Carnegie Mellon University

From the SelectedWorks of Ole J Mengshoel

May, 2008

Understanding the Role of Noise in Stochastic Local Search: Analysis and Experiments

Ole J Mengshoel, *Carnegie Mellon University*

Available at: https://works.bepress.com/ole_mengshoel/2/

Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 172 (2008) 955–990
Artificial Intelligence
www.elsevier.com/locate/artint

Understanding the role of noise in stochastic local search: Analysis and experiments

Ole J. Mengshoel

RIACS, NASA Ames Research Center, Mail Stop 269-3, Moffett Field, CA 94035, USA

Received 3 November 2006; received in revised form 17 September 2007; accepted 17 September 2007

Available online 1 February 2008

Abstract

Stochastic local search (SLS) algorithms have recently been proven to be among the best approaches to solving computationally hard problems. SLS algorithms typically have a number of parameters, optimized empirically, that characterize and determine their performance. In this article, we focus on the noise parameter. The theoretical foundation of SLS, including an understanding of how to the optimal noise varies with problem difficulty, is lagging compared to the strong empirical results obtained using these algorithms. A purely empirical approach to understanding and optimizing SLS noise, as problem instances vary, can be very computationally intensive. To complement existing experimental results, we formulate and analyze several Markov chain models of SLS in this article. In particular, we compute expected hitting times and show that they are rational functions for individual problem instances as well as their mixtures. Expected hitting time curves are analytical counterparts to noise response curves reported in the experimental literature. Hitting time analysis using polynomials and convex functions is also discussed. In addition, we present examples and experimental results illustrating the impact of varying noise probability on SLS run time. In experiments, where most probable explanations in Bayesian networks are computed, we use synthetic problem instances as well as problem instances from applications. We believe that our results provide an improved theoretical understanding of the role of noise in stochastic local search, thereby providing a foundation for further progress in this area.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Stochastic local search; Noise; Markov chain models; Expected hitting times; Rational functions; Noise response curves; Probabilistic reasoning; Bayesian networks; Most probable explanation; Systematic experiments; Polynomial approximation; Convexity

1. Introduction

The stochastic local search (SLS) approach has proven to be highly competitive for solving a range of hard computational problems including satisfiability of propositional logic formulas [11,18,45,46] as well as computing the most probable explanation [22,27,30] and the maximum a posteriori hypothesis [36,37] in Bayesian networks. While the details of different SLS algorithms vary [18], by definition they all use stochasticity or noise. In this article we focus on noise during local search rather than, say, noisy initialization.

Empirically, it turns out that noise has a dramatic impact on the run time of SLS algorithms [14,17,26,44,45]. Intuitively, there is a fundamental trade-off between using high and low levels of noise in SLS. Let $0 \leqslant p \leqslant 1$ represent the probability of taking a noise step. The argument for using *low* noise p is that this enables an SLS algorithm to greedily

E-mail address: omengshoel@riacs.edu.

0004-3702/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2007.09.010

climb hills without taking unnecessary downhill noise steps. The argument for using *high* noise p is that this provides the SLS algorithm with a powerful mechanism to escape local (but non-global) optima [17]. Empirically—and depending on the problem instance, the SLS algorithm, and its input parameters including noise level—an approximately optimal noise level $\hat{p}^*$ can be found. The difficulty of approximating the optimal noise p^* [26] has led to the development of adaptive noise, in which p is not static but varies as the SLS algorithm runs [7,14,26]. However, noise adaptation is still an active area of research and we believe the present work provides several key insights that should benefit further progress.

Our main contributions in this article are as follows. Based on the seminal WALKSAT architecture [44,45], we introduce a simple but general SLS algorithm called SIMPLESLS. SIMPLESLS performs noisy steps with probability p, and greedy steps with probability $1 - p$. We show that expected hitting times for SIMPLESLS are rational functions $P(p)/Q(p)$, where $P(p)$ and $Q(p)$ are polynomials. This explicitly provides a functional form corresponding to noise response curves previously only established empirically in the SLS literature [7,8,14,16,26]. We also consider the use of polynomials and convex functions. Convexity is important because local optimality implies global optimality, a dramatic simplification compared to unrestricted optimization. Rational functions as well as their special case polynomials can be used to analytically determine optimal noise levels; the latter are used in experiments in this article.

Using Markov chain analysis, and in particular by analyzing expected hitting times for trap Markov chains, we clearly show the impact of different settings of the noise parameter p when the difficulty of the problem instance varies, a key concern in stochastic local search. Trap Markov chains are an idealized model for how SLS gets trapped by local (but non-global) optima in a search space, and how noise p impacts the capability of SLS to escape such traps. Further, we show that optimal noise probability p^* varies dramatically depending on the problem instance. In particular, the optimal noise parameter varies from $p^* = 0$ for easy problem instances to p^* close to 1 for hard problem instances.

We back up our analysis with empirical results using Bayesian networks (BNs), both synthetic and from applications. BNs play a central role in a wide range of uncertainty reasoning applications including diagnosis, probabilistic risk analysis, language understanding, intelligent data analysis, error correction coding, and biological analysis. Many interesting computational BN problems, including MPE computation, are NP-complete or harder [37,40,47], hence heuristic methods such as SLS are of interest. In this work we study the problem of computing the most probable explanation (MPE) in Bayesian networks. We use an SLS algorithm known as stochastic greedy search (SGS) to search for MPEs in BNs. SGS can simulate SIMPLESLS and is a flexible, operator-based SLS approach in which different initialization and search operators can easily be investigated [27,30].

Our approach to generating synthetic BNs includes satisfiability (SAT) as a special case (see [40,47] for the reduction). Let V be the number of variables in propositional logic or the number of root nodes in BNs. Further, let C be the number of clauses in propositional logic or the number of non-root nodes in BNs. For $V > 0$, the ratio C/V is well-defined and has turned out to be useful in predicting inference difficulty for randomly generated problem instances [33,34]. An easy–hard–easy pattern in inference difficulty as a function of the C/V-ratio has been observed for SAT [34]. For BNs, an easy–hard–harder pattern has been established when inference difficulty is measured in terms of upper bounds on minimal maximal clique size (or treewidth) [29,33]. Upper bounds on optimal clique tree size and optimal maximal clique size can be computed using tree clustering [24]. In this article, we use the C/V-ratio directly to characterize the difficulty of synthetic BNs for SLS.

There is a clear relationship between our Markov chain approach and observed SLS run times. We illustrate that our idealized trap Markov chain models are relevant to experiments with real problem instances. For a few small problem instances we show complete search spaces and derive corresponding Markov chains. With these in hand, we compare (i) bounding hitting time results derived from idealized trap Markov chain models; (ii) analytical hitting time results derived from Markov chain models (which were created from real problem instances along with the behavior of SIMPLESLS); (iii) real observed SGS run times for the same problem instances; and (iv) polynomial regression results for the SGS run times A key point relating Markov chain models to classes of real problem instances is suggested by the following: Increasing problem instance hardness as controlled by C/V-ratio corresponds, roughly speaking, to increasing the size of the trap in a trap Markov chain. Consequently, mixtures of problem instances that are easy on average (small C/V and small traps) should be solved by greedier (less noisy) SLS algorithms than mixtures of problem instances that are hard on average (large C/V and large traps). In experiments with synthetic problem instances, we indeed observe these patterns as C/V is varied. To complement our experiments with synthetic problems,

we also investigate BNs from applications while also using more advanced initialization and noise algorithms. Here, we found that noise can sometimes have a rather minor impact on SLS performance while in other cases the impact can be dramatic. Generally, the empirical results support our Markov chain-based analysis.

We believe this work is significant for the following reasons. First, by using Markov chain hitting time analysis and introducing an explicit noise parameter p, this research provides an improved understanding of what role noise plays in stochastic local search. Such theoretical understanding has traditionally been limited [14,16,44], even though there exists research based on Markov chains which explores the role of traps and local maxima in SLS [15]. Second, while the experimental results of SLS are very impressive, their empirical basis means that these algorithms are quite computationally intense to optimize [26]. We believe that this work paves the way for improved approaches to optimize the noise level in stochastic local search; optimization can take place in a more principled fashion once a better understanding of the role of noise has been established. Third, we believe that Markov chain analysis and in particular expected hitting times, and more generally stochastic process theory, has been under-utilized when researching SLS algorithms. Hopefully, the impact of other parameters that describe problem instances and parameters that control the SLS search processes can be analyzed in a similar way by utilizing techniques from stochastic process theory.

The rest of this article is organized as follows. Preliminary concepts are introduced in Section 2. In Section 3 we present a simple but easy-to-analyze SLS algorithm called SIMPLESLS. Section 4 presents our three Markov chain models of SIMPLESLS, namely the exact, naive, and trap Markov chain models. Section 5 provides in-depth numerical analysis and discussion of examples of trap Markov chains and their expected hitting times. In Section 6 we present general expected hitting time and run time results. Section 7 provides empirical results using synthetic and application problem instances, specifically Bayesian networks, before we conclude in Section 8.

2. Preliminaries

We assume that the reader is familiar with fundamental definitions and results from the areas of graph theory, probability theory, and statistics; and in particular Markov chains [23] and Bayesian networks [39]. Some of the most pertinent concepts are briefly reviewed in this section.

A direct and natural way to analyze an SLS algorithm's operation on a problem instance is as a discrete time Markov chain with a discrete state space, defined as follows.

Definition 1 *(Markov chain).* A (discrete time, discrete state space) Markov chain $\{X_n,\ n = 0, 1, 2, \ldots\}$ is defined by a 3-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{P})$ where $\mathcal{S} = \{s_1, \ldots, s_k\}$ defines the set of k states while $\mathcal{V} = (\pi_1, \ldots, \pi_k)$, a k-dimensional vector, defines the initial probability distribution. The conditional state transition probabilities $\mathcal{P}$ can be characterized by means of a $k \times k$ matrix.

Only time-homogeneous Markov chains will be considered here. Many of the Markov chain models discussed below, including the trap Markov chains introduced in Section 4.3, are random walks with so-called boundary states $\{s_1, s_k\}$ and internal states $\{s_2, \ldots, s_{k-1}\}$. Further, the noise level p acts as a parameter in some of the Markov chain models we discuss in the following.

In $\mathcal{M}$, some states $\mathcal{O} \subset \mathcal{S}$ are of particular interest since they represent optimal states, and we introduce the following definition.

Definition 2 *(SLS model).* Let $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{P})$ be a Markov chain. Further, assume an objective function $f : \mathcal{S} \to \mathbb{R}$ and an optimal objective function value $f^* \in \mathbb{R}$ that defines optimal states $\mathcal{O} = \{s \mid s \in \mathcal{S} \text{ and } f(s) = f^*\}$. An SLS model is defined as a 2-tuple $(\mathcal{M}, \mathcal{O})$.

The objective function f and the optimal states $\mathcal{O}$ are independent of the SLS algorithm and its parameters. Note that neither $\mathcal{M}$ nor $\mathcal{O}$ are, in general, explicitly specified. Rather, they are induced by the objective function (or problem instance), the SLS algorithm, and the SLS algorithm's parameter settings. In fact, finding an $s^* \in \mathcal{O}$ is the purpose of computation, and it is given implicitly by means of the objective function f and the optimal objective function value $f^* \in \mathbb{R}$. Without loss of generality our main focus is on maximization problems in this article.

Consider an SLS model $(\mathcal{M}, \mathcal{O})$. A hitting time analysis of the Markov chain $\mathcal{M}$ gives the expected number of steps needed to reach $s^* \in \mathcal{O}$. Expected hitting times, to be introduced in Definition 5, are based on first passage times

and their expectations, which we introduce now. In the following definition, X_j is an arbitrary random variable among the random variables $\{X_n,\ n = 0, 1, 2, \ldots\}$ of a Markov chain.

Definition 3 *(First passage time).* Consider an SLS model $(\mathcal{M}, \mathcal{O})$ and let $s_i \in \mathcal{S}$ where $\mathcal{S}$ is $\mathcal{M}$'s states. The first passage time T into $s^* \in \mathcal{O}$, where $|\mathcal{O}| = 1$, is given by $T = \min(j \geqslant 0:\ X_j = s^*\}$. The expected value of T, given initial state $X_0 = s_i$, is defined as

$$m_i := E(T \mid X_0 = s_i).$$

In this article, m_i is sometimes a function of noise p, in which case we say $m_i(p)$. Note also that Definition 3 can easily be generalized to cover passage time into multiple optimal states, $|\mathcal{O}| > 1$, however to simplify the exposition we generally focus on the one-state case here.

We often consider problem instances represented as bitstrings of length n, $\boldsymbol{b} \in \{0, 1\}^n$, in which case $s^* = \boldsymbol{b}^* \in \{0, 1\}^n$. We are interested in maximization: finding a $\boldsymbol{b}^* \in \mathcal{O}$ such that $f(\boldsymbol{b}^*) \geqslant f(\boldsymbol{b})$ for all $\boldsymbol{b} \in \{0, 1\}^n$. The following definition formally introduces the useful concept of unitation as summing over a bitstring.

Definition 4 *(Unitation).* Let $\boldsymbol{b} = b_1 b_2 \ldots b_n \in \{0, 1\}^n$ be a bitstring of length n. The number of ones (or unitation) of $\boldsymbol{b} = b_1 \ldots b_n$ is defined as $u(\boldsymbol{b}) = \sum_{i=1}^{n} b_i$.

Counting the number of ones is, after an easy transformation of the search space, equivalent to counting the number of correct bits. The number of correct bits is the number of bits that are equal between $\boldsymbol{b}^*$ and the current state of SLS search $\boldsymbol{c}$. More formally, let $\boldsymbol{b}^*$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$,$\in \{0, 1\}^n$. In order to normalize the search space, one can use the transformations $\boldsymbol{b} := (\boldsymbol{b}^* \texttt{ xor } \boldsymbol{c})$ and $\boldsymbol{d} := \bar{\boldsymbol{b}}$ where xor denotes exclusive or and $\bar{\boldsymbol{b}}$ denotes the bit-wise inversion of $\boldsymbol{b}$. Now, take $u(\boldsymbol{d})$ in this transformed search space in order to obtain the number of correct bits; clearly $\boldsymbol{d}^* = 1 \ldots 1$. To simplify notation, we often gloss over the transformations, and say that $\boldsymbol{b}^* = 1 \ldots 1$ without loss of generality [6]. See Fig. 7 for concrete examples.

Using conditional expectations, one obtains from Definition 3 the following well-known result.

Theorem 5 *(Expected hitting time). Let $\mathcal{M}$ be a Markov chain with state space $\mathcal{S} = \{s_1, \ldots, s_k\}$ and first passage time T (into s_k). The expected hitting time h is then*

$$h := \sum_{i=1}^{k} E(T \mid X_0 = i) \Pr(X_0 = i) = \sum_{i=1}^{k} m_i \pi_i. \tag{1}$$

Expected hitting time can be used to analyze the expected time to reach an optimal state, and is therefore closely related to the observed run time for an SLS algorithm. In the context of SLS, the hitting time h is with respect to some state in $\mathcal{O}$ and depends on the algorithm's input parameters including the problem instance. Our main interest in this article is expected hitting time as a function of noise p. Typically, we therefore get $h(p)$ instead of just h (as in (1)), and reserve the short form "expected hitting time" for $h(p)$. By studying $h(p)$ and varying p along the x-axis in graphs, we obtain expected hitting time curves that are counterparts to experimental noise response curves. Clearly, one would like to find SLS parameters such that the minimal expected hitting time h^* is obtained. Often, the search for minimal expected hitting time h^* has an empirical component and the notation $\hat{h}^*$ may be used.

In the experimental part of this work we will focus on an SLS approach to computing the most probable explanation [22,27,30] in Bayesian networks (BNs). This problem is interesting in its own right and also generalizes satisfiability [40,47]. BN nodes can be continuous, however we will here restrict ourselves to discrete BN nodes.

Definition 6 *(Bayesian network).* A Bayesian network is a tuple $\beta = (\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{P})$, where $(\boldsymbol{X}, \boldsymbol{E})$ is a DAG with an associated set of conditional probability distributions $\boldsymbol{P} = \{\Pr(X_1 \mid \Pi_{X_1}), \ldots \Pr(X_n \mid \Pi_{X_n})\}$. Here, $\Pr(X_i \mid \Pi_{X_i})$ is the conditional probability distribution for $X_i \in \boldsymbol{X}$. Let π_{X_i} represent the instantiation of the parents Π_{X_i} of X_i. The independence assumptions encoded in $(\boldsymbol{X}, \boldsymbol{E})$ imply the joint probability distribution

$$\Pr(\boldsymbol{x}) = \Pr(x_1, \ldots, x_n) = \Pr(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \Pr(x_i \mid \pi_{X_i}). \tag{2}$$

A BN may be provided with *observations* or *evidence* by setting or clamping m nodes $\{O_1, \ldots, O_m\} \subset \boldsymbol{X}$ to known states $\boldsymbol{o} = \{O_1 = o_1, \ldots, O_m = o_m\} = \{o_1, \ldots, o_m\}$. These nodes are called *observation nodes* and need to be considered in explanations, defined as follows.

Definition 7 *(Explanation).* Consider a BN $\beta = (\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{P})$ with $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and observations $\boldsymbol{o} = \{o_1, \ldots, o_m\}$ for $m < n$. An explanation $\boldsymbol{x}$ assigns states to all non-evidence nodes $\{X_{m+1}, \ldots, X_n\}$: $\boldsymbol{x} = \{x_{m+1}, \ldots, x_n\} = \{X_{m+1} = x_{m+1}, \ldots, X_n = x_n\}$.

Among all explanations in a BN, the u most probable ones are of particular interest.

Definition 8 *(Most probable explanation (MPE)).* Let $\boldsymbol{x}$ range over all explanations in a BN β. Finding a most probable explanation (MPE) in β is the problem of computing an explanation $\boldsymbol{x}^*$ such that $\Pr(\boldsymbol{x}^*) \geqslant \Pr(\boldsymbol{x})$. The u most probable explanations is $\boldsymbol{X}^* = \{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_u^*\}$ where $\Pr(\boldsymbol{x}^*) = \Pr(\boldsymbol{x}_1^*) = \cdots = \Pr(\boldsymbol{x}_u^*)$ and $\Pr(\boldsymbol{x}_i^*) \geqslant \Pr(\boldsymbol{x})$ for $1 \leqslant i \leqslant u$.

