristic search $P_0 \to P_1 \to P_2 \to \ldots$ can be also denoted as a sequence of corresponding probability distributions $p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} p_2 \xrightarrow{\tau} \ldots$ where p_i is the probability vector for P_i and this sequence is generated by iterating a transition rule $\tau : \Delta^n \to \Delta^n$ where $\Delta^n = \{\langle s_0, s_1, \ldots, s_{n-1}\rangle \mid \sum_{j=0}^{n-1} s_j = 1,\ s_j \geq 0$ for all $s_j\}$.*

Example 6. Suppose in Definition 7 we have $\Omega = \{0, 1, 2, 3, 4, 5\}$ and $P_0 = \{1, 0, 3, 1, 1, 3, 2, 2, 4, 0\}$. Then, P_0 can be represented by the probability vector $p_0 = \langle 0.2, 0.3, 0.2, 0.2, 0.1, 0.0\rangle$. Here the proportional representation given by p_0 determines the sample P_0 once the sample size is known.

Note that in Definition 7, Δ^n serves as both the space of samples of Ω and the space of probability distributions over Ω.

From Definitions 6 and 7 it is clear that a stochastic refinement search is an instance of random heuristic search. Figure 6 shows stochastic refinement search versus random heuristic search. As shown in this figure, a stochastic refinement operator σ operates on the elements of a sample S_i while a transition rule τ works directly on the corresponding sample distribution p_i. A main difference between a stochastic refinement search and a random heuristic search is that in general, a random heuristic search does not consider any ordering over Ω or for the transition rule τ. However, a stochastic refinement search is a directed search due to the generality order defined by the refinement operators. In the following we define a monotonic random heuristic search and show that an upward (or downward) stochastic refinement search can be viewed as a monotonic random heuristic search.

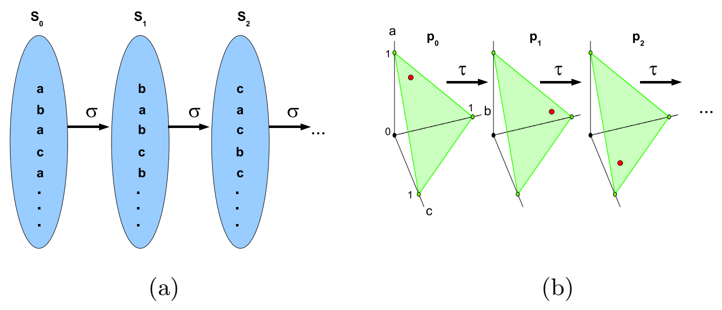


Fig. 6. (a) Stochastic refinement search (b) Random heuristic search. A stochastic refinement operator σ, operates on the elements of a sample S_i while a transition rule τ works directly on the corresponding probability vector p_i.

Definition 8 (Monotonic random heuristic search). *Let the search space Ω and the random heuristic search $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots, \ldots$ be defined as in Definition 7 and $\leq$ be a binary relation on Ω such that $\langle \Omega, \leq \rangle$ is a quasi-ordered set. If for each i, $x \in P_i$ is replaced by $x' \in P_{i+1}$ and we have $x \leq x'$ then the random heuristic search $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots$ is said to be monotonic with respect to $\leq$.*

The following Proposition follows directly from Definitions 2, 6 and 8.

Proposition 1. *A stochastic refinement search is a monotonic random heuristic search.*

Proof. According to Definitions 2 and 6, for each i, $C_i \in S_i$ and $C_{i+1} \in S_{i+1}$ we have $C_i \succeq C_{i+1}$ and therefore according to Definition 8 a stochastic refinement search is a monotonic random heuristic search with respect to $\succeq$.

The advantage of studying stochastic refinement search as a random heuristic search is that we can use the theoretical results from random heuristic search in order to analyse the behaviour and convergence of the search. In Definition 7, each state P_{i+1} of the search only depends on the previous state P_i. Hence, it can be shown that a random heuristic search can be described as a Markov chain. This property has been used [17] to estimate the probability that a particular population is generated in the next iteration. An analysis of the convergence of random heuristic search, which can also be applied to stochastic refinement search, has been discussed in [17].