As is common, we compute just one MPE $\boldsymbol{x}^* \in \boldsymbol{X}^*$ even when multiple MPEs exist in a BN. Computation of the M most probable explanations, for $M \geqslant 1$ and where explanations do not have the same probability, is a generalization that has also been investigated [50]. Following Pearl, we sometimes denote computing an MPE as belief revision, while computing the marginal distribution over a BN node is also denoted belief updating [39]. Many of the computational BN problems of interest are hard. Exact MPE computation is NP-hard [47] and the problem of relative approximation of an MPE also been shown NP-hard [1]. Belief updating is computationally hard also [4,40].

3. Stochastic local search

We discuss a simplified variant of SLS, SIMPLESLS, based on the seminal WALKSAT architecture [16,44,45]. SIMPLESLS is not intended to be competitive with state-of-the-art SLS algorithms, which often are tailored to the domain and problem instances at hand [18]. Rather, SIMPLESLS is intended to enable analysis and capture what is common to SLS algorithms based on WALKSAT, in particular with respect to their noisy search components.

The SIMPLESLS algorithm takes as input these parameters: n—bitstring length; p—noise probability; f—an objective function used to evaluate $f(\boldsymbol{b})$ where $\boldsymbol{b} = b_1 b_2 \ldots b_n \in \{0, 1\}^n$ is a bit string of length n; f^*—optimum value of f; MAX-FLIPS—the number of flips before restart; and finally MAX-TRIES—the number of restarts before termination. SIMPLESLS iteratively takes search steps, which are either greedy or noisy as further discussed below.

SIMPLESLS maintains a current estimate of $\boldsymbol{b}^*$, namely $\hat{\boldsymbol{b}}^*$, as well as the current state $\boldsymbol{b}$. To initialize $\boldsymbol{b}$, SIMPLESLS puts $b_i := 0$ with $\Pr(1/2)$ and $b_i := 1$ with $\Pr(1/2)$ for all $1 \leqslant i \leqslant n$. Such initialization uniformly at random is common in SLS algorithms [18]. SIMPLESLS initializes $\hat{\boldsymbol{b}}^*$ in the same manner. Then, one-bit flips are repeatedly made to $\boldsymbol{b}$ until success or restart, described below, takes place. A one-bit flip means that $\boldsymbol{b}$'s ith bit, where $1 \leqslant i \leqslant n$, is picked and then flipped. Suppose the current bitstring is $\boldsymbol{b} = b_1 b_2 \ldots b_i \ldots b_{n-1} b_n$. Then a flip is to set $b_i' := \bar{b}_i$ and the new bitstring $\boldsymbol{b}'$ is formed as $\boldsymbol{b}' := b_1 b_2 \ldots \bar{b}_i \ldots b_{n-1} b_n$ and then setting $\boldsymbol{b} := \boldsymbol{b}'$. If $f(\boldsymbol{b}) \geqslant f(\hat{\boldsymbol{b}}^*)$ then $\hat{\boldsymbol{b}}^* := \boldsymbol{b}$. Further, if $f(\hat{\boldsymbol{b}}^*) = f^*$ then SIMPLESLS terminates successfully.

The way in which the ith bit in $\boldsymbol{b}$ is picked depends on the search operator or algorithm used. A random variable O is now introduced, representing the random selection of which operator to apply next. To keep our analysis simple we assume exactly two local search operators or operator types, o_G (greedy) and o_N (noisy) and hence $\Pr(O = o_G) + \Pr(O = o_N) = 1$.[1] The SIMPLESLS algorithm repeatedly instantiates the random variable O by randomly picking one of the two operators o_G and o_N as follows:

Greedy operator $O = o_G$: With probability $\Pr(O = o_G) = 1 - p$, a greedy step is made from $\boldsymbol{b}$ to $\boldsymbol{b}'$, maximizing objective function increase from $f(\boldsymbol{b})$ to $f(\boldsymbol{b}')$. If there is tie between k candidate bitstrings $\{\boldsymbol{b}_1', \ldots, \boldsymbol{b}_k'\}$, the algorithm picks one of them uniformly at random. If no $f(\boldsymbol{b}') \geqslant f(\boldsymbol{b})$, it puts $\boldsymbol{b}' := \boldsymbol{b}$.

[1] In the SGS system, used for experimentation in Section 7, o_N is implemented by the NU operator and o_G is implemented by either the BM operator or the GM operator.

Noisy operator $O = o_N$: With probability $\Pr(O = o_N) = p$, the algorithm makes a noise step as follows. First, an integer $1 \leqslant i \leqslant n$ is picked uniformly at random. This ith bit is then flipped, forming $\boldsymbol{b}'$.

The SIMPLESLS algorithm iteratively flips bits until f^* is reached or MAX-FLIPS flips have been performed. Once MAX-FLIPS flips have been done, a new try is started, and the process continues until MAX-TRIES has been reached. The approach is closely related to WALKSAT [44,45], in particular its random noise variant.

In our analysis here MAX-TRIES = MAX-FLIPS = ∞ is used; we do vary MAX-FLIPS in some of our experiments. We assume a Las Vegas algorithm that is guaranteed to eventually terminate with $\hat{\boldsymbol{b}}^* = \boldsymbol{b}^*$. This latter assumption simplifies our analysis and exposition since we need only be concerned with a randomly varying run time T. One can of course also fix the run time and let the approximated output $\hat{\boldsymbol{b}}^*$ be a random variable; the analysis is then different and we shall not follow this route in this article. Our approach is closely related to previous stochastic local search (SLS) algorithms for satisfiability [11,16,18,43,45,46] and Bayesian network problems [22,27,30,36,37]. It is somewhat related to previous research on stochastic simulation and guided local search. Stochastic simulation, which can be used to compute MPEs [39], is essentially Gibbs sampling in Bayesian networks. Even though the Gibbs sampler in many respects is general, it is quite different from most SLS approaches in that it typically operates in cycles [25]. A cyclic Gibbs sampler iterates systematically over all non-evidence nodes in a BN. SLS algorithms, on the other hand, are generally more opportunistic and do not operate on such fixed schedules. Stochastic simulation has been shown to be outperformed by the SLS approach of combining greedy and noisy search [22], and we do not investigate stochastic simulation in this article. There is also another class of local search algorithms, called guided local search [20,35], which emphasizes changing the cost function rather than employing noise. Guided local search algorithms such as GLS [35] and GLS+ [20] are clearly very interesting, however our focus in this article is on stochastic local search algorithms and their analysis.

4. Markov chain models of stochastic local search

Important aspects of the behavior of many stochastic local search (SLS) algorithms can be represented by means of discrete-time Markov chains. In Section 4.1 we discuss an exact Markov chain analysis of SLS along with its pros and cons. We then go on to provide approximate models and results. In Section 4.2 a simple 3-state Markov chain model is discussed, while in Section 4.3 a more general model, trap Markov chains, is developed.

Readers who find the naive and trap Markov chain models too restrictive may want to skim Sections 4.2, 4.3, and 5, and instead consider our more general analysis in Sections 4.1 and 6 as well as experimental results in Section 7.

4.1. Exact Markov chain analysis

Clearly, key aspects of the operation of SIMPLESLS and similar SLS algorithms on a specific problem instance can be formalized as simulation of a Markov chain. The structure of the underlying exact Markov chain is that of a hypercube, where each hypercube state $\boldsymbol{b}$ represents a bitstring. A state $\boldsymbol{b} \in \{0, 1\}^n$ in such a Markov chain has n neighbors, namely those bitstrings one flip away. Stated formally, $\boldsymbol{b}'$ is a neighbor to $\boldsymbol{b}$ if $\boldsymbol{b}'$ can be obtained by flipping one of $\boldsymbol{b}$'s bits.

Definition 9 *(Neighborhood).* Let $\boldsymbol{b}$ be a bitstring of length n. $\boldsymbol{b}$'s neighborhood $n(\boldsymbol{b})$, strict neighborhood $n'(\boldsymbol{b})$, and non-neighborhood $\bar{n}(\boldsymbol{b})$ are defined as follows:

$$n(\boldsymbol{b}) = \left\{ \boldsymbol{c} \in \{0, 1\}^n \;\middle|\; \sum_{i=1}^{n} |b_i - c_i| \leqslant 1 \right\}$$
$$n'(\boldsymbol{b}) = n(\boldsymbol{b}) - \{\boldsymbol{b}\}$$
$$\bar{n}(\boldsymbol{b}) = \{0, 1\}^n - n(\boldsymbol{b}).$$

The following Markov chain model is introduced in order to reflect the performance of SIMPLESLS as stated formally in Theorem 11.

Definition 10 *(Exact Markov chain model).* The exact Markov chain model $\mathcal{M}$ of SIMPLESLS has states $\mathcal{S} = \{s_0, s_1, \ldots\} = \{\boldsymbol{b} \mid \boldsymbol{b} \in \{0,1\}^n\}$ and an initial probability distribution $\mathcal{V}$ with $\Pr(X_0 = s_i) = 1/2^n$ for $1 \leqslant i \leqslant 2^n$. The transition probability matrix $\mathcal{P}$ is a stochastic matrix given by

$$\Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j) = 0 \text{ if } \boldsymbol{b}_{j+1} \in \bar{n}(\boldsymbol{b}_j) \tag{3}$$

$$\Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j) \geqslant 0 \text{ if } \boldsymbol{b}_{j+1} \in n(\boldsymbol{b}_j). \tag{4}$$

Theorem 11. SIMPLESLS *simulates an exact Markov chain model up to MAX-FLIPS flips.*

Proof. Obviously, $\mathcal{S}$ and $\mathcal{V}$ are as stated and we now consider $\mathcal{P}$. Clearly, SIMPLESLS can be regarded as a stochastic process $\{X_i, i = 0, 1, 2, \ldots\}$ over the discrete state space $\mathcal{S} = \{\boldsymbol{b} \mid \boldsymbol{b} \in \{0,1\}^n\}$. Further, for fewer than MAX-FLIPS flips, the next state $\boldsymbol{b}_{j+1}$ is independent of the past given the current state $\boldsymbol{b}_j$, therefore

$$\Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_0 = \boldsymbol{b}_0, \ldots, X_j = \boldsymbol{b}_j) = \Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j),$$

which defines a Markov chain. For the transition probabilities $\mathcal{P}$ there are by construction two SIMPLESLS operators to consider, namely o_G and o_N. Since the number of flips is less than MAX-FLIPS we obtain

$$\Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j) = \begin{cases} \Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j, O_j = o_G) & \text{if } o_G \text{ picked} \\ \Pr(X_{j+1} = \boldsymbol{b}_{j+1} \mid X_j = \boldsymbol{b}_j, O_j = o_N) & \text{if } o_N \text{ picked.} \end{cases}$$

In both cases, conditions (3) and (4) are obeyed and the theorem follows. □

Here is an example; see Definition 4 for a formal introduction of unitation $u(\boldsymbol{b})$.

Example 12. An example hypercube representing an exact Markov chain model of a 5-bit SLS search space is shown in Fig. 1.

We now consider a state that is a local minimum. In such a case the neighboring states $n'(\boldsymbol{b})$ must have the same or higher objective function value, and since without loss of generality we have assumed that SIMPLESLS performs maximization, the following result is obtained.

Lemma 13. *If the current state $\boldsymbol{b}_j$ in* SIMPLESLS *is a local minimum, the next state $\boldsymbol{b}_{j+1}$ will always be a state in the strict neighborhood*: $\boldsymbol{b}_{j+1} \in n'(\boldsymbol{b}_j)$.

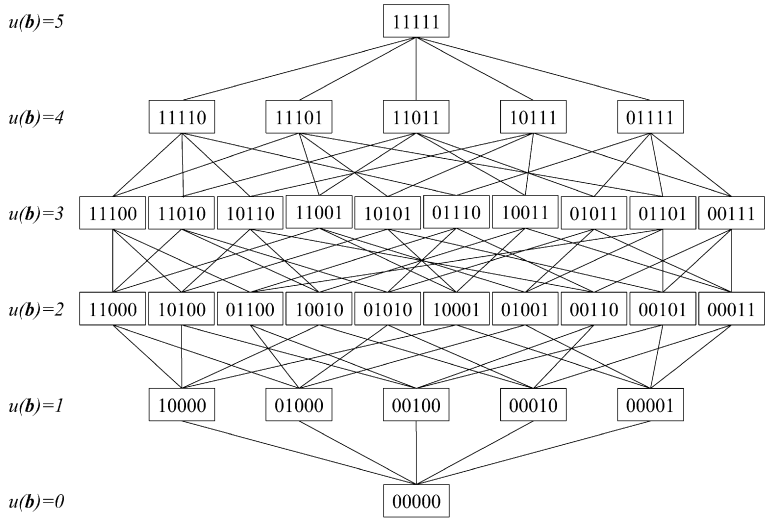

Fig. 1. The hypercube over all 5-bit bitstrings $\boldsymbol{b} \in \{0,1\}^5$. Neighboring bitstrings, bitstrings in which one bit differs, have an edge between them. The unitation $u(\boldsymbol{b})$ for all bitstrings at the same level is shown to the left.

Proof. Suppose an arbitrary candidate for next state $\boldsymbol{b}_{j+1}$ is denoted $\boldsymbol{b}'$: $\boldsymbol{b}' \in n(\boldsymbol{b}_j)$. SIMPLESLS applies one of two operators, $O = o_G$ (greedy) or $O = o_N$ (noisy). If $O = o_N$, a bit is always flipped such that $\boldsymbol{b}_{j+1} \neq \boldsymbol{b}_j$. If $O = o_G$, $\boldsymbol{b}_{j+1} := \boldsymbol{b}_j$ only if no $f(\boldsymbol{b}') \geqslant f(\boldsymbol{b}_j)$, where $\boldsymbol{b}' \in n'(\boldsymbol{b})$, exists. However, this is clearly not the case here since $\boldsymbol{b}_j$ by assumption is a local minimum and thus $f(\boldsymbol{b}') \geqslant f(\boldsymbol{b}_j)$ for all $\boldsymbol{b}' \in n'(\boldsymbol{b}_j)$. □

Since SIMPLESLS searches in a state space defined by an n-dimensional hypercube, one might be tempted to also perform all analysis in this space. However, such exactness comes at a steep price. Since the size of $\mathcal{P}$ is $|\{0,1\}^n| \times |\{0,1\}^n| = 2^{n+1}$ and the size of $\mathcal{V}$ is $|\{0,1\}^n| = 2^n$, the specification of $\mathcal{M}$ becomes impossibly large even for moderately sized problem instances. For small n, exact Markov chain analysis is feasible. For large n, exact Markov chain analysis may be infeasible.

In the following we provide Markov chain analysis results, similar to above, however they are approximate since we operate on smaller state spaces compared to the exact space traversed by SLS algorithms such as SIMPLESLS.

4.2. Naive Markov chain analysis

Here, we are interested in an abstract Markov chain model of SLS algorithms, and introduce the following simple random walk model, which constitutes a stepping stone for the trap Markov chains in Section 4.3.

Definition 14 *(Naive Markov chain model).* The naive Markov chain model $(\mathcal{M}, \mathcal{O})$ has Markov chain $\mathcal{M}$ with states $\mathcal{S} = \{s_0, s_1, s_2\} = \{0, 1, 2\}$, initial probability distribution $\mathcal{V} = (\pi_0, \pi_1, \pi_2)$, transition probability matrix

$$\mathcal{P} = \begin{bmatrix} 1 & 0 & 0 \\ 1-b-c & b & c \\ 0 & a & 1-a \end{bmatrix}, \tag{5}$$

and $\mathcal{O} = \{s_0\} = \{0\}$.

The naive Markov chain model, along with an example, is illustrated in Fig. 2. The values of a, b, and c as well as the values of the initial probabilities π_0, π_1, and π_2 all depend on the problem instance and SLS algorithm at hand. In (5), $s_0 = 0$ represents the optimum $\mathcal{O}$; $s_1 = 1$ represents the search space close to $s_0 = 0$, and $s_2 = 2$ represents the search space distant from $s_0 = 0$. Distance is measured using Hamming distance $d(\boldsymbol{b}_1, \boldsymbol{b}_2)$ between two bitstrings, $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \{0,1\}^n$. There is also a threshold $0 \leqslant z < n$. Formally, s_0 represents $\{\boldsymbol{b}^*\}$, s_1 represents $\{\boldsymbol{b} \mid \boldsymbol{b} \in \{0,1\}^n, 0 < d(\boldsymbol{b}, \boldsymbol{b}^*) \leqslant z\}$, and s_2 represents $\{\boldsymbol{b} \mid \boldsymbol{b} \in \{0,1\}^n, d(\boldsymbol{b}, \boldsymbol{b}^*) > z\}$. The state $s_2 = 2$ can be used to represent search space traps [11,15], a topic we will discuss in more detail in Section 4.3. The states $s_0 = 0$ and $s_1 = 1$ represent the part of the search space where a strong SLS initialization algorithm will reliably start search, leading to a low number of flips before reaching $\mathcal{O}$, while $s_2 = 2$ gives a high number of flips. Suppose that

$$\Pr(X_{i+1} = 1 \mid X_i = 1) = b < \Pr(X_{i+1} = 2 \mid X_i = 2) = 1 - a.$$

Here, the probability b of staying at s_1, in other words close to s_0 while not transitioning to the optimum state s_0 or further away to s_2, is smaller than the probability $1 - a$ of staying at s_2, distant from optimum. In other words, once

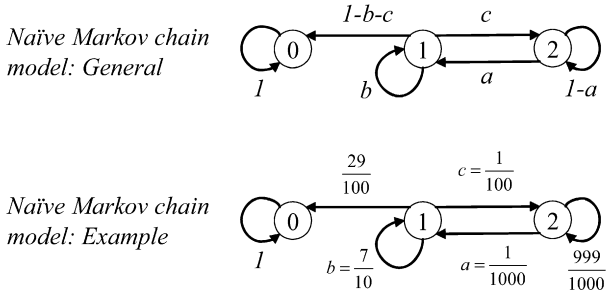

Fig. 2. A naive Markov chain model (top) with parameters a, b, and c as well as an example of parameter settings (bottom). This is a top–down model, in which each state represents multiple states in the underlying exact Markov chain model of the SLS search process.

SIMPLESLS is close to optimum it may be quite likely to find optimum, while on the other hand if SIMPLESLS is far from optimum it may be likely to stay far away. These concepts are reflected in the numerical example in Fig. 2.

Expected times for first entering the optimum state $s_0 = 0$ is given by our following result.

Lemma 15. *In the naive Markov chain model, suppose that* $a(b + c - 1) \neq 0$ *and* $b \neq 1$. *Then, the first passage times* m_0, m_1, *and* m_2 *are*

$$m_0 = 0$$
$$m_1 = -\frac{a + c}{a(b + c - 1)}$$
$$m_2 = -\frac{a - b + 1}{a(b + c - 1)}.$$

Proof. The fact that $m_0 = 0$ is obvious. A hitting time analysis of the Markov chain (5) gives these two simultaneous equations:

$$m_1 = 1 + b \times m_1 + c \times m_2$$
$$m_2 = 1 + a \times m_1 + (1 - a) \times m_2,$$

which under the assumption $a(b + c - 1) \neq 0$ has the above solutions. □

Here is an illustration of the potentially dramatic difference between SLS starting in state $s_1 = 1$ (giving first passage time m_1) versus in state $s_2 = 2$ (giving first passage time m_2).

Example 16. Let, in (5), $a = 1/1000$, $b = 7/10$, and $c = 1/100$. Using Lemma 15 we obtain

$$m_1 = -\frac{a + c}{a(b + c - 1)} = 37.93$$
$$m_2 = -\frac{a - b + 1}{a(b + c - 1)} = 1038.$$

Suppose now that we are able to increase the probability of "good" jumps from state $s_2 = 2$ (distant from optimum) to state $s_1 = 1$ (close to optimum) to $a = 1/100$, while only increasing the probability of "bad" jumps from state $s_1 = 1$ to state $s_2 = 2$ to $c = 1/50$. These changes to a and c may for example be implemented by increasing the SIMPLESLS noise p, giving the following result.

Example 17. Let, in (5), $a = 1/100$, $b = 7/10$, and $c = 1/50$. Using Lemma 15 again gives

$$m_1 = -\frac{a + c}{a(b + c - 1)} = 10.71$$
$$m_2 = -\frac{a - b + 1}{a(b + c - 1)} = 110.7.$$

In this case, we see improvements for both m_1 and m_2 compared to Example 16; relatively speaking the almost 10-fold reduction in passage time m_2 is most significant.

Benefits of the naive Markov chain model include its simplicity and ease of specification. This again leads to improved understanding of SLS through the calculation of passage times, extending previous research [15]. However, this naive model is perhaps too simple to capture several important facets of SLS algorithms, in particular there is no direct representation of noise probability p. We now turn to more realistic Markov chain models, trap Markov chain models, in which p is explicitly represented and there is an attempt to strike a balance between the state space sizes of exact and naive Markov chain models.

4.3. Trap Markov chain analysis

In the following, we continue to use Markov chains but introduce a few extensions, namely a noise parameter p, a variable sub-string length $\ell \leqslant n$, and an approach to vary problem difficulty at a higher abstraction level than for the naive Markov chain model. To make the size of $\mathcal{M}$ tractable even for large problem instances, we focus on objective functions f in which it is reasonable to count the number of correct bits in the current bitstring $\boldsymbol{b}$ compared to the optimum $\boldsymbol{b}^*$.

We introduce an approach and a class of Markov chains inspired by deceptive trap functions [6]. Trap functions are related to search space traps, which are portions of the search space that are attractive to SLS but do not contain optima [11,15]. In deceptive trap functions, only the number of 1's (or the unitation) and not their positions determine objective function values. In deceptive trap functions over bitstrings of length $\ell = n$, there is a local deceptive optimum at 0 representing $\boldsymbol{b} = 0 \ldots 0$, a global optimum at ℓ representing $\boldsymbol{b}^* = 1 \ldots 1$, and a slope-change location at z.

Definition 18 *(Trap function).* A trap function $g(x; \ell, z)$ (abbreviated $g(x)$) of length ℓ with slope-change location $z \in \{0, \ldots, \ell - 1\}$ is a function with domain $\{0, \ldots, \ell\} \subset \mathbb{N}$ where $g(x) > g(x+1)$ for $0 \leqslant x \leqslant z-1$ and $g(x) < g(x+1)$ for $z \leqslant x \leqslant \ell - 1$. There is a unique global optimum $g^* = g(\ell)$ and $g(z+1) > g(z-1)$ for $z > 0$.

Intuitively, a trap function is such that for $x < z$, an SLS algorithm such as SIMPLESLS will greedily (using o_G) move towards the *local optimum* $x = 0$, representing trapping in the part of search space dominated by local maxima. For $x \geqslant z$, SIMPLESLS will greedily (again using o_G) move towards the *global optimum* at $x = \ell$, representing search in the part of the search space dominated by the global optimum $\boldsymbol{b}^*$. The size of the domain as well as the placement of the slope-change parameter z determine the difficulty of the trap function for SLS. Here is an example trap function related to SAT. Note that this example is a simplification compared to actual SAT problem instances, which we return to in Section 7.

Example 19 *(Easy SAT instance).* Consider a conjunctive normal form (CNF) formula with $V = 5$ variables and $C = 20$ clauses. Further, assume that the formula has exactly one satisfying assignment and that the number of satisfied clauses g varies from 15 to 20 as follows: $g(0) = 16$, $g(1) = 15$, $g(2) = 17$, $g(3) = 18$, $g(4) = 19$, and $g(5) = 20$. This is a trap function with slope-change location $z = 1$, since $g(0) > g(1)$, $g(5) > g(4) > g(3) > g(2) > g(1)$, $g(2) > g(0)$, and $g^* = g(5)$.

Example 19 is illustrated in Fig. 3 along with other examples of varying difficulty. A key point here is that moving the slope-change location z from the left to the right corresponds to increasing problem instance hardness. Intuitively, easy problem instances (with slope-change location close to $\boldsymbol{b} = 0 \ldots 0$) should be solved by a greedier algorithm than a hard problem instance (with slope-change location close to $\boldsymbol{b} = 11 \ldots 1$). Two complete search spaces are shown in Fig. 4.

We assume that SLS searches over bitstrings rather than integers (as used in Definition 18), hence we introduce the following definition.

Definition 20 *(Binary trap function).* Let $\boldsymbol{b} \in \{0, 1\}^\ell$ be a bitstring and let $g(x; \ell, z)$ be a trap function as introduced in Definition 18. Then $f(\boldsymbol{b}; \ell, z) := g(u(\boldsymbol{b}); \ell, z)$, abbreviated $f(\boldsymbol{b})$, defines a binary trap function with parameters ℓ and z.

The concept of trap functions defined over bitstrings might seem somewhat abstract, but it can be used to represent critical aspects of how SLS algorithms get trapped when applied to NP-hard problems such as SAT and MPE. For SAT, a bitstring $\boldsymbol{b} \in \{0, 1\}^\ell$ represents a truth assignment to all $\ell = n$ variables in a CNF formula, for MPE it represents an explanation over $\ell = n$ binary nodes.

Definition 21 *(Trap).* Let $\boldsymbol{c} \in \{0, 1\}^\ell$ be a bitstring and let $\boldsymbol{b}^* \in \{0, 1\}^\ell$ be an optimal bitstring. A greedy neighbor $\boldsymbol{g} \in n(\boldsymbol{c})$ is a neighbor that is reachable by a greedy SIMPLESLS step. Then $\boldsymbol{c}$ is a trap state (bitstring) if all greedy steps from $\boldsymbol{c}$ increase the distance to all optima: $d(\boldsymbol{g}, \boldsymbol{b}^*) > d(\boldsymbol{c}, \boldsymbol{b}^*)$ for all $\boldsymbol{g}$ and $\boldsymbol{b}^*$. The search space's trap is defined as $\mathcal{T} = \{\boldsymbol{c} \text{ is trap state} \mid \boldsymbol{c} \in \{0, 1\}^\ell\}$.

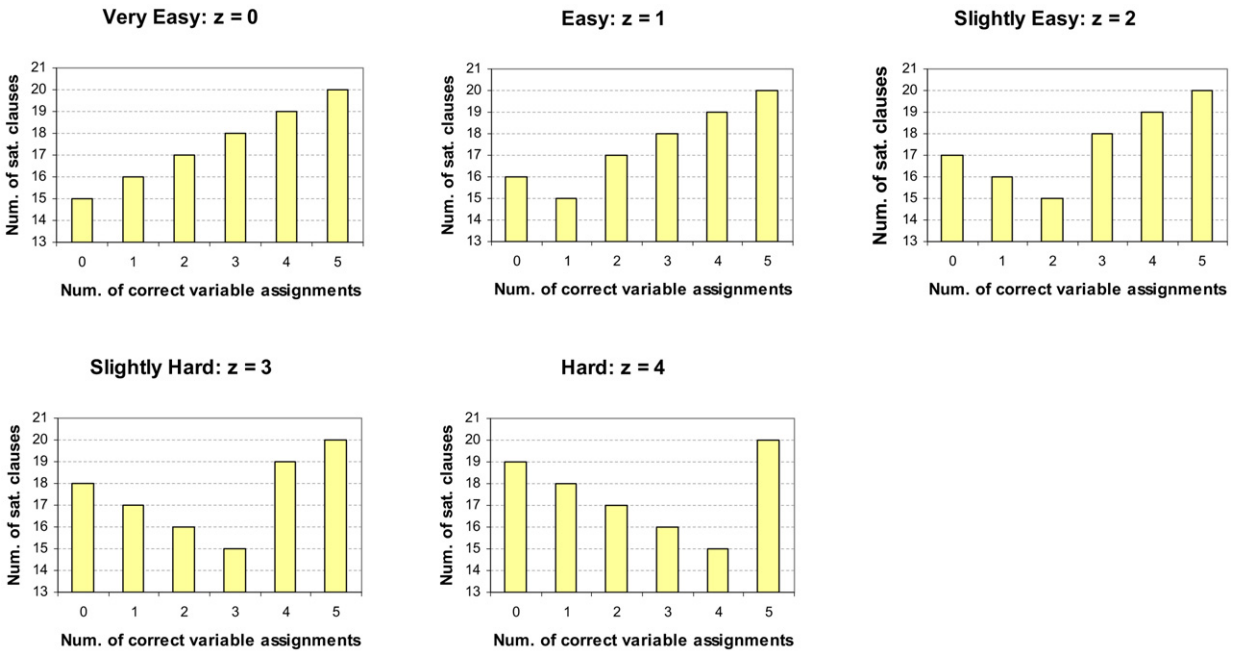

Fig. 3. Trap functions $g(x;\ell,z) = g(x;5,z)$ based on the satisfiability problem (3SAT) with $V = 5$ variables and $C = 20$ clauses, assuming one satisfying assignment. We consider five hypothetical problem instances, ranging from very easy (with slope-change location $z = 0$) to hard (where $z = 4$). For an assignment to all $V = 5$ variables, the number of correct assignments ranges from $x = 0$ to $x = 5$ as shown on the x-axis. The number of satisfied clauses is shown on the y-axis. The complete search spaces for Very Easy and Hard are shown in Fig. 4.

Our concept of trap is related to search space traps [11,15] as well as search space reachability analysis, where states from which a solution is not reachable are determined [51].

It is easy to show that the trap size of a binary trap function is as follows.

Lemma 22. *Let $f(\boldsymbol{b};\ell,z)$ be a binary trap function. The trap size $|\mathcal{T}|$ of f is given by*

$$|\mathcal{T}| = \begin{cases} 0 & \text{if } z = o \\ \sum_{i=0}^{z-1} \binom{\ell}{i} & \text{if } z \geqslant 1. \end{cases} \tag{6}$$

The following result holds in general, where $\mathcal{T}_{\max}$ is the largest trap possible over $\{0,1\}^n$ and $\mathcal{T}_{\min}$ is the smallest trap possible.

Lemma 23. *The maximal trap size $|\mathcal{T}_{\max}|$ over $\{0,1\}^n$ is given by $|\mathcal{T}_{\max}| = 2^n - n - 1$; the minimal trap size is $|\mathcal{T}_{\min}| = 0$.*

Proof. In the worst case there is only one optimum $\boldsymbol{b}^* \in \mathcal{O}$. Clearly, $n(\boldsymbol{b}^*)$ can not be part of the trap, $n(\boldsymbol{b}^*) \notin \mathcal{T}_{\max}$, and since $|n(\boldsymbol{b}^*)| = n + 1$ we obtain the desired result. $|\mathcal{T}_{\min}| = 0$ is obvious. □

Given the above result, $f(\boldsymbol{b};\ell,0)$ and $f(\boldsymbol{b};\ell,\ell-1)$ play distinguished roles as bounding binary trap functions: $f(\boldsymbol{b};\ell,0)$ is the easiest trap function over $\{0,1\}^\ell$ and in fact achieves $|\mathcal{T}_{\min}| = 0$, while $f(\boldsymbol{b};\ell,\ell-1)$ is the hardest trap function over $\{0,1\}^\ell$, achieving $|\mathcal{T}_{\max}| = 2^\ell - \ell - 1$ as follows from (6).

A trap Markov chain is induced by a problem instance (such as the ones in Fig. 3) and an SLS algorithm, specifically SIMPLESLS, along with its input parameters including noise parameter p. Given a trap functions $g(x;\ell,z)$ we construct a trap Markov chain $(\mathcal{S},\mathcal{V},\mathcal{P})$ such that $|\mathcal{S}| = \ell + 1$. When constructing $\mathcal{P}$, we take into account the underlying binary trap function $f(\boldsymbol{b};\ell,z)$, and also map the slope-change location z into a slope-change state. The following definition of a trap Markov chain (TMC) describes the performance of SIMPLESLS on a trap function as stated formally in Theorem 25.

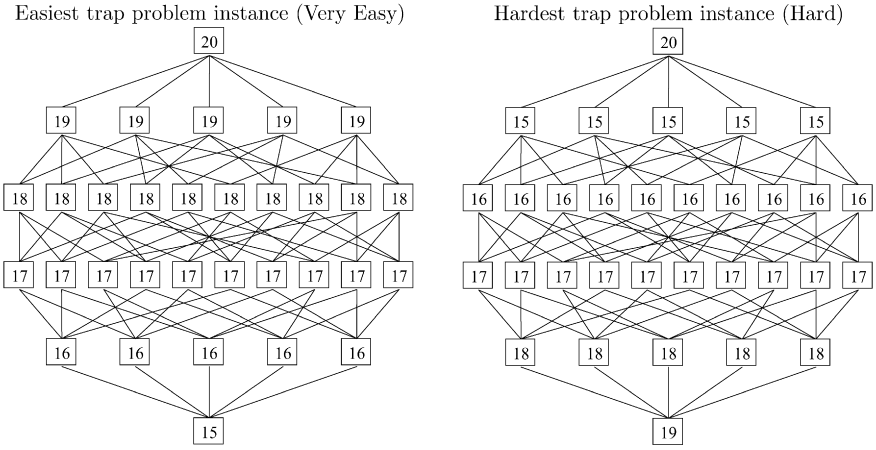

Fig. 4. The complete search spaces for two trap functions $f(\boldsymbol{b}; 5, 0)$ and $f(\boldsymbol{b}; 5, 4)$ representing hypothetical SAT problem instances, namely the easiest and hardest ones according to the trap function definition. The number of satisfied clauses is shown for each point in the search spaces, which are laid out as shown in Fig. 1.

Definition 24 *(Trap Markov chain).* An ℓ-bit trap Markov chain with change state z and noise parameter p, abbreviated as $\mathrm{TMC}(p; \ell, z)$, is defined as follows. It has $\ell + 1$ states $\mathcal{S} = \{0, \ldots, \ell\}$ and the initial distribution $\mathcal{V}$ is defined, for $x \in \mathcal{S}$, as

$$\Pr(X_1 = x) = \frac{\binom{\ell}{x}}{2^\ell}. \tag{7}$$

The state transition probabilities of $\mathcal{P}$ are for $X_i = \ell$ defined as

$$\Pr(X_{i+1} = \ell \mid X_i = \ell) = 1 - p$$
$$\Pr(X_{i+1} = \ell - 1 \mid X_i = \ell) = p,$$

and for $X_i = 0$, when $z > 0$, defined as

$$\Pr(X_{i+1} = 0 \mid X_i = 0) = 1 - p$$
$$\Pr(X_{i+1} = 1 \mid X_i = 0) = p,$$

while $\Pr(X_{i+1} = 1 \mid X_i = 0) = 1$ when $z = 0$. For internal states $X_i \neq 0$ and $X_i \neq \ell$ we have

$$\Pr(X_{i+1} = x + 1 \mid X_i = x) = \begin{cases} \frac{\ell - x}{\ell} p & \text{for } 1 \leqslant x \leqslant z - 1 \\ 1 - \frac{x}{\ell} p & \text{for } z \leqslant x \leqslant \ell - 1 \end{cases}$$
$$\Pr(X_{i+1} = x - 1 \mid X_i = x) = \begin{cases} 1 - \frac{\ell - x}{\ell} p & \text{for } 1 \leqslant x \leqslant z - 1 \\ \frac{x}{\ell} p & \text{for } z \leqslant x \leqslant \ell - 1 \end{cases}$$

with $\Pr(X_{i+1} = y \mid X_i = x) = 0$ for all x and y not listed above.

We emphasize that trap Markov chains (TMCs) are idealized models. Quantitatively, the purpose of TMCs is to provide an interesting range of expected hitting times $h(p)$ including bounding cases (for examples see the lower bound Very Easy $h_{5,0}(p)$ and the upper bound Hard $h_{5,4}(p)$ in Section 5.1). Qualitatively, the purpose of TMCs is to display different shapes of expected hitting time curves, ranging from ones that are monotonically increasing with p (and where $p = 0$ is the optimal noise level) to ones with a decreasing–increasing pattern and optimal noise levels that increase with the value of the slope-change state z. One should not expect to find real-world models of non-trivial size that exactly match TMCs, and our intention is not to fit real SLS behavior to TMCs. For results showing that these idealized models aid in the understanding of experiments with real problem instances, we refer to Section 7.1.

Examples of trap Markov chains are presented in Section 5. The reason for using the terminology "trap Markov chain" in Definition 24 should become clear with the following result, where we formally state how trap functions are processed by SIMPLESLS.