4.3 Examples of Stochastic Refinement Search

In this section we discuss examples of ILP systems which use some form of stochastic refinement. In each case we characterise some aspects of the stochastic refinement with respect to the results presented in this paper.

Golem - Golem [4] is a bottom-up ILP system which employs determinate least general generalisation under relative subsumption (RLGG). Golem combines random sampling and a hill-climbing search to construct RLGGs with new randomly sampled positive examples at each iteration. The refinement in Golem can be viewed as an upward binary stochastic refinement. As shown in Example 5, with a sample size $s = 2k$ where k is the upper bound for number of clauses in the target theory, the probability that there is a pair of clauses C_1 and C_2, such that $RLGG(C_1, C_2)$ is consistent, is at least $1 - e^{-2} = 0.86$. The $RLGG$ is constructed with new sampled examples at each iteration until it converges to a consistent clause which covers a partition of positive examples.

SFoil - SFoil [11] is a top-down stochastic ILP system which combines Foil with a stochastic search in the form of simulated annealing. The stochastic search is used to choose the next literal to be added to the body of the clause. Hence, the refinement in SFoil can be viewed as downward unary stochastic refinement. The behaviour of the search can be analysed using the framework of random heuristic search. The following analysis is adopted from [17]. The sample size for simulated annealing is $s = 1$ and given a population (i.e. probability vector p) and an objective function f, the next population (i.e. probability vector q) is obtained by the following stochastic procedure: 1) sample q from a neighborhood $N(p)$ of p 2) if $f(q) < f(p)$ then the next generation is q, otherwise the next generation is q with probability $e^{(f(p)-f(q))/T_t}$ where T_t is the temperature at generation t. Then the heuristic function h which determines the stochastic transition between two distinct states i, j is defined as follows: $h(t,j)_i = \frac{[i \in N(j)]}{|N(j)|}([f(i) < f(j)] + [f(i) \geq f(j)]e^{(f(j)-f(i))/T_t})$ where $[expr]$ returns 1 if $expr$ is true and 0 otherwise.

GA-Progol - GA-Progol [15] is a stochastic ILP system in which the A^*-like search of Progol is replaced by a genetic algorithm. The stochastic refinement in GA-Progol includes both unary stochastic refinement (mutation) and binary stochastic refinement (crossover). GA-Progol also uses stochastic lgg and mgs operators relative to a bottom clause ($lgg_\perp$ and $mgs_\perp$). As for simulated annealing, a simple genetic algorithm can be characterised [17] within the framework of random heuristic search.

Aleph - Aleph [14] implements several randomised search methods including randomised local searches such as GSAT, Walksat, simulated annealing and also a randomised rapid restart search [18]. Randomised local searches in Aleph can be characterised as stochastic refinement search. Note that in some cases the stochastic refinement operators in Aleph are bi-directional and therefore the corresponding stochastic searches are non-monotonic.

QG - Quick Generalisation (QG) algorithm [8] constructs maximally general consistent clauses by stochastically pruning Progol bottom clauses. A sampling algorithm based on the QG algorithm returns the clause with highest positive compression from a sample of s calls to QG. The algorithm can be made arbitrarily efficient by choice of sample size (down to 1). The QG sampling algorithm can be viewed as a stochastic refinement search with unary refinement operator which randomly samples from “fringe” clauses (i.e. maximally general consistent clauses in the hypothesis space). A minimum sample size for a QG sampling algorithm can be estimated based on the percentage of consistent clauses which have positive compression.

ProGolem - ProGolem [6] implements an efficient (and non-determinate) variant of Golem's RLGG for the subsumption relative to $\bot$. ProGolem combines bottom-clause construction in Progol with a Golem control strategy which uses Asymmetric Relative Minimal Generalisation (ARMG) in place of determinate RLGG. ARMG in ProGolem can be viewed as stochastic binary refinement operator relative to $\bot$. Stochastic refinement relative to a bottom clause is discussed in Section 5. ProGolem combines random sampling and a beam search to construct ARMGs with new examples at each iteration.