Theorem 25 *(Trap Markov chain). Let $f(\boldsymbol{b}; \ell, z)$ be an ℓ-bit binary trap function with slope-change location z. If f is given as input to* SIMPLESLS *along with noise probability p, then a trap Markov chain* $\mathrm{TMC}(p; \ell, z)$ *is simulated up to MAX-FLIPS flips.*

Proof. Results for the boundary states ℓ and 0 follow easily with the exception of $z = 0$. For $z = 0$, Lemma 13 applies with $\boldsymbol{b}_j = 0 \ldots 0$ and hence $\Pr(X_{i+1} = 1 \mid X_i = 0) = 1$. We now turn to the internal states $0 < x < \ell$. First, consider $\Pr(X_{i+1} = x + 1 \mid X_i = x)$. Conditioning on SIMPLESLS operators o_N and o_G and using the law of total probability gives

$$\begin{aligned}\Pr(X_{i+1} = x + 1 \mid X_i = x) = {} & \Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_N)\Pr(O_i = o_N) \\ & + \Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_G)\Pr(O_i = o_G).\end{aligned}$$

There are two cases to consider, namely Case (i) $0 < x < z$ and Case (ii) $z \leqslant x < \ell$. Case (i): Suppose that $0 < x < z$. In this case $g(x + 1) < g(x - 1)$, which means that SIMPLESLS only moves from x to $x + 1$ in case of a noise operation o_N, in other words $\Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_G) = 0$ while $\Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_N) > 0$. Thus, we obtain

$$\Pr(X_{i+1} = x + 1 \mid X_i = x) = \Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_N)\Pr(O_i = o_N).$$

Since $\Pr(O_i = o_N) = p$ and $\Pr(X_{i+1} = x + 1 \mid X_i = x, O_i = o_N) = (\ell - x)/\ell$, we get

$$\Pr(X_{i+1} = x + 1 \mid X_i = x) = \frac{\ell - x}{\ell} p,$$

which by law of total probability and the fact that we have a random walk without internal self-loops gives

$$\Pr(X_{i+1} = x - 1 \mid X_i = x) = 1 - \frac{\ell - x}{\ell} p.$$

Case (ii): Next, suppose that $z \leqslant x < \ell$. In this case $g(x + 1) > g(x - 1)$. The derivation is similar to above, resulting in

$$\begin{aligned}\Pr(X_{i+1} = x - 1 \mid X_i = x) &= \frac{x}{\ell} p \\ \Pr(X_{i+1} = x + 1 \mid X_i = x) &= 1 - \frac{x}{\ell} p,\end{aligned}$$

thus concluding the proof. □

Trap Markov chains are related to the naive Markov chain model in Section 4.2 as well as the so-called simple and branched Markov chain models introduced by Hoos [15]. The simple and branched Markov chain models capture similar phenomena, but our trap Markov chains have a few novel and important features. First, a trap Markov chain $\mathrm{TMC}(p; \ell, z)$ is parametrized with a noise parameter p, which is essential when analyzing the impact of noise on SLS. Second, while it is based on empirical considerations, the trap Markov chain is an analytically derived model (see above as well as earlier work [6]). Markov chains have also been used in the analysis of genetic and evolutionary algorithms [3,9,12,48]. Generally, this analysis emphasizes population-level effects and is quite different from our analysis in this article. In particular, we do not know of any analysis of genetic or evolutionary algorithms that includes noise level p as an explicit parameter. Our approach allows, for example, for the derivation of closed form solutions for expected hitting times which is defined as follows.

Definition 26 *(TMC hitting time).* The notation $h_{k,z}(p)$ is used for the expected hitting time for a trap Markov chain $\mathrm{TMC}(p; k, z)$.

Expected hitting times are, in a sense, more important than their underlying Markov chains, and we often do not explicitly show the Markov chains. There are several reasons for our emphasis on expected hitting times: First, their empirical counterpart, noise response curves, are a common way of reporting results in the SLS literature [7,8,14,16, 26]. Second, optimal noise levels p^* can be derived analytically from expected hitting time formulas as we will see in Section 5. Third, displaying the Markov chain transition matrices is in practice impossible for the many non-trivial search spaces used in this article, while expected hitting time expressions are often compact. We do, however, show and discuss the Markov chains for a few smaller problem instances in Section 7.1.

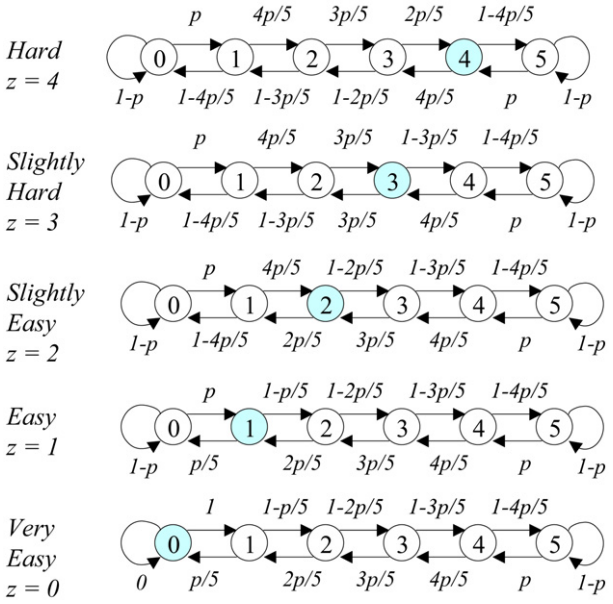

Fig. 5. Trap Markov chain models TMC(p; 5, z) for bitstrings of length $\ell = 5$, and where the slope-change state z is colored and varies from $z = 0$ (at the bottom) to $z = 4$ (at the top). Each of these trap Markov chain models represents the combined effect of a problem instance (see Fig. 3) and SIMPLESLS. These are top–down models where each state represents multiple states in the underlying exact Markov chain model of the SLS process.

5. Trap Markov chain examples

In this section we illustrate the trap Markov chain model, presented in Section 4.3, by means of examples.

5.1. Analysis of trap Markov chains

Perhaps the easiest way to illustrate the utility of trap Markov chain models is to discuss concrete problem instances.

Example 27 *(Trap Markov chains).* See Fig. 5 for graph representations of the transition probabilities for the TMC(p; 5, 0), TMC(p; 5, 1), TMC(p; 5, 2), TMC(p; 5, 3), and TMC(p; 5, 4) Markov chains.

In preparation for a formal result, we provide some intuition. Let us consider TMC(p; 5, 0), the very easy case, and suppose first that $p = 0$. It is then easily seen from Fig. 5 that regardless of the initial state $0 \leqslant X_0 \leqslant 5$, SIMPLESLS search proceeds directly towards state 5 without making any transitions from $X_i = k$ to $X_{i+1} = k - 1$ for $k > 1$, $i > 0$. Once $p > 0$, there is a chance that SIMPLESLS makes such transitions, and the search becomes longer.

The following result, illustrated in Fig. 6, gives expected hitting times for SIMPLESLS as formalized in Example 27. This shows how the noise probability p impacts the expected hitting time $h_{k,z}(p)$ for problem instances of varying difficulty. Difficulty increases with z, reflecting Lemma 22.

Lemma 28. *Assuming* SIMPLESLS *initialization uniformly at random, the expected hitting times for* TMC(p; 5, z), *where* $0 \leqslant z < 5$, *are for* SIMPLESLS *as follows*:

$$h_{5,0}(p) = \frac{24p^4 - 1870p^3 + 2875p^2 - 625p - 7500}{8(p-5)(2p-5)(20p + 3p^2 - 25)}$$

$$h_{5,1}(p) = \frac{48\,875p - 39\,175p^2 + 17\,865p^3 + 194p^4 + 625}{32p(4p-5)(3p-5)(p-5)(2p-5)}$$

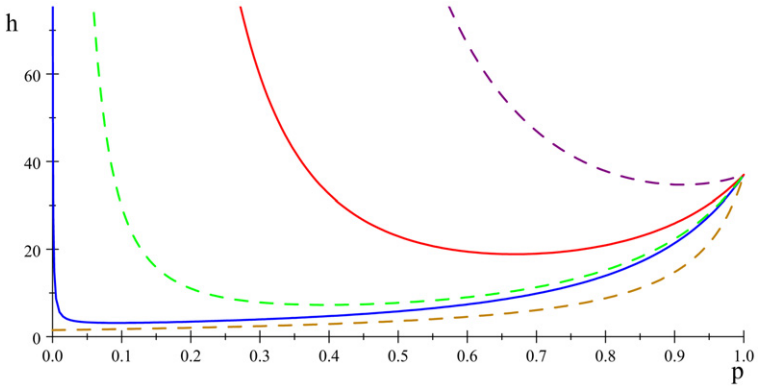

Fig. 6. The expected hitting time $h(p)$ as a function of noise probability p for the five 5-bit trap Markov chain models. The curves are, starting at the bottom, for: $h_{5,0}(p)$ (dashed brown), $h_{5,1}(p)$ (solid blue), $h_{5,2}(p)$ (dashed green), $h_{5,3}(p)$ (solid red), $h_{5,4}(p)$ (dashed purple). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$h_{5,2}(p) = \frac{750p - 18\,625p^2 + 9670p^3 - 4112p^4 - 1875}{64p^2(4p-5)(3p-5)(2p-5)}$$
$$h_{5,3}(p) = \frac{33p^4 + 1465p^3 + 1550p^2 - 750p + 1250}{48p^3(4p-5)(3p-5)}$$
$$h_{5,4}(p) = \frac{-912p^4 - 4530p^3 - 3875p^2 + 3250p - 8125}{384p^4(4p-5)}$$

Proof. The Markov chain's initial probability distribution is

$$\mathcal{V} = (\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5),$$

which according to (7) is

$$\mathcal{V} = \left(\frac{\binom{5}{0}}{2^5}, \frac{\binom{5}{1}}{2^5}, \frac{\binom{5}{2}}{2^5}, \frac{\binom{5}{3}}{2^5}, \frac{\binom{5}{4}}{2^5}, \frac{\binom{5}{5}}{2^5}\right). \tag{8}$$

We focus on $h_{5,3}(p)$. In order to obtain $h_{5,3}(p)$ from the TMC$(p; 5, 3)$ model we form the following simultaneous system of equations of expected first passage times m_i for $0 \leqslant i \leqslant 5$:

$$m_0 = 1 + (1-p)m_0 + pm_1$$
$$m_1 = 1 + \left(1 - \frac{4}{5}p\right)m_0 + \frac{4}{5}pm_2$$
$$m_2 = 1 + \left(1 - \frac{3}{5}p\right)m_1 + \frac{3}{5}pm_3$$
$$m_3 = 1 + \frac{3}{5}pm_2 + \left(1 - \frac{3}{5}p\right)m_4$$
$$m_4 = 1 + \frac{4}{5}pm_3 + \left(1 - \frac{4}{5}p\right)m_5$$
$$m_5 = 0,$$

which when solved gives first passage times $\{m_0, \ldots, m_5\}$ as follows

$$m_0 = \frac{775p^2 - 375p - 205p^3 + 204p^4 + 625}{300p^3 - 420p^4 + 144p^5}$$
$$m_1 = \frac{475p^2 - 375p + 215p^3 + 60p^4 + 625}{300p^3 - 420p^4 + 144p^5}$$

$$m_2 = \frac{925p^2 - 750p + 140p^3 + 24p^4 + 625}{300p^3 - 420p^4 + 144p^5}$$
$$m_3 = \frac{285p^2 - 50p - 60p^3 + 125}{100p^2 - 140p^3 + 48p^4}$$
$$m_4 = \frac{15p + 22p^2 + 25}{25p - 35p^2 + 12p^3}\, m_5 = 0.$$

Introducing (8), we now compute $E(T \mid X_0 = i)\Pr(X_0 = i) = m_i\pi_i$, for $0 \leqslant i \leqslant 5$, as follows

$$m_0\pi_0 = \frac{775p^2 - 375p - 205p^3 + 204p^4 + 625}{300p^3 - 420p^4 + 144p^5}\frac{\binom{5}{0}}{2^5}$$
$$m_1\pi_1 = \frac{475p^2 - 375p + 215p^3 + 60p^4 + 625}{300p^3 - 420p^4 + 144p^5}\frac{\binom{5}{1}}{2^5}$$
$$m_2\pi_2 = \frac{925p^2 - 750p + 140p^3 + 24p^4 + 625}{300p^3 - 420p^4 + 144p^5}\frac{\binom{5}{2}}{2^5}$$
$$m_3\pi_3 = \frac{285p^2 - 50p - 60p^3 + 125}{100p^2 - 140p^3 + 48p^4}\frac{\binom{5}{3}}{2^5}$$
$$m_4\pi_4 = \frac{15p + 22p^2 + 25}{25p - 35p^2 + 12p^3}\frac{\binom{5}{4}}{2^5}.$$

Adding up, we obtain $h_{5,3}(p) = \sum_{i=0}^{5} m_i\pi_i$ as desired. The remaining $h_{5,j}(p)$ for $j \in \{0, \ldots, 2\}$ and $j = 4$ are developed in a similar manner and to save space we do not detail the proof here. □

Illustrating Lemma 28, the graphs in Fig. 6 show the impact of varying noise p for the example TMCs. The difference in the shapes of the curves for the easiest case $h_{5,0}(p)$, compared to the hardest case $h_{5,4}(p)$, is quite dramatic. One extreme, $h_{5,0}(p)$, is monotonically increasing. The other extreme, $h_{5,4}(p)$, is first monotonically decreasing, then monotonically increasing. Clearly, this has an impact on the optimal noise level, which we discuss next.

5.2. Optimal noise level for trap Markov chains

Fig. 6 clearly shows the difference in optimal noise levels p^*. For $h_{5,0}(p)$, it is intuitively clear that one should use low noise p, more specifically $p^*_{5,0} = 0$, since this enables SIMPLESLS to greedily hill-climb to $\boldsymbol{b} = 11111$ without taking unnecessary downhill noise steps. For $h_{5,4}(p)$, on the other hand, one should intuitively use high noise p in order to let SIMPLESLS more easily escape the trap $\boldsymbol{b} = 00000$ containing the local (but non-global) optimum. Given that expected hitting times $h(p)$ are rational functions, for example as derived in Lemma 28, optimal noise probabilities can be derived analytically. The following example of deriving optimal noise p^* illustrates the use of the quotient rule.

Example 29. Consider $h_{5,3}(p)$ from Theorem 28; using (16) gives

$$h'_{5,3}(p) = -\frac{198p^6 + 17\,580p^5 + 1850p^4 - 72\,250p^3 + 96\,250p^2 - 106\,250p + 46\,875}{3456p^8 - 20\,160p^7 + 43\,800p^6 - 42\,000p^5 + 15\,000p^4},$$

and by solving for p in $h'_{5,3}(p) = 0$ and checking boundaries $p = 0$ and $p = 1$ we obtain optimal noise probability $p^*_{5,3} = 0.6687$.

5.3. Discussion of trap Markov chains

Several points may be made with respect to Lemma 28 and Fig. 6. First, we note that these are all convex rational functions of the form $h(p) = P(p)/Q(p)$, where $P(p)$ and $Q(p)$ are polynomials. Further, if we for a short moment disallow the use of restarts, these examples illustrate how $p = 0$ can lead to unbounded hitting times $h(p)$; see Fig. 6.

The reason for this is that $p = 0$ can, for certain (unfortunate) initializations, give unbounded search in the trap part of TMCs. Fig. 6 clearly shows the unboundedness of $h_{5,z}(p)$ for $z \geqslant 1$ and $p = 0$. Similar trapping effects with respect to local minima can take place in real problem instances, illustrating the need for $p > 0$ or MAX-FLIPS $< \infty$ in SLS.[2]

Second, while the trap function setting is restricted, similar patterns appear in experiments in the literature [14,26] as well as in Section 7. For instance, six different SLS algorithms each had, when tested on 400 hard random 3SAT problem instances using variable noise, a relatively clear noise level where it performed optimally [26]. (Note that the dependent variable was fraction of problem instances solved rather than mean run time, thus the curves were concave, with a maximum value as optimum, rather than convex as ours [26].) Along similar lines, three empirical noise response curves for the Novelty$^+$ SLS algorithm all have convex shapes similar to $h_{5,3}(p)$ [14]. In all these cases, performance improves with increasing noise until it hits an optimum, and performance then deteriorates with increasing noise. The pattern is in other words similar to the curves for $h_{5,2}(p)$, $h_{5,3}(p)$, and $h_{5,4}(p)$. The benefit of very small noise levels, illustrated by $h_{5,0}(p)$ and $h_{5,1}(p)$ has, to our knowledge, received less attention in the literature. However, there are SLS results in the areas of planning and scheduling where small noise levels have empirically been shown to be optimal [7,8].

Third, we notice that the curves for $h_{5,z}(p)$, for $0 \leqslant z \leqslant 4$, get closer as p increases, and in particular that

$$\lim_{p \to 1} h_{5,0}(p) = \cdots = \lim_{p \to 1} h_{5,4}(p) = \frac{887}{24}.$$

There is a much greater performance difference between the problem instances for small p compared to for large p. Specifically, fix p_1 and Δp, say $p_1 = 0.7$ and $\Delta p = 0.1$. Now, form $p_1^- = p_1 - \Delta p = 0.6$ and $p_1^+ = p_1 + \Delta p = 0.8$. Clearly, the difference in performance between the problem instances $h_{5,0}(p)$ through $h_{5,4}(p)$ is much greater for p_1^- than for p_1^+. This suggest that in general, setting the noise level too low, to $p^* - \Delta p$, may be more detrimental than setting it too high, to $p^* + \Delta p$, when operating under conditions of uncertainty about p^* and the problem instance distribution. Similar recommendations have in fact already been made based on experimental observations [17].

We believe these results shed additional light on the significant impact of varying noise as observed in experiments [17,26,45]. When one does not know much about the problem instance distribution ahead of time, these results may also argue in favor of the use of adaptive noise [14,26].

6. Hitting time analysis

Experimentally, it has been observed that curves for mean SLS run times [14,16] and the fraction of solved instances when using SLS [26], plotted as functions of noise p, have shapes suggesting underlying convex functions as well as rational functions for hitting times. We note that convexity of hitting times is also observed in Section 5.1 and in experiments in Section 7.

We now provide several general SLS hitting time results, focusing on rational functions, convexity, and polynomials. The justifications are a bit different for these cases. The condition of rationality is supported by our analysis in Section 6.1 below. For convexity, the justification is empirical results in the literature as well as in this article. For polynomials, our justification is partly Weierstrass' theorem, partly our empirical results (see Section 7). The analysis is always with respect to a specific SLS model $(\mathcal{M}, \mathcal{O})$, where $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{P})$ has $k := |\mathcal{S}|$ states. Further, T is (as before) a random variable representing hitting time.

The results in this section apply to hitting times in general, and do not depend on the approximate Markov chain models developed in Section 4 and Section 5. In other words, readers who found the approximate Markov chain models too restrictive may still want to consider the more general analysis provided in this section.

6.1. Single problem instances

We consider single problem instances and assume that the noise probability $p = \Pr(O = o_N)$ is the independent parameter. Let $P(p)$ and $Q(p)$ be polynomials. Based on our results in Section 5.1, one might hypothesize that the

[2] An alternative way of escaping from traps is to use restarts, and an SLS practitioner will most likely use both non-zero noise and non-infinite MAX-FLIPS. While the topic of restart is crucially important in SLS, our main focus in this article is not on the effect of restart, but on the effect of noise on SLS, and a detailed discussion of the joint effect of restarts and noise is beyond the scope of this article.

expected hitting time for a problem instance has the form of a rational function $h(p) = P(p)/Q(p)$, and this is indeed supported by the following analysis. From linear algebra we know that an $n \times n$ system of equations has no solution, exactly one solution, or infinitely many solutions. While in theory there might exist conditions under which a system of hitting time equations for SLS have no or infinitely many solutions, the one solution case is clearly of greatest practical interest. We will in this article assume the existence of exactly one solution that can be found by Gaussian elimination, as discussed below, in a finite number of steps. In the proof of the following theorem, the key idea is to perform Gaussian elimination in a symbolic fashion such that the noise parameter p is preserved throughout the derivation.

Theorem 30. *Consider an SLS model $(\mathcal{M}, \mathcal{O})$, where the Markov Chain $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{P})$ is defined over a bitstring of length n and with noise parameter p. Let $\kappa = 2^n$, assume optimum states $\{s_\lambda, \ldots, s_\kappa\} = \mathcal{O}$ where $\lambda \leqslant \kappa$, and form a system of equations for $\mathcal{M}$'s expected first passage times m_i for $1 \leqslant i \leqslant \kappa$. There exists an equivalent upper triangular system $\boldsymbol{U}\boldsymbol{m} = \boldsymbol{b}$, where $\boldsymbol{m} = (m_1, \ldots, m_{\lambda-1})^T$, in which all coefficients in $\boldsymbol{U}$ and $\boldsymbol{b}$ are rational functions of p.*

Proof. We form, based on $\mathcal{M}$, this system of equations for expected first passage times into $\mathcal{O}$:

$$
\begin{aligned}
m_1 &= 1 + f_{1,1}(p)m_1 + f_{1,2}(p)m_2 + \cdots + f_{1,\kappa-1}(p)m_{\kappa-1} + f_{1,\kappa}(p)m_\kappa \\
m_2 &= 1 + f_{2,1}(p)m_1 + f_{2,2}(p)m_2 + \cdots + f_{2,\kappa-1}(p)m_{\kappa-1} + f_{2,\kappa}(p)m_\kappa \\
&\cdots \\
m_{\lambda-1} &= 1 + f_{\lambda-1,1}(p)m_1 + f_{\lambda-1,2}(p)m_2 + \cdots + f_{\lambda-1,\kappa-1}(p)m_{\kappa-1} + f_{\lambda-1,\kappa}(p)m_\kappa \\
m_\lambda &= 0 \\
&\cdots \\
m_\kappa &= 0,
\end{aligned}
$$

where $f_{j,i}(p) = (\alpha_{j,i} + \beta_{j,i}p)/\gamma_{j,i}$ for constants $\alpha_{j,i}, \beta_{j,i} \in \mathbb{N}, \gamma_{j,i} \in \mathbb{N}^+$, and $\sum_{i=1}^{\kappa} f_{j,i}(p) = 1$ for $1 \leqslant j \leqslant \kappa$. Clearly, this system can be written as

$$
\begin{aligned}
\big(1 - f_{1,1}(p)\big)m_1 - f_{1,2}(p)m_2 - \cdots - f_{1,\kappa-1}(p)m_{\kappa-1} - f_{1,\kappa}(p)m_\kappa &= 1 \\
-f_{2,1}(p)m_1 + \big(1 - f_{2,2}(p)\big)m_2 - \cdots - f_{2,\kappa-1}(p)m_{\kappa-1} - f_{2,\kappa}(p)m_\kappa &= 1 \\
&\cdots \\
-f_{\lambda-1,1}(p)m_1 - f_{\lambda-1,1}(p)m_1 - \cdots + \big(1 - f_{\lambda-1,\kappa-1}(p)\big)m_{\lambda-1} - f_{\lambda-1,\kappa}(p)m_\kappa &= 1 \\
m_\lambda &= 0 \\
&\cdots \\
m_\kappa &= 0.
\end{aligned}
$$

We proceed by means of induction on the number of elementary operations, iteratively creating a system of equations $S^{(t+1)}$from system $S^{(t)}$ for $t \geqslant 1$. Here, $S^{(1)}$ is the following system created from the above by dropping the trivially rational $m_\lambda = \cdots = m_\kappa = 0$, substituting $m_\lambda = \cdots = m_\kappa = 0$ into the other equations, and performing a slight renaming:

$$
\begin{aligned}
a^{(1)}_{1,1}(p)m_1 + a^{(1)}_{1,2}(p)m_1 + \cdots + a^{(1)}_{1,\lambda-1}(p)m_{\lambda-1} &= b^{(1)}_1(p) \\
a^{(1)}_{2,1}(p)m_1 + a^{(1)}_{2,2}(p)m_2 - \cdots + a^{(1)}_{2,\lambda-1}(p)m_{\lambda-1} &= b^{(1)}_2(p) \\
&\cdots \\
a^{(1)}_{\lambda-1,1}(p)m_1 + a^{(1)}_{\lambda-1,2}(p)m_2 - \cdots + a^{(1)}_{\lambda-1,\lambda-1}(p)m_{\lambda-1} &= b^{(1)}_{\kappa-1}(p).
\end{aligned}
$$

For the base case $t = 1$, $a^{(1)}_{i,j}(p)$ and $b^{(1)}_i(p)$ are clearly rational. Following Gaussian elimination, we have for the tth step (where $t \geqslant 2$):

$$r_{i,k}^{(t)}(p) = a_{i,k}^{(t-1)}(p)/a_{k,k}^{(t-1)}(p) \quad \text{for } k+1 \leqslant i \leqslant \lambda - 1 \text{ and } t \leqslant k \leqslant \lambda - 1 \tag{9}$$

$$a_{i,j}^{(t)}(p) = a_{i,j}^{(t-1)}(p) - r_{i,k}^{(t)} \times a_{k,j}^{(t-1)}(p) \quad \text{for } k+1 \leqslant i, j \leqslant \lambda - 1 \tag{10}$$

$$b_{i}^{(t)}(p) = b_{i}^{(t-1)}(p) - r_{i,k}^{(t)} \times b_{k}^{(t-1)}(p) \quad \text{for } k+1 \leqslant i, j \leqslant \lambda - 1 \text{ and } t \leqslant k \leqslant \lambda - 1 \tag{11}$$

Under the inductive hypothesis that $a_{i,k}^{(t-1)}(p), a_{k,k}^{(t-1)}(p), a_{i,j}^{(t-1)}(p), a_{k,j}^{(t-1)}(p), b_{i}^{(t-1)}(p)$ and $b_{k}^{(t-1)}(p)$ are all rational, it follows that the left hand sides in (9), (10), and (11) are all rational as well, since rationality is closed under addition and multiplication as applied there. (Rational functions are in abstract algebra known to be fields, which are closed under multiplication and addition.) Consequently, after a finite number of Gaussian elimination steps, an upper triangular system $\boldsymbol{U}\boldsymbol{m} = \boldsymbol{b}$ is obtained where all coefficients in $\boldsymbol{U}$ and $\boldsymbol{b}$ are rational functions in p. □

The above theorem creates upper triangular systems; we now turn to the impact of Gaussian back elimination on such systems.

Theorem 31. *Suppose that we have a system of equations, in upper triangular form $\boldsymbol{U}\boldsymbol{m} = \boldsymbol{b}$, for expected first passage times $m_i(p)$, where $1 \leqslant i \leqslant \lambda - 1 < \kappa$. Further suppose that all entries in $\boldsymbol{U}$ and $\boldsymbol{b}$ are non-zero rational functions of p. Then there exists a rational function $P_i(p)/Q_i(p)$ such that $m_i(p) = P_i(p)/Q_i(p)$.*

Proof. We perform back substitution on $\boldsymbol{U}\boldsymbol{m} = \boldsymbol{b}$. For arbitrary $m_i(p)$ we consequently have:

$$m_i(p) = \frac{b_i(p) - \sum_{k=i+1}^{\lambda-1} a_{i,k}(p) m_k(p)}{a_{i,i}(p)} \quad \text{for } i = \lambda - 1, \lambda - 2, \ldots, 1.$$

From Theorem 30 we know that $b_i(p)$ in $\boldsymbol{b}$ and $a_{i,k}(p)$, and $a_{i,i}(p)$ in $\boldsymbol{U}$ are rational, hence any $m_i(p)$ above is also rational due to closure under addition and multiplication. □

The above result shows how the noise probability p impacts the expected first passages times m_i, where $1 \leqslant m_i \leqslant 2^n$. In particular, we have rational function $m_i(p) = P_i(p)/Q_i(p)$ where we put $P_i(p) = 0$ for $\lambda \leqslant i \leqslant \kappa$.

We next show that the expected hitting time $E(T)$ is a rational function $h(p)$, if we assume that first passage times are rational functions.

Theorem 32. *Let the first passage time $m_i(p) = E(T \mid X_0 = i)$ be a rational function of p for any $1 \leqslant i \leqslant k$, and suppose that $\Pr(X_0 = i)$ is constant. Then the expected hitting time $E(T)$ is a rational function of p, $h(p)$.*

Proof. For random variables T and C, the law of conditional expectation says that $E(T)$ is given by

$$E(T) = \sum_{i \in \Omega(C)} E(T \mid C = i) \Pr(C = i). \tag{12}$$

Here, we have $\Omega(C) = \{1, \ldots, k\}$ and $E(T \mid C = i) = E(T \mid X_0 = i) = m_i(p)$. Since rational functions are closed under multiplication with constant $\Pr(X_0 = i)$, and under addition, we obtain the desired result. □

The following result follows easily from our results so far in this section and gives expected hitting times for SIMPLESLS in general.

Corollary 33 *(Rationality of* SIMPLESLS *hitting time). Consider an SLS model $(\mathcal{M}, \mathcal{O})$, where $\mathcal{M} = (\mathcal{S}, \mathcal{V}, \mathcal{P})$ is an exact Markov Chain defined over a bitstring of length n and with noise parameter p. The expected hitting time for $\mathcal{M}$ is a rational function of p, $h(p) = P(p)/Q(p)$, where $P(p)$ and $Q(p)$ are polynomials.*

Proof. From Theorem 31 it follows that first passage times are $m_i(p) = P_i(p)/Q_i(p)$ for $1 \leqslant i \leqslant 2^n$, where $P_i(p)$ and $Q_i(p)$ are polynomials. Applying Theorem 32, it follows that the expected hitting time $h(p)$ for $\mathcal{M}$ is a rational function $h(p)$. □

The above result generalizes Theorem 28, and shows that rational functions are of great interest in the analysis of SLS. In particular, they are analytical counterparts to noise response curves from the experimental literature on SLS [7,8,14,16,26].

For a couple of reasons, we often employ polynomial regression rather than rational function regression. The first reason is that over an interval $[a, b]$, in our case $[a, b] = [0, 1]$ since $0 \leqslant p \leqslant 1$, Weierstrass' celebrated theorem tells us that polynomials $P(x)$ that give arbitrary good approximations exist.

Theorem 34 *(Weierstrass). Suppose that $f(x)$ is continuous on $[a, b]$ and let $\varepsilon > 0$. Then there exists a polynomial $P(x)$ such that*

$$\left\| f(x) - P(x) \right\| < \varepsilon,$$

where $\| \cdot \|$ denotes the uniform norm over the interval $[a, b]$.

The second reason for our use of polynomial regression is that it is better understood and more wide-spread than rational function regression. For these reasons we use polynomial regression rather than rational function regression in experiments in Section 7.

We now provide a sufficient condition for the expected hitting time $E(T)$ to be a convex function $h(p)$.

Theorem 35. *Let the first passage time $m_i = E(T \mid X_0 = i)$ be a convex function of p for any $1 \leqslant i \leqslant k$, and suppose that $\Pr(X_0 = i)$ is constant. Then the expected hitting time $E(T)$ is a convex function of p, $h(p)$.*

Proof. Similar to the proof of Theorem 32, we use conditional expectation (12) and observe that convexity is preserved under nonnegative multiplication and addition, and we obtain the desired result. □

Experimental results giving noise response curves for individual problem instances are reported in Section 7.1 (for synthetic instances) and in Section 7.2 (for BNs from applications).

6.2. Mixtures of problem instances

We now investigate multiple problem instances, or classes of instances, using finite mixture distributions. Again, we consider noise probability $p = \Pr(O = o_N)$ to be the independent parameter. We assume that our problem instances come from a probability distribution as follows.

Definition 36 *(SLS mixture model).* Suppose there are ξ SLS models $\{c_1, \ldots, c_\xi\}$, with $c_i = (\mathcal{M}_i, \mathcal{O}_i)$ for $1 \leqslant i \leqslant \xi$. If each SLS model is observed with probability $\Pr(C = c_i)$, where $\sum_{i=1}^{\xi} \Pr(C = c_i) = 1$, then $\{(c_1, \Pr(C = c_1)), \ldots, (c_\xi, \Pr(C = c_\xi))\}$ defines an SLS mixture model.

Definition 37 *(First passage time, mixture).* Consider an SLS mixture model $\{(c_1, \Pr(C = c_1)), \ldots, (c_\xi, \Pr(C = c_\xi))\}$. Let T be a conditional first passage time random variable $\Pr(T = t \mid C = c_i)$ for $1 \leqslant i \leqslant \xi$. The first passage time of the mixture is defined as

$$\Pr(T = t) = \sum_{i=1}^{\xi} \Pr(T = t \mid C = c_i) \Pr(C = c_i).$$

This is a finite mixture distribution with ξ mixture components. There are a number of reasons why such mixtures are interesting for stochastic local search. Algorithms for NP-hard problem are typically developed with a class or mixture of problem instances in mind. Problem instances may be dynamically generated, and thus the generation process induces an SLS mixture. During early system design the system being modeled is not completely known, and a mixture of BNs may represent all possibilities [28]. Finally, a given C/V-ratio also defines a problem instance mixture [33,34] and in Section 7.2 we empirically investigate such mixtures.

In a mixture, there is a distinct Markov chain for each problem instance. Informally, we first pick the ith Markov chain with probability $\Pr(C = c_i)$, and then use that Markov chain to give $\Pr(T = t \mid C = c_i)$ along with the chain's passage time (Definition 3) and instance hitting time (Theorem 5).

Definition 38 *(SLS mixture hitting time).* Consider an SLS mixture model with ξ components and expected hitting times $h_j(p)$ for $1 \leqslant j \leqslant \xi$. The SLS mixture hitting time $H(p)$ is defined as

$$H(p) = \sum_{j=1}^{\xi} h_j(p)\Pr(C = c_j).$$

Having formally introduced $H(p)$, we next show what this function actually is. To keep the notation simple, we assume that the state space size is the same for each Markov chain $\mathcal{M}_i$ in the SLS mixture model $\{c_1, \ldots, c_\xi\} = \{(\mathcal{M}_1, \mathcal{O}_1), \ldots, (\mathcal{M}_\xi, \mathcal{O}_\xi)\}$.

Theorem 39. *Let, for an SLS mixture model with ξ mixture components, first passage time be a random variable T. The expected hitting time $E(T)$ is the mixture hitting time $H(p)$; $E(T) = H(p)$.*

Proof. The expected value for T for the mixture, given initial state $X_0 = s_i$ and component $C = c_j$ in the SLS mixture model, is defined similar to Definition 3 as $E(T \mid X_0 = s_i, C = c_j)$. Using the law of conditional expectation (12), we obtain

$$E(T \mid C = c_j) = \sum_{i=1}^{\kappa} E(T \mid X_0 = s_i, C = c_j)\Pr(X_0 = s_i \mid C = c_j). \tag{13}$$

Using on (13) the law of conditional expectation (12) again, we obtain

$$\begin{aligned} E(T) &= \sum_{j=1}^{\xi} E(T \mid C = c_j)\Pr(C = c_j) \\ &= \sum_{j=1}^{\xi} h_j(p)\Pr(C = c_j), \end{aligned} \tag{14}$$

where we used $E(T \mid C = c_j) = h_j(p)$ from Corollary 33. By Definition 38, (14) is $H(p)$ and we have the desired result. □

We now turn to rational functions; our interest in them is motivated by our results in Section 6.1.

Theorem 40. *Consider an SLS mixture model $\{(c_1, \Pr(C = c_1)), \ldots, (c_\xi, \Pr(C = c_\xi))\}$. Suppose that the individual component hitting times $h_i(p)$, for $1 \leqslant i \leqslant \xi$, are rational functions. Then the SLS mixture hitting time $H(p)$ is also a rational function.*

Proof. Since $h_i(p)$ by assumption is a rational function, it can be written as $h_i(p) = P_i(p)/Q_i(p)$, where $P_i(p)$ and $Q_i(p)$ are polynomials. Due to closure under multiplication, $\Pr(C = c_i)h_i(p) = \Pr(C = c_i)P_i(p)/Q_i(p)$ is also a rational function. There is also closure under addition, and thus

$$H(p) = \sum_{i=1}^{\xi} \Pr(C = c_i)h_i(p),$$

which is the definition of $H(p)$, is also a rational function. □

From the results above, the expected hitting time for SIMPLESLS for a mixture of problem instances can be derived.