5 Stochastic Refinement Relative to a Bottom Clause

In this section we study a special case of stochastic refinement search where a refinement space bounded by a most specific (bottom) clause is explored. This refinement space defines the search space of the ILP systems Progol and Aleph and different variants of these systems including stochastic variants described in the previous section (e.g. *QG*, ProGolem). The lattice structure and refinement operators for the subsumption order relative to $\bot$ were studied in [16]. It was shown that the refinement space of a Progol-like ILP system can be characterised using the subsumption order relative to $\bot$. It was also shown [16,6] that, unlike for the general subsumption order, efficient lgg-like operators can be implemented for the subsumption relative to a bottom clause (e.g. $lgg_\bot$ and $armg_\bot$). The following definitions are a summary of the subsumption order relative to $\bot$.

Definition 9 **($\overrightarrow{\mathcal{L}}_\bot$).** *Let $\overrightarrow{\bot}$ be the bottom clause and $\overrightarrow{C}$ a definite ordered clause as defined in [16]. $\overrightarrow{\top}$ is $\overrightarrow{\bot}$ with all variables replaced with new and distinct variables. $\theta_\top$ is a variable substitution such that $\overrightarrow{\top}\theta_\top = \overrightarrow{\bot}$. $\overrightarrow{C}$ is in $\overrightarrow{\mathcal{L}}_\bot$ if $\overrightarrow{C}\theta_\top$ is a subsequence of $\overrightarrow{\bot}$.*

Definition 10 (Subsumption relative to $\bot$). *Let $\overrightarrow{\bot}$, $\theta_\top$ and $\overrightarrow{\mathcal{L}}_\bot$ be as defined in Definition 9 and $\overrightarrow{C}$ and $\overrightarrow{D}$ be ordered clauses in $\overrightarrow{\mathcal{L}}_\bot$. We say $\overrightarrow{C}$ subsumes $\overrightarrow{D}$ relative to $\bot$, denoted by $\overrightarrow{C} \succeq_\bot \overrightarrow{D}$, if $\overrightarrow{C}\theta_\top$ is a subsequence of $\overrightarrow{D}\,\theta_\top$.*

5.1 Divisibility of the Subsumption Lattice Relative to $\bot$

In this section we review the concept of a divisible lattice as described in [5] and we show that the subsumption lattice relative to $\bot$ is a divisible lattice. The following definitions are adapted from [5] for refinement relative to $\bot$.

Definition 11. *Let $\langle G, \succeq \rangle$ be a lattice, $\bot$ be the bottom element and ρ be an upward refinement operator for this lattice. Then the depth of a clause in a unary* *refinement graph, denoted by $d_{\perp}^{(1)}$, is defined as follows: $d_{\perp}^{(1)}(C) = min_n[\exists D \sim C, D \in \rho^n(\perp)]$. Similarly, the depth of a clause in a binary refinement graph, denoted by $d_{\perp}^{(2)}$, is defined as follows: $d_{\perp}^{(2)}(C) = min_n[\exists D \sim C, D \in \rho^n(\perp, \perp)]$.*

Definition 12. *Lattice $\langle G, \succeq \rangle$ is said to be divisible with respect to ρ if for all $E \in G$ there are $C, D \in G$ such that $E \sim lgg(C, D)$ and $d_{\perp}^{(1)}(E) = d_{\perp}^{(1)}(C) + d_{\perp}^{(1)}(D)$ and $|d_{\perp}^{(1)}(C) - d_{\perp}^{(1)}(D)| \leq 1$.*

Theorem 4. *Let lattice $\langle G, \succeq \rangle$ be divisible with respect to ρ. Then for all $C \in G \backslash \{\perp\}$ we have $d_{\perp}^{(2)}(C) \leq \lceil log_2(d_{\perp}^{(1)}(C)) \rceil + 1$.*