Corollary 41 *(Rationality of* SIMPLESLS *mixture). Consider an SLS mixture model $\{(c_1, \Pr(C = c_1)), \ldots, (c_\xi, \Pr(C = c_\xi))\}$. Suppose that each $c_i = (\mathcal{M}_i, \mathcal{O}_i)$, for $1 \leqslant c_i \leqslant \xi$, is an exact* SIMPLESLS *model with noise parameter p. The expected hitting time for this mixture is a rational function of p, $H(p) = P(p)/Q(p)$, where $P(p)$ and $Q(p)$ are polynomials.*

Proof. From Corollary 33 it follows that a SIMPLESLS hitting time $h_i(p)$, where $1 \leqslant i \leqslant \xi$, is a rational function. Applying Theorem 40 we conclude that the mixture $H(p)$ is also a rational function. □

The trap Markov chain results presented in Sections 4.3 and 5 were originally intended to be models of individual problem instances. However, since the hitting time for a problem instance is a rational function, it follows—as stated above—that the hitting time for a mixture of problem instances (from some class, for example as defined by a particular C/V-ratio) is also a rational function. Hence, one can also use our trap Markov chain results as a model for the hitting time of a mixture of problem instances. This is a mathematical consequence of the functional form of hitting times rather than an artifact of our analysis.

We now turn our attention to convex functions.

Theorem 42. *Suppose that problem instance hitting times $h_i(p)$, for $1 \leqslant c_i \leqslant \xi$, are convex functions. Then the mixture hitting time $H(p)$ is also a convex function.*

Proof. Since $h_i(p)$ is convex, $\Pr(C = c_i)h_i(p)$ is convex due to the fact that convexity is preserved under nonnegative multiplication. Further,

$$H(p) = \sum_{i=1}^{\xi} \Pr(C = c_i)h_i(p)$$

is convex since convexity is preserved under addition. □

Convexity is important because local optimality means global optimality in convex functions, thus simplifying optimization algorithms. Polynomials are also helpful in noise optimization, and Section 7.2 contains experiments with mixtures of problem instances along with polynomial regression results.

6.3. Optimal noise levels

What is the optimal value p^* of an SLS noise parameter p? This is the question that will be discussed now, in light of our analysis earlier in this section.

Definition 43 *(Hitting time minimization).* Let the SLS noise be p and let $h(p)$ be an expected hitting time. The optimal noise probability, for minimizing $h(p)$, is defined as

$$p_h^* = \arg \min_{0 \leqslant p \leqslant 1} h(p), \tag{15}$$

with minimal expected hitting time $h^* = h(p_h^*)$.

We note that $h(p)$ in (15) can be the expected hitting time for one problem instance, $h(p) = h_i(p)$, or a problem instance distribution, $h(p) = H(p)$. Optimal noise level p_h^* is therefore with respect to one problem instance or a mixture of problem instances.

If we make certain assumptions about the form of $h(p)$, further progress can be made. We now consider differentiable $h(p)$, which it certainly is if it is a rational function or a polynomial. We may then take the derivative $h'(p)$ and in order to find the optimal SLS noise level p_h^* solve the equation $h'(p) = 0$. Forming $h'(p)$ for polynomial $h(p)$ is trivial, however it is a mathematical fact that in the general case, polynomials have complex roots. In other words, solving $h'(p) = 0$ or $r'(p) = 0$ (where $r(p)$ is expected run time, see Section 6.4) may give complex solutions only. Certain strict subsets of polynomials, for instance polynomials with real coefficients of odd degree, have at least one real root. Given this setting, there are two options for $h(p)$ or $r(p)$ of odd degree (so $h'(p) = 0$ and $r'(p)$ have even degree, thus they may not have at least one root): (i) Disregard all $h(p)$ or $r(p)$ of even degree a priori, since at least one real root for $h'(p) = 0$ or $r'(p) = 0$ is not guaranteed. (ii) Not disregard all $h(p)$ or $r(p)$ of even degree a priori, since even though there is no guarantee for one or more real roots, in many cases (for example, in all cases except two in Table 3 containing $r'(p)$) real roots may be found and provide useful insight. In a noise optimization algorithm

based on computing $h'(p) = 0$ one would perhaps prefer to follow approach (i). In this article, however, the purpose is analysis and insight rather than optimization algorithm development and thus we prefer (ii).

When $h(p)$ is a rational function, $h(p) = P(p)/Q(p)$, the well-known quotient rule has this form:

$$h'(p) = \frac{Q(p)P'(p) - P(p)Q'(p)}{Q(p)^2}. \tag{16}$$

In addition to a better understanding of the noise phenomenon, the results above may also serve—in future research—as a basis for improved algorithms for computing p_h^*. For example, rational function or polynomial regression may play a role in such improved algorithms. When p_h^* is estimated (in part) empirically, the notation $\hat{p}_h^*$ and terminology such as optimized noise level or estimated optimal noise level is used.

6.4. Initialization and restart

Using more advanced initialization algorithms than initialization uniformly at random has proven to be a powerful way to improve SLS performance [22,30,31,36,37]. SIMPLESLS can clearly support different ways of initializing $\boldsymbol{b}$, and SGS in fact takes an initialization algorithm portfolio $\mathbb{I}$ as input. In the Markov chain $\mathcal{M}$, the initial distribution $\mathcal{V}$ then needs to reflect the particular initialization algorithm portfolio. In general, each initialization algorithm or operator has a distinct initialization distribution $\mathcal{V}$, and Theorem 28 can be adapted correspondingly to reflect this.

Another technique, namely restarts or using MAX-FLIPS $< \infty$, has been shown to be beneficial in SLS [38] as well as in systematic search [10]. In many cases, restarts play a central role in SLS and using a close to optimal MAX-FLIPS value is essential for strong performance [38,43]; for TMCs this was observed in Section 5.3. In recent research, an approach to dynamically optimizing the SLS restart parameter MAX-FLIPS has been developed [41,42], based on learned Bayesian networks that predict inference run times [19].

In general, it is consequently important to distinguish between expected run time and expected hitting time. We now formally introduce expected run time.

Definition 44 *(Expected run time).* Let X, a random variable, be the number of flips performed by SIMPLESLS when the noise probability is p. The expected run time (or number of flips) is defined as $r(p) = E(X)$.

Note that the concepts of run time and expected run time do not (necessarily) use Markov chains. Our hitting time results, on the other hand, rely on the use of Markov chains. Unfortunately, SLS restarts, which may take place when MAX-FLIPS $\neq \infty$ in SIMPLESLS, may violate the Markov property.[3]

Expected run times are well-defined even when restarts occur, and we are interested in their minimization.

Definition 45 *(Run time minimization).* Let $r(p)$ be the expected run time (or number of flips) for SIMPLESLS. The optimal noise probability, which minimizes $r(p)$, is defined as

$$p_r^* = \arg \min_{0 \leqslant p \leqslant 1} r(p),$$

with minimal expected run time $r^* = r(p_r^*)$.

If MAX-FLIPS $= \infty$ then obviously $p_r^* = p_h^*$; however in general it is clear that $p_r^* \neq p_h^*$. In the theoretical part of this article we are discussing the optimal noise probability with respect to expected hitting time, p_h^*, while in the experimental part of this article we are in addition discussing optimal noise probability with respect to expected run time, p_r^*. As it should be clear from the context whether p_r^* or p_h^* is referred to, we will simply say "optimal noise probability" and p^*. In Section 7, we empirically investigate estimates of expected hitting times $\hat{h}(p)$ as well as expected run times $\hat{r}(p)$ using very similar techniques. It turns out that the noise response curves for expected run times are similar to those for expected hitting times.

[3] In making this claim, we assume that Markov chain states do not reflect the number of flips made. One could perhaps expand the exact Markov chain model to also reflect number of flips made by SIMPLESLS, however this is beyond the scope of this work.

7. Experiments

Our analysis made certain assumptions and simplifications that raise questions such as the following: Is there a relationship between the trap Markov chain model and real problem instances? What is the SLS search behavior on sets of problem instances, corresponding to SLS mixtures? What is the impact of varying noise, on SLS run times, in more realistic, large-scale problem instances from applications?

To answer these questions, we here report on real search behavior from experiments with Bayesian networks using SGS, an SLS system. There is strong evidence that a problem instance that is hard for one SLS algorithm is also hard for other SLS algorithms [17], hence we investigate one SLS system in depth. Stochastic greedy search (SGS), which can simulate SIMPLESLS, computes MPEs and supports multiple search operators and initialization operators [27,30]. SGS has these input parameters: β—a BN; f^*—MPE probability $\Pr(\boldsymbol{x}_i^*)$; $\mathbb{I}$—a portfolio of initialization algorithms; $\mathbb{S}$—a portfolio of search algorithms; MAX-FLIPS—the number of flips before a restart; and MAX-TRIES—the number of tries before termination. For $\mathbb{S}$, our focus is on greedy operators, named BM and GM, as well as noisy operators, named NU, BS, and GS. NU implements uniform noise o_N as described in Section 3. BS and GS are noisy but biased towards improving the current explanation's probability. BM and GM, which correspond to o_G, are pure hill-climbers and maximize gain without any noise. Without going into too much detail, which is covered elsewhere [27,30], suffice it to say that BM and BS operate on gain in probability based on (2), while GM and GS are probabilistic generalizations of GSAT gain [45,46] and have turned out to be powerful in BNs with many deterministic nodes [27,30].

In Section 7.1, based on 100 small synthetic BNs, we investigate the connection between trap functions, trap Markov chains, and real SGS search behavior. In Section 7.2 we investigate real SGS search behavior using 800 synthetic BNs, emphasizing BNs of varying C/V-ratio and approximation using polynomial regression. In Section 7.3, we experiment with SGS using application BNs, and also investigate noise strategies that go beyond uniform random noise as well as more advanced initialization algorithms than initialization uniformly at random.

7.1. Experiments with small synthetic Bayesian networks

What is the relationship between the TMC models from Sections 4 and 5 and real SLS search behavior? In order to clearly link the TMC analysis and experimental parts of this article, we here investigate 100 randomly generated 3SAT problem instances, represented as BNs, where the search space is the same size as the example trap Markov chains used in our analysis. In the following, we first discuss how a sample of synthetic problem instances was generated, and then discuss noise response experiments for all satisfiable problem instances in the sample. A detailed analysis then follows, where we show the complete search spaces for two interesting problem instances, analytically derive their Markov chains, and compare analytical results with empirical SGS search behavior.

7.1.1. Methodology for generating small synthetic Bayesian networks

Let, in a Bayesian network (BN), V be the number of root nodes and C the number of non-root nodes. It has been demonstrated that the ratio C/V is a key parameter for SAT and BN inference hardness for randomly generated problem instances [27,33,34]. For BNs, the C/V-ratio can be used to predict upper and lower bounds on the optimal maximal clique size (or treewidth) of the induced clique tree for bipartite BNs randomly generated using the BPART algorithm [27,29,33]. The BPART algorithm has these input parameters: Q—CPT type of root nodes; F—CPT type of the non-root nodes; V—the number of root nodes; C—the number of leaf nodes; S—the number of states per node; P—the number of parents per non-root node; and R—regularity of the BN's underlying graph.

The input parameters of BPART were set as follows to generate 3SAT BNs for experimentation: The CPT type of the root nodes was $Q = \texttt{uniform}$; the CPT type of the non-root nodes was $F = \texttt{or}$; the number of root nodes was $V = 5$; the number of leaf nodes was $C = 20$; the number of states per node was $S = 2$; the number of parents per leaf node was $P = 3$; and irregular BNs were created by setting $R = \textbf{false}$. This gives $C/V = 4.0$, which lies somewhat below the phase transitions of SAT [34], meaning that most generated problems would be satisfiable, but some might not be. Using these parameter settings, 100 problem instances were generated. The existence of satisfying assignments was checked by processing the BNs, with clamped leaf nodes, using the HUGIN tree clustering system, which implements the tree clustering algorithm [5,24].

The SGS system [27,30,31] with no restarts, uniform initialization in $\mathbb{I}$, and search portfolio $\mathbb{S}(p) := \{(p,\text{UN}), (1-p,\text{GM})\}$ was employed, with varying noise probability. Following standard methodology, experiments with SGS were only conducted using instances with one or more satisfying assignments.

7.1.2. Noise experiments using small synthetic Bayesian networks

Experimentation progressed in two phases: First, noise responses for all satisfiable problem instances were generated empirically by varying the noise level from $p = 0.1$ to $p = 0.9$ in increments of $\Delta p = 0.05$. The empirically observed optimal values for the noise parameter ranged from $\hat{p}^* = 0.1$ to $\hat{p}^* = 0.7$. Second, more detailed studies were performed, focusing on: (i) expected hitting time results derived analytically for a few real problem instances; (ii) real SGS search behavior for the same problem instances; and (iii) polynomial regression results based on SGS's search behavior under varying p.

The complete search spaces for the easiest and hardest problem instances in the sample, denoted β_{81} and β_8 respectively, are illustrated in Fig. 7. Let $\boldsymbol{b}^+ \in \{0,1\}^5$ range over all almost-satisfying assignments, defined as assignments with $C - 1 = 19$ satisfied clauses. Such almost-satisfying assignments are of interest because they may form local optima that trap the SIMPLESLS search process. From Fig. 7 we see that both β_{81} and β_8 have eight almost-satisfying assignments. For the easy problem instance β_{81}, Hamming distance to optimum $\boldsymbol{b}^*$ is $d(\boldsymbol{b}^+, \boldsymbol{b}^*) = 1$ or $d(\boldsymbol{b}^+, \boldsymbol{b}^*) = 2$ in all but two cases. Clearly, the three search space states with $d(\boldsymbol{b}^+, \boldsymbol{b}^*) = 1$ do not form local optima in β_{81}. For the hard problem instance β_8, on the other hard, $d(\boldsymbol{b}^+, \boldsymbol{b}^*) = 3$ or $d(\boldsymbol{b}^+, \boldsymbol{b}^*) = 4$ in all but one case and all of the almost-satisfying assignments, or "19" states, form a local optimum.

More specifically, the eight "19" states in β_{81} are all connected, and three of them have the optimal "20" state as a neighbor. Hence, these "19" states form a plateau from which "20" is reached with relative ease. In β_8, the eight "19" states are all connected also; however for this harder problem instance these states form a local optimum since none of them has "20" as a neighbor. Based on this inspection of search spaces, we expect β_{81} to be easier than β_8 for SIMPLESLS because β_{81} does not trap the search process to the same degree. More generally, this illustrates that instances can be easier because of the number or relative location of almost-optimal states or more generally local optima in the search space, which in idealized form is illustrated in trap functions.

To provide a more detailed quantitative analysis, we created two distinct Markov chains based on the two problem instances, also taking into account the behavior of SGS. These two Markov chain models, which we denote $\text{MC}(p; \beta_{81})$ and $\text{MC}(p; \beta_8)$, are derived by inspecting the complete search spaces of β_8 and β_{81} respectively. These approximate Markov chains are shown in Fig. 8. In the following table we compare $\Pr(X_{i+1} = 3 \mid X_i = 2)$ and $\Pr(X_{i+1} = 4 \mid X_i = 3)$, which lead search towards optimum, for $\text{TMC}(p; 5, 0)$, $\text{TMC}(p; 5, 4)$, $\text{MC}(p; \beta_{81})$, and $\text{MC}(p; \beta_8)$:

	$\Pr(X_{i+1} = 3 \mid X_i = 2)$	$\Pr(X_{i+1} = 4 \mid X_i = 3)$
$\text{TMC}(p; 5, 0)$—Very easy	$1.0 - 0.4p$	$1.0 - 0.6p$
$\text{MC}(p; \beta_{81})$—Easiest in sample	$0.85 - 0.25p$	$0.78 - 0.38p$
$\text{MC}(p; \beta_8)$—Hardest in sample	$0.23 + 0.37p$	$0.1 + 0.3p$
$\text{TMC}(p; 5, 4)$—Hard	$0.6p$	$0.4p$

Just by considering the signs of the coefficients in front of p in this table, we clearly see the similarity between $\text{TMC}(p; 5, 0)$ and $\text{MC}(p; \beta_{81})$ on the one hand and $\text{TMC}(p; 5, 4)$ and $\text{MC}(p; \beta_8)$ on the other. For $\text{TMC}(p; 5, 0)$ and $\text{MC}(p; \beta_{81})$, *increasing* the noise parameter p *decreases* the probabilities $\Pr(X_{i+1} = 3 \mid X_i = 2)$ and $\Pr(X_{i+1} = 4 \mid X_i = 3)$ that SIMPLESLS moves towards the optimal state $s^* = 5$. This reflects that pure or almost pure hill-climbing, with no or minimal noise p, is optimal in the underlying search spaces. For $\text{TMC}(p; 5, 4)$ and $\text{MC}(p; \beta_8)$, on the other hand, *increasing* the noise parameter p *increases* the probabilities $\Pr(X_{i+1} = 3 \mid X_i = 2)$ and $\Pr(X_{i+1} = 4 \mid X_i = 3)$ that SIMPLESLS moves towards $s^* = 5$. This shows how, in order to counter-act search being trapped in parts of the search space that do not contain optima, the noise parameter p can be increased. There is a similarity between the search traps observed in the idealized TMC models and the search traps as reflected in the Markov chains of the real problem instances β_8 and β_{81}.

With the Markov chains $\text{MC}(p; \beta_{81})$ and $\text{MC}(p; \beta_8)$ in hand, we can analytically derive expected hitting time curves, and compare them to empirical noise response curves. In Fig. 9, we show: (i) expected hitting times derived from extreme trap Markov chains (TMCs); (ii) expected hitting time curves and run times, in the form of rational

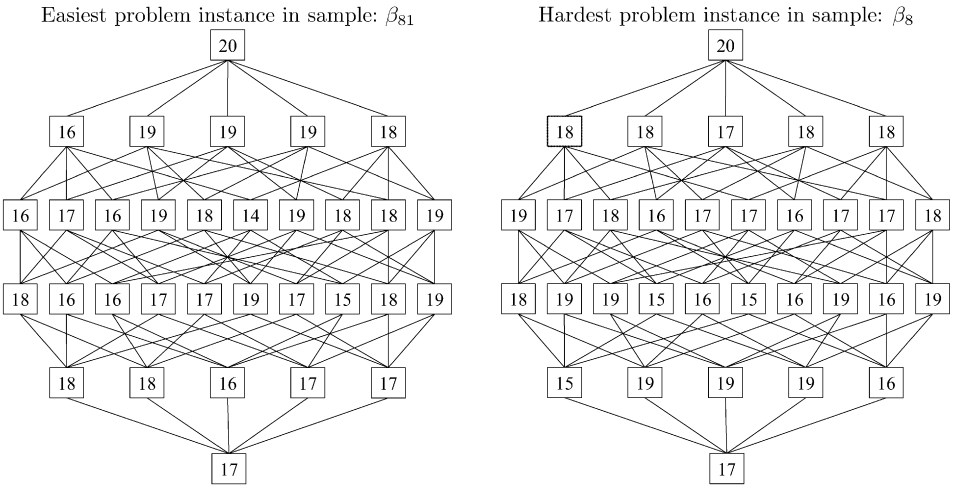

Fig. 7. The complete search spaces for two randomly generated BNs that represent instances of the satisfiability problem (3SAT) with $V = 5$ variables and $C = 20$ clauses. We show, for instances with one satisfying assignment, the easiest and the hardest problem instances from a random sample of 100 BNs. The number of satisfied clauses is shown for each state in the search space.