Proof. The proof is similar to the proof of this theorem for general downward binary refinement operators (Theorem 16 in [5]). □

Theorem 5. *Lattice $\langle \overrightarrow{\mathcal{L}}_{\perp}, \succeq_{\perp} \rangle$ is divisible with respect to $\rho_{\perp}$ where $\rho_{\perp}$ is an upward refinement operator for this lattice.*

Sketch proof. We need to show that the conditions in Definition 12 hold for refinement operator $\rho_{\perp}$. The proof follows from previous results on refinement operators relative to $\perp$ that there are finite chains of upward covers from $\overrightarrow{\perp}$ to $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overrightarrow{E}$ and from $\overrightarrow{E}$ to $\overrightarrow{C}$, $\overrightarrow{D}$ (Lemma 6 and Lemma 7 in [16]) and that the refinement steps from $\overrightarrow{\perp}$ to $\overrightarrow{E} = lgg_{\perp}(\overrightarrow{C}, \overrightarrow{D})$ is defined from the refinement steps from $\overrightarrow{\perp}$ to $\overrightarrow{C}$ and $\overrightarrow{D}$ (Proposition 5 in [16]). □

From Theorem 5 and Theorem 4 it follows that the minimum length of σ-chains for upward stochastic binary refinement operator relative to $\perp$ is logarithmically related to the minimum length of σ-chains for upward stochastic unary refinement operator relative to $\perp$.

Proposition 2. *Let $S_0 \xrightarrow{\sigma_1} S_1 \xrightarrow{\sigma_1} S_2 \xrightarrow{\sigma_1} \ldots$ and $S_0 \xrightarrow{\sigma_2} S_1' \xrightarrow{\sigma_2} S_2' \xrightarrow{\sigma_2} \ldots$ be stochastic refinement searches such that σ_1 is an upward unary and σ_2 an upward binary stochastic refinement operators relative to $\perp$. Then the minimum length of σ_2-chains from S_0 to a target clause is logarithmically related to the minimum length of σ_1-chains from S_0 to the target clause.*

Proposition 2 suggests that a stochastic refinement search which uses a binary refinement operator relative to $\perp$ (e.g. $lgg_{\perp}$, $armg_{\perp}$) requires logarithmically less refinement steps to find a target clause compared to a stochastic refinement search which uses a unary refinement operator. In other words, a binary stochastic refinement has a logarithmic convergence relative to a unary stochastic refinement.

The ILP system ProGolem [6] uses Asymmetric Relative Minimal Generalisation (ARMG or $armg_{\perp}$), which as mentioned in Section 4.3, can be viewed as stochastic binary refinement operator relative to $\perp$. ARMGs are constructed by adding randomly sampled positive examples at each iteration. According to Theorem 3, when using lgg with a proper sample size one could guarantee that with a high probability a consistent clause is generated. This theorem is also true for $ARMG$ and it can be shown that with a high probability a consistent $ARMG$ can be generated. According to Proposition 1, we have a monotonic increase in the positive coverage and Proposition 2 suggest that the stochastic binary refinement search used in ProGolem should have a logarithmic convergence. The logarithmic convergence described in Proposition 2 is consistent with quick convergence of ProGolem on different datasets (e.g. Fig. 5.b in [6]).

6 Conclusions

The refinement graph theory has been viewed as the main theoretical foundation of ILP [9]. Since the publication of this theory, there have been attempts to build ILP systems based on stochastic and randomised methods. However, to date there are very little theory to support the developments of these systems. We believe this paper is a first step in this direction. In this paper we discussed how the refinement theory and relevant concepts such as refinement operators can be adapted for a stochastic ILP search. Stochastic refinement is introduced as a probability distribution over a set of clauses which can be viewed as a prior in a stochastic ILP search. We gave an analysis of stochastic refinement operators within the framework of stochastic refinement search. We studied the properties of a stochastic refinement search as two well known Markovian approaches: 1) a Gibbs sampling algorithm and 2) a random heuristic search. We have shown that a minimum sample size can be set so that in each iteration a consistent clause is generated with a high probability. A stochastic refinement search can be viewed as a monotonic random heuristic search. The advantage of studying stochastic refinement search as a random heuristic search is that we can use the theoretical results from random heuristic search in order to analyse the behaviour and convergence of the search. This paper also provided some theoretical insights to explain the relative advantages of using stochastic lgg-like operators as in the ILP systems Golem and ProGolem. The proposed framework in this paper can be extended in several ways. For example, it is possible to define stochastic refinement operators and stochastic refinement search for theories. This is especially important as the search space of theories are more complex and more difficult to search and randomised and stochastic methods should work better. As another future work, we intend to explore methods for learning probabilities in stochastic refinement operators. This could be similar to learning probabilities for a Stochastic Logic Program (SLP) [7], especially that stochastic refinement operators can be easily implemented as SLPs.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and A. Srinivasan for helpful discussions. The first author was supported by the BBSRC grant BB/C519670/1. The second author thanks the Royal Academy of Engineering and Microsoft for funding his present 5 year Research Chair.

References

1. Gelman, A., Carlin, J., Stern, H., Rubin, D.: Bayesian Data Analysis. Chapman and Hall/CRC, Boca Raton (2003)
2. Haussler, D., Kearns, M., Shapire, R.: Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. Machine Learning 14(1), 83–113 (1994)
3. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
4. Muggleton, S., Feng, C.: Efficient induction of logic programs. In: Muggleton, S. (ed.) Inductive Logic Programming, pp. 281–298. Academic Press, London (1992)
5. Muggleton, S., Marginean, F.: Binary refinement. In: Proceedings of Workshop on Logic-Based Artificial Intelligence (1999)
6. Muggleton, S., Santos, J., Tamaddoni-Nezhad, A.: ProGolem: a system based on relative minimal generalisation. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 131–148. Springer, Heidelberg (2010)
7. Muggleton, S.H.: Stochastic logic programs. In: De Raedt, L. (ed.) Advances in Inductive Logic Programming, pp. 254–264. IOS Press, Amsterdam (1996)
8. Muggleton, S.H., Tamaddoni-Nezhad, A.: QG/GA: A stochastic search for Progol. Machine Learning 70(2-3), 123–133 (2007)
9. Nienhuys-Cheng, S.-H., De Wolf, R.: Foundations of Inductive Logic Programming. LNCS (LNAI), vol. 1228. Springer, Heidelberg (1997)
10. Paes, A., Zelezny, F., Zaverucha, G., Page, D., Srinivasan, A.: ILP Through Propositionalization and Stochastic k-Term DNF Learning. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 379–393. Springer, Heidelberg (2007)
11. Pompe, U., Kovacic, M., Kononenko, I.: SFOIL: Stochastic approach to inductive logic programming. In: Proceedings of the Second Electrotechnical and Computer Science Conference ERK, vol. 93, pp. 189–192. Citeseer (1993)
12. Ruckert, U., Kramer, S.: Stochastic Local Search in k-term DNF Learning. In: Proc. 20th International Conf. on Machine Learning, pp. 648–655 (2003)
13. Serrurier, M., Prade, H., Richard, G.: A simulated annealing framework for ILP. In: Camacho, R., King, R., Srinivasan, A. (eds.) ILP 2004. LNCS (LNAI), vol. 3194, pp. 288–304. Springer, Heidelberg (2004)
14. Srinivasan, A.: The Aleph Manual. University of Oxford, Oxford (2007)
15. Tamaddoni-Nezhad, A., Muggleton, S.H.: A genetic algorithms approach to ILP. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 285–300. Springer, Heidelberg (2003)
16. Tamaddoni-Nezhad, A., Muggleton, S.H.: The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause. Machine Learning 76(1), 37–72 (2009)
17. Vose, M.D.: Random heuristic search. Theoretical Computer Science 229(1), 103–142 (1999)
18. Zelezny, F., Srinivasan, A., Page, D.: Randomised restarted search in ILP. Machine Learning, 64 1(3), 183–208 (2006)