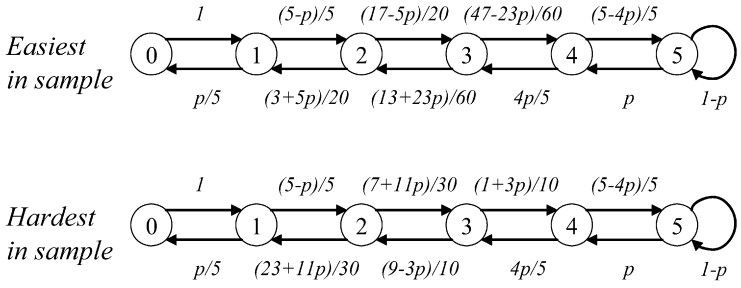

Fig. 8. Markov chains created from considering the search spaces of two problem instances β_{81} (top) and β_8 (bottom) along with the behavior of SIMPLESLS. Fig. 7 shows the underlying search spaces.

functions, derived from the Markov chains in Fig. 8; and (iii) data points reporting real SGS behavior on the same problem instances. There is a very good correspondence between analytical hitting time results in (ii) and observed SGS run times in (iii). For both (ii) and (iii), (i) provides lower bounding hitting time $h_{5,0}(p)$ and upper bounding hitting time $h_{5,4}(p)$. To our knowledge, similar displays that compare analytical and experiments results have not been reported in the literature earlier.

We learned above that closed-form expressions of expected hitting times can be found for β_{81} and β_8. Next, we determine experimentally polynomial regression approximations $\hat{P}(p)$ for SGS run times. Our polynomial approximations are shown in Table 1, and in Fig. 10 we present empirical results along with regression curves. Regression results are significant according to the R^2 values, and Fig. 10 shows very good correspondence between empirical and analytical results. The polynomial regression lines are very close except for towards the less interesting endpoints of their domain $[0.1, 0.9]$, as is typically the case.

7.2. Experiments with large synthetic Bayesian networks

In this section we systematically and simultaneously vary the hardness of problem instances as well as the SLS noise probability. We report on observed SGS search behavior, and how it can be fitted using polynomials. Here, real search results for BNs corresponding to SAT instances are reported for $V = 30$, with $C/V = 2.0$ to $C/V = 3.4$, where for each C/V-value 100 problem instances were generated and searched over. Hence, the real search behavior of SGS for 800 problem instances is summarized and analyzed in this section. To our knowledge, similar noise response experiments have not been reported in the literature earlier.

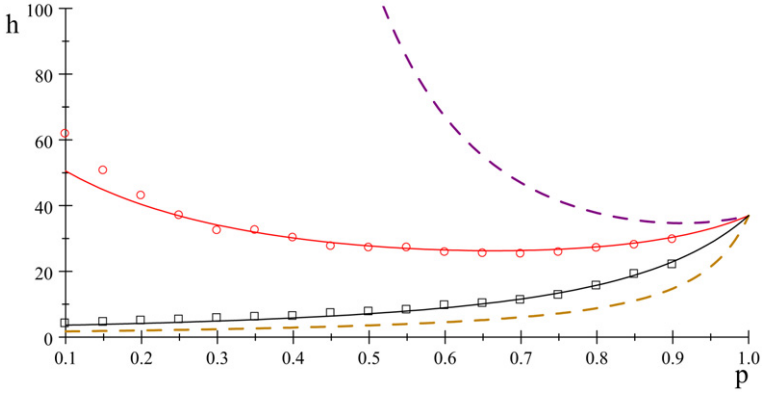

Fig. 9. The expected hitting time h as a function of noise probability p for the two extreme problem instances β_8 and β_{81} (black line for β_{81} and red line for β_8), picked from a sample of 100, along with actual SGS behavior for the same two instances (black squares for β_{81} and red circles for β_8). Expected hitting times $h_{5,0}(p)$ (dashed brown) and $h_{5,4}(p)$ (dashed purple) for the two 5-bit trap Markov chain models TMC$(p, 5, 0)$ and TMC$(p, 5, 4)$ are also shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Regression polynomials of order $k = 4$, $k = 5$, and $k = 6$ created from empirical SGS search data for the two extreme problem instances β_{81} and β_8 (Easiest and Hardest in sample respectively)

Instance	Order	Polynomial approximation $\hat{h}(p)$	R^2
β_{81}	4	$112.72p^4 - 160.74p^3 + 89.826p^2 - 13.69p + 5.0822$	0.9973
β_{81}	5	$-36.836p^5 + 209.41p^4 - 255.9p^3 + 133.1p^2 - 22.627p + 5.7406$	0.9973
β_{81}	6	$-2455.5p^6 + 7697.9p^5 - 9428.3p^4 + 5770.7p^3 - 1840.9p^2 + 294.85p - 13.635$	0.9983
β_8	4	$495.22p^4 - 1151.8p^3 + 1021.2p^2 - 418.25p + 94.197$	0.9959
β_8	5	$-682.5p^5 + 2201.5p^4 - 2724.4p^3 + 1673.8p^2 - 536.31p + 101.33$	0.9970
β_8	6	$-1161.6p^6 + 2802.4p^5 - 1878.8p^4 - 372.05p^3 + 982.67p^2 - 440.52p + 96.57$	0.9971

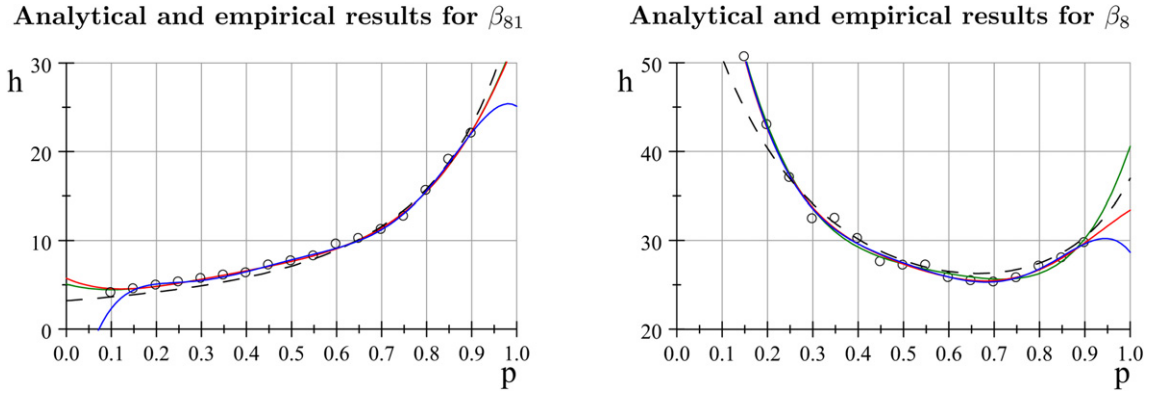

Fig. 10. Comparison of (i) analytical hitting times derived from Markov chains MC$(p; \beta)$ (black dashed line); (ii) empirical data points generated from real SGS search behavior (black circles); and (iii) polynomial regression curves estimated from the empirical data points. The polynomial regressions lines are of order k, with $k = 4$ (green line), $k = 5$ (red line), and $k = 6$ (blue line). *Left*: Results for problem instance β_{81}; *Right*: Results for problem instance β_8. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.2.1. Methodology for generating large synthetic Bayesian networks

For the experiments reported here, where SAT-like BNs were generated, we again used the BPART approach discussed in Section 7.1. Here, BPART's input parameters were set as follows, generating larger problem instances compared to those in Section 7.1: The CPT type of the root nodes was $Q = \texttt{uniform}$; the CPT type of the non-root nodes was $F = \texttt{or}$; the number of root nodes was $V = 30$; the number of states per node was $S = 2$; the number of

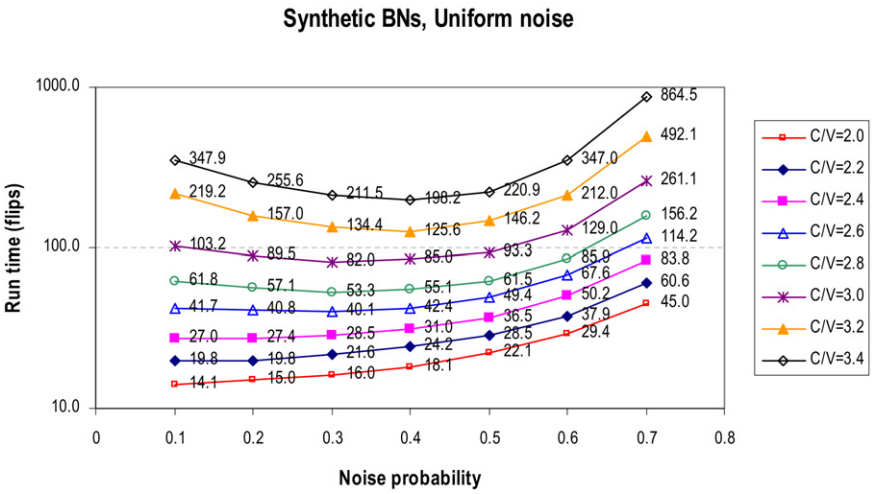

Fig. 11. Experimental results for synthetic BNs with C/V-ratios ranging from $C/V = 2.0$ to $C/V = 3.4$. Sample means for the run times (number of flips) is shown as a function of SLS noise probability p. Note that all these piecewise linear functions are convex.

parents per leaf node was $P = 3$; and irregular BNs were created by setting $R = \textbf{false}$. We varied the number of leaf nodes while keeping $V = 30$ constant, giving C/V-ratios varying from $C/V = 2.0$ to $C/V = 3.4$. This is in the satisfiable region of SAT [34] where solutions exist with very high probability. The existence of solutions was also checked by processing the BNs using the HUGIN tree clustering system, which implements the tree clustering algorithm [5,24].

7.2.2. Noise experiments using large synthetic Bayesian networks

The purpose of the second set of BN experiments was to investigate in more detail the combined effect of varying noise p and varying the hardness of larger, more realistically sized problem instances. For the purpose of these experiments we measured instance hardness by means of the C/V-ratio. The stochastic local search algorithm SGS [27,30] was employed, using a restart parameter value MAX-FLIPS optimized for the C/V-ratio and p at hand. The search portfolio $\mathbb{S}_N(p) := \{(p, \text{UN}), (1-p, \text{GM})\}$ was used, with noise probability varying from $p = 0.1$ to $p = 0.7$ in increments of $\Delta p = 0.1$. Initialization uniformly at random was used in $\mathbb{I}$.

Fig. 11 summarizes the experimental results in the form of noise response curves. Each BN was searched 100 times by SGS, and since for each C/V-ratio 100 BNs were generated, each data point in Fig. 11 represents 10,000 successful searches by SGS. There are eight different C/V-ratios, ranging from $C/V = 2.0$ to $C/V = 3.4$. The noise probability p was varied as reflected on the x-axis of Fig. 11. The y-axis measures the mean number of flips until an optimum $\boldsymbol{b}^*$ was found. For the relatively easy $C/V = 2.0$ BNs, the sample mean is monotonically increasing with the noise p, and the experimentally determined global minimum run time $\hat{r}^*$ was found at the minimal noise level investigated, so $\hat{p}^* = 0.1$. For the hardest problem instances, with $C/V = 3.4$, the sample mean is first monotonically decreasing, then increasing, as a function of p. Here, the experimentally determined optimal noise level is $\hat{p}^* = 0.4$. For $C/V = 2.6$, the sample average is close to constant from $p = 0.1$. to $p = 0.4$; it then increases monotonically. Results for the other intermediate C/V-ratios, $2.0 < C/V < 2.6$ and $2.6 < C/V < 3.4$, are similar to the $C/V = 2.0$ case or the $C/V = 3.4$ case respectively.

As argued in Section 6.2, similar analysis approaches can be used for mixtures, one of which has been sampled here for each C/V-ratio, as for individual problem instances. The noise response curves in Fig. 11 are similar to those reported for TMCs in Fig. 6 and for smaller problem instances in Fig. 10, illustrating how our results carry over from individual problem instances to mixtures of problem instances as predicted by our analysis in Section 6. Polynomial approximation results are shown in Table 2 and in Fig. 12. For each C/V-ratio we give three alternatives, namely polynomials of orders 4, 5, and 6. Polynomials of smaller order did not give good results. From $C/V = 2.0$ to $C/V = 2.8$, all polynomials give rather similar results. From $C/V = 3.0$ to $C/V = 3.4$, the polynomials of order $k = 4$ provide visibly poorer results as can be seen in Fig. 12 and from R^2 in Table 2.

Qualitatively, these results are consistent with the analysis in Sections 4, 5, and 6 as well as earlier experiments [16,17,26,45]. A key point here is that increasing average problem instance hardness, as controlled by the C/V-ratio,

Table 2
Polynomial approximations for SGS sample average run times (flips), as a function of noise level p, for varying C/V-ratios. The polynomials were determined from experimental data using non-linear regression. For each C/V-ratio three alternatives are given, namely polynomials of order 4, 5, and 6

C/V	Order	Polynomial approximation, $\hat{r}(p)$	R^2
2.0	4	$y = 512.9p^4 - 530.2p^3 + 224.2p^2 - 30.88p + 15.43$	0.9998
2.0	5	$y = 1500p^5 - 2486p^4 + 1694p^3 - 525.6p^2 + 80.83p + 9.771$	1.0
2.0	6	$y = 2798p^6 - 5216p^5 + 3861p^4 - 1298p^3 + 205.9p^2 - 5.501p + 13.51$	1.0
2.2	4	$y = 1255p^4 - 1568p^3 + 740.9p^2 - 133.1p + 27.17$	0.9999
2.2	5	$y = 1112p^5 - 968.7p^4 + 81.31p^3 + 184.9p^2 - 50.21p + 22.97$	1.0
2.2	6	$y = 1285p^6 - 1973p^5 + 1947p^4 - 1293p^3 + 520.9p^2 - 89.87p + 24.69$	1.0
2.4	4	$y = 1554p^4 - 1766p^3 + 758.2p^2 - 127.2p + 33.84$	0.9999
2.4	5	$y = 2249p^5 - 2944p^4 + 1570p^3 - 366.4p^2 + 40.40p + 25.36$	1.0
2.4	6	$y = 833.3p^6 + 249.1p^5 - 1054p^4 + 679.5p^3 - 148.6p^2 + 14.69p + 26.48$	1.0
2.6	4	$y = 1998p^4 - 2186p^3 + 919.9p^2 - 168.8p + 51.50$	0.9996
2.6	5	$y = 4797p^5 - 7596p^4 + 4930p^3 - 1479p^2 + 188.6p + 33.41$	1.0
2.6	6	$y = 1246p^6 + 1807p^5 - 4770p^4 + 3597p^3 - 1153p^2 + 150.1p + 35.08$	1.0
2.8	4	$y = 3732p^4 - 4380p^3 + 1941p^2 - 392.6p + 85.94$	0.9994
2.8	5	$y = 8302p^5 - 12872p^4 + 7935p^3 - 2210p^2 + 225.9p + 54.63$	1.0
2.8	6	$y = -27081p^6 + 73297p^5 - 74297p^4 + 36887p^3 - 9289p^2 + 1061p + 18.42$	1.0
3.0	4	$y = 9418p^4 - 12094p^3 + 5708p^2 - 1179p + 175.8$	0.9984
3.0	5	$y = 23729p^5 - 38041p^4 + 23104p^3 - 6157p^2 + 588.9p + 86.35$	1.0
3.0	6	$y = -22734p^6 + 78291p^5 - 89606p^4 + 47409p^3 - 12100p^2 + 1290p + 55.95$	1.0
3.2	4	$y = 22176p^4 - 29756p^3 + 14929p^2 - 3402p + 438.8$	0.9981
3.2	5	$y = 46501p^5 - 70825p^4 + 39220p^3 - 8321p^2 + 61.77p + 263.4$	0.9996
3.2	6	$y = 270595p^6 - 602928p^5 + 542934p^4 - 250071p^3 + 62412p^2 - 8286p + 625.3$	1.0
3.4	4	$y = 39431p^4 - 51539p^3 + 24947p^2 - 5500p + 698.2$	0.9985
3.4	5	$y = 82600p^5 - 125769p^4 + 70985p^3 - 16353p^2 + 653.6p + 386.7$	1.0
3.4	6	$y = 165545p^6 - 314708p^5 + 249717p^4 - 105998p^3 + 26920p^2 - 4454p + 608.1$	1.0

corresponds to moving the slope-change state z in a trap Markov chain towards the optimum state (see Fig. 5). In other words, problem instances that are easy on average (corresponding to TMCs with slope-change states z close to $00\ldots0$) should be solved by a greedier SLS algorithm than problem instances that are hard on average (corresponding to TMCs with slope-change state z close to $11\ldots1$). We also note that the piecewise linear curves are convex for all C/V-ratios. What is novel here, compared to earlier experiments that we know of, is the different pattern for the easy $C/V = 2.0$ case versus the hard $C/V = 3.4$ case, our extensive regression analysis using polynomials, and the clear display of convexity across a wide range of C/V.

7.2.3. Optimal noise level experiments using large synthetic Bayesian networks

From earlier analysis and experiments, we know that the noise probability has a significant impact on the number of search steps needed to reach an optimum. How does the optimal noise probability p^* change as the C/V-ratio changes? This is the research question investigated in this section.

We wanted to minimize the number of flips as a function of the C/V-ratio. We first found approximate optima based on the regression polynomials reported in Table 2. This was done by taking derivatives $\frac{d}{dp}\hat{r}(p) = \hat{r}'(p)$ and then solving the equation $\hat{r}'(p) = 0$ with respect to p in order to compute optimized noise $\hat{p}^*$. The results are shown in Table 3. For each value of C/V, one empirical optimum and three regression polynomials, of different orders, are presented.

In a few cases, $\hat{r}'(p) = 0$ had multiple candidate solutions but it was obvious which one was most reasonable to choose as reflected in the table. Further, in two cases there were only complex solutions and in one case ($C/V = 2.4$ and polynomial of order 6) the solution was less than zero so it was set to zero. As discussed in Section 6.3, even-degree polynomials are not guaranteed to have real roots. Given this fact, we have still opted to include in Table 3 the cases where $r'(p)$ has degree 4. Even though there is no guarantee for a real root, in most cases (specifically, in all cases except two in Table 3 containing $r'(p)$) a real root was found and thus additional insight is provided.

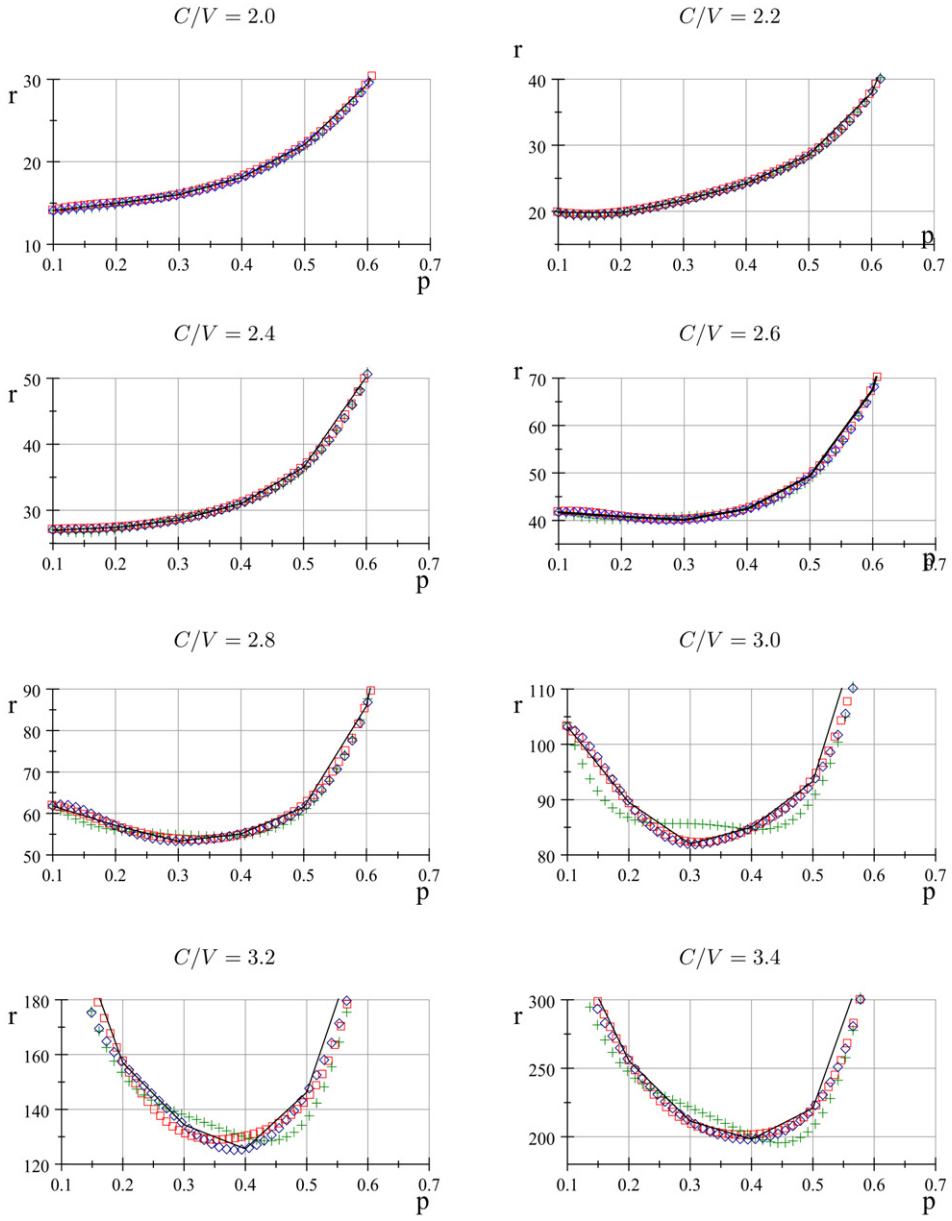

Fig. 12. Polynomial approximations for SGS average run times (flips), as a function of noise level p, for BNs with varying C/V-ratios. For each C/V-ratio, polynomial regression lines of varying order k are presented, with $k = 4$ (line indicated by green crosses), $k = 5$ (line indicated by red squares), and $k = 6$ (line indicated by blue diamonds). The means of the measured data, connected with straight lines (black), are also included. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Clearly, there is a good correspondence between the polynomials and the empirical results, especially for the polynomials of order $k = 6$ for the higher C/V-values. Considering Fig. 12, and excluding $C/V = 2.0$ and $C/V = 2.2$, one can see that $k = 5$ and $k = 6$ give quite similar results for $\hat{p}^*$.

7.3. Experiments with application Bayesian networks

It is of great interest to consider also more advanced initialization and noise strategies on problem instances from applications. Here, we report empirical SGS results for application BNs, many of which are taken from Friedman's Bayesian Network Repository at http://www.cs.huji.ac.il/labs/compbio/Repository/. The application BNs investigated

Table 3
Noise optimization based on derivatives of the polynomial approximations for SGS sample average run times (number of flips), as a function of noise level $p = x$, for varying C/V-ratios. For each C/V-ratio three alternatives are given, namely a polynomial of order 4, 5, and 6. In addition, the optimal noise levels from the experiments are shown

C/V	Order	Derivative of polynomial approximation, $\hat{r}'(p)$	Optimized noise $\hat{p}^*$
2.0	4	$2051.6p^3 - 1591p^2 + 448.4p - 30.88$	9.95×10^{-2}
2.0	5	$7498p^4 - 9945p^3 + 5083p^2 - 1051.2p + 80.83$	Complex
2.0	6	$7712p^5 - 9864p^4 + 7786p^3 - 3878p^2 + 1042p - 89.87$	1.55×10^{-2}
2.0	N/A	Empirical	0.1
2.2	4	$5020p^3 - 4704p^2 + 1482p - 133.1$	0.149
2.2	5	$5560p^4 - 3875p^3 + 243.9p^2 + 369.8p - 50.21$	0.148
2.2	6	$7712p^5 - 9864p^4 + 7786p^3 - 3878p^2 + 1042p - 89.87$	0.147
2.2	N/A	Empirical	0.1 and 0.2
2.4	4	$6215p^3 - 5297p^2 + 1516p - 127.2$	0.144 24
2.4	5	$11246p^4 - 11778p^3 + 4711p^2 - 732.8p + 40.40$	Complex
2.4	6	$5000p^5 + 1245p^4 - 4217p^3 + 2038p^2 - 297.1p + 14.69$	0
2.4	N/A	Empirical	0.1
2.6	4	$7994p^3 - 6559p^2 + 1840p - 168.8$	0.196
2.6	5	$23986p^4 - 30383p^3 + 14789p^2 - 2957p + 188.6$	0.279
2.6	6	$7476p^5 + 9034p^4 - 19079p^3 + 10792p^2 - 2306p + 150.1$	0.280
2.6	N/A	Empirical	0.3
2.8	4	$14929p^3 - 13140p^2 + 3882p - 392.6$	0.366
2.8	5	$41512p^4 - 51488p^3 + 23805p^2 - 4420p + 225.9$	0.327
2.8	6	$-162486p^5 + 366485p^4 - 297188p^3 + 110661p^2 - 18578p + 1061$	0.305
2.8	N/A	Empirical	0.3
3.0	4	$37671p^3 - 36282p^2 + 11416p - 1179$	0.412
3.0	5	$118645p^4 - 152164p^3 + 69312p^2 - 12314p + 588.9$	0.315
3.0	6	$-136404p^5 + 391455p^4 - 358424p^3 + 142227p^2 - 24200p + 1290$	0.307
3.0	N/A	Empirical	0.3
3.2	4	$88704p^3 - 89268p^2 + 29858p - 3402$	0.437
3.2	5	$232505p^4 - 283300p^3 + 117660p^2 - 16643p + 61.77$	0.355
3.2	6	$1623570p^5 - 3014640p^4 + 2171736p^3 - 750213p^2 + 124824p - 8286$	0.384
3.2	N/A	Empirical	0.4
3.4	4	$157724p^3 - 154617p^2 + 49894p - 5500$	0.448
3.4	5	$413000p^4 - 503076p^3 + 212955p^2 - 32706p + 653.6$	0.389
3.4	6	$993270p^5 - 1573540p^4 + 998868p^3 - 317994p^2 + 3840p - 4454$	0.394
3.4	N/A	Empirical	0.4

are Mildew, Munin1, Pir3, and Water. The Mildew BN is for determining the amount of fungicides to use to counteract mildew attacks on wheat. The Munin1 network is a medical BN from the field of electromyography [2]. The Pir3 BN is for information filtering for the purpose of battlefield situation awareness [21,32]. The Water BN models the biological processes of water purification.

The purpose of these experiments was to investigate the effect of varying p, and also investigate SLS strategies that go beyond uniform random noise and initialization uniformly at random. More specifically, we compared the following two search algorithm portfolios $\mathbb{S}_G(p)$ and $\mathbb{S}_U(p)$.

Definition 46 *(Guided noise).* The guided noise portfolio $\mathbb{S}_G(p)$ is a function of noise probability p and is defined as

$$\mathbb{S}_G(p) := \left\{ \left(\frac{p}{2}, \mathrm{BS}\right), \left(\frac{p}{2}, \mathrm{GS}\right), \left(\frac{1-p}{2}, \mathrm{BM}\right), \left(\frac{1-p}{2}, \mathrm{GM}\right) \right\}.$$

Definition 47 *(Uniform noise).* The uniform noise portfolio $\mathbb{S}_U(p)$ is a function of noise probability p and is defined as

$$\mathbb{S}_U(p) := \left\{ \left(p, \mathrm{NU}\right), \left(\frac{1-p}{2}, \mathrm{BM}\right), \left(\frac{1-p}{2}, \mathrm{GM}\right) \right\}.$$

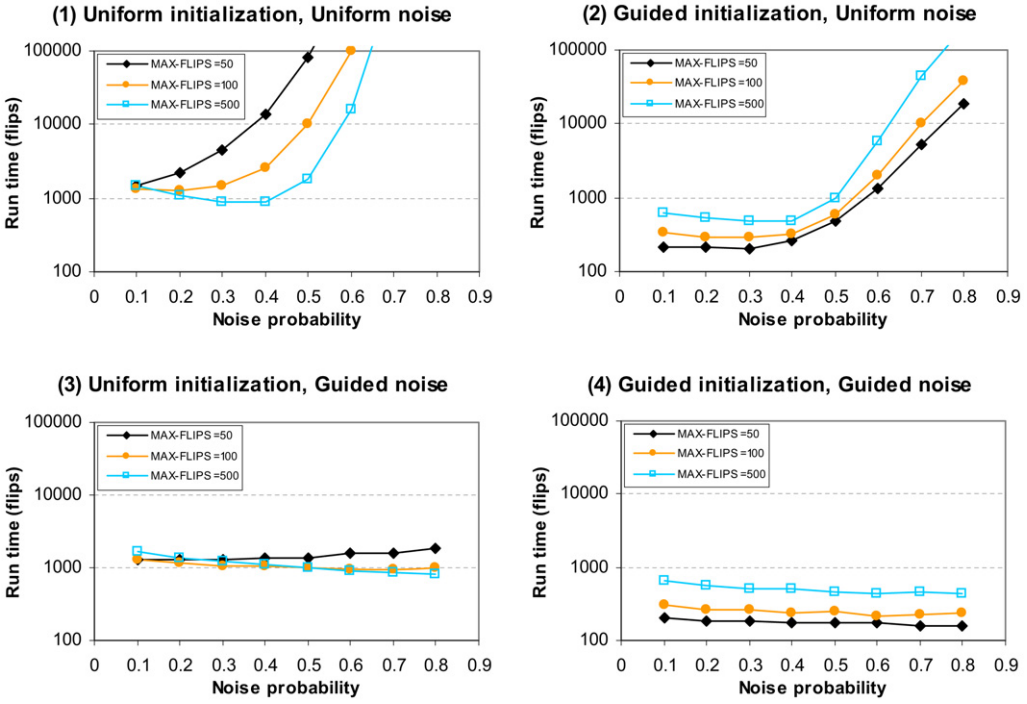

Fig. 13. Empirical results for the Water BN under four different conditions (1), (2), (3), and (4). In all cases, the noise probability p, varying from $p = 0.1$ to $p = 0.8$, is shown on the x-axis. The mean run time (measured in flips) is displayed using a logarithmic scale on the y-axis. Each point represents the sample mean of 1000 runs. For each condition, three different values of MAX-FLIPS were investigated as shown.

In experiments, we used either $\mathbb{S}_G(p)$ or $\mathbb{S}_U(p)$ as the search algorithm portfolio of SGS. We note that all experiments using $\mathbb{S}_G(p)$ do not use uniform random noise, but more advanced approaches to noisy search. In addition, we also compared $\mathbb{I}_U$ (initialization uniformly at random) versus $\mathbb{I}_G$ (guided initialization), and also varied MAX-FLIPS.

7.3.1. Varying noise, initialization, and restart point: One application BN

The purpose of our first set of experiments was to establish the effect, if any, of using different variants of SGS for one particular BN, Water. Specifically, we varied both $\mathbb{I}$ and $\mathbb{S}$. We studied the impact of varying the noise p, from $p = 0.1$ to $p = 0.8$, under conditions of different noise and initialization portfolios as well as different values of the MAX-FLIPS restart parameter. The two orthogonal dimensions investigated were:

- Initialization portfolio $\mathbb{I}$: Uniform initialization $\mathbb{I}_U$ versus guided initialization $\mathbb{I}_G$. Here, uniform means initialization uniformly at random while guided means the use of forward simulation [13].
- Search portfolio $\mathbb{S}$: Uniform noise $\mathbb{S}_U(p)$ versus guided noise $\mathbb{S}_G(p)$.

The four conditions investigated were: (1)—$\mathbb{I}_U$ and $\mathbb{S}_U(p)$; (2)—$\mathbb{I}_G$ and $\mathbb{S}_U(p)$; (3)—$\mathbb{I}_U$ and $\mathbb{S}_G(p)$; and (4)—$\mathbb{I}_G$ and $\mathbb{S}_G(p)$. Fig. 13 presents the results for these four different conditions in the form of sample means and piecewise linear approximations for $\hat{r}(p)$. In all cases, $\hat{r}(p)$ is convex or close to convex. There are quite different run time responses to changes in noise, depending on the initialization operator used in $\mathbb{I}$, the noise operator used in $\mathbb{S}$, and the value of the MAX-FLIPS parameter. Clearly, condition (4) has the shortest run time and also the overall average run time is shortest; the impact of varying p on $\hat{r}(p)$ is small. Condition (1) has the longest run times; here the impact of varying p is large. Overall, perhaps the greatest impact on $\hat{r}(p)$ is due to whether uniform noise $\mathbb{S}_U(p)$ (top row, (1) and (2)) or guided noise $\mathbb{S}_G(p)$ is used (bottom row, (3) and (4)). For uniform noise, the impact of varying noise p is dramatic and $\hat{p}^* \leqslant 0.4$ for a given MAX-FLIPS level in all cases. For guided noise, on the other hand, the effect of varying p on $\hat{r}(p)$ is rather minimal. Further, and perhaps surprisingly, $\hat{p}^*$ is in (3) and (4) quite large in all cases but one, namely MAX-FLIPS = 50 in (3). Overall, the non-trivial interactions between BN, SLS algorithm parameters,

and noise p are illustrated here. However, the piecewise linear approximation are convex or close to convex in all cases, thus supporting our analytical results.

7.3.2. Varying noise and restart point: All application BNs

The purpose of the second set of application BN experiments was to establish the effect, if any, of varying the noise and the value of MAX-FLIPS for different applications BNs. Two variants of SGS, namely SGS with uniform noise $\mathbb{S}_U(p)$ and SGS with guided noise $\mathbb{S}_G(p)$, were used. Each SGS variant was tested with four different BNs—Mildew, Munin1, Pir3, and Water—giving a total of eight conditions as shown in Fig. 14. Further, a BN-specific optimal initialization algorithm—either forward simulation [13] or a randomized variant of the Viterbi algorithm [31, 49]—was used in $\mathbb{I}$ for each application BN. Forward simulation was used for Munin1 and Water. Mildew and Pir3 were respectively initialized using the forward and backward variants of the randomized Viterbi algorithm.

For each condition, noise was varied from $p = 0.1$ to $p = 0.8$, and different values for the MAX-FLIPS restart parameter were also used. Results, in the form of sample means and piecewise linear approximations for $\hat{r}(p)$, are reported in Fig. 14. Each data point represents the mean for 1000 runs. In general, guided noise $\mathbb{S}_G(p)$ performed better than uniform noise $\mathbb{S}_U(p)$, however the effect on estimated run time $\hat{r}(p)$ of increasing noise varied dramatically between problem instances. For Pir3, initialization was strong and increasing noise only hurt performance. For Munin1, there was a marked difference between uniform noise and guided noise. Uniform noise hurt performance, while guided noise had, except for MAX-FLIPS $= 10$, minor impact on run time. Water and Mildew had somewhat similar performance. Low levels of uniform noise were helpful for some values of MAX-FLIPS, while for other values of MAX-FLIPS increasing noise from $p = 0.1$ did not help. For Mildew, rather high levels of guided noise were optimal for all MAX-FLIPS levels investigated. This was also the case for Water, except for MAX-FLIPS $= 10$. Except for some of the curves for Pir3, these piecewise linear curves are clearly convex or close to convex, thus supporting our analytical results.

8. Conclusion and future work

The use of randomization, in the form of noisy initialization and noisy local search steps, has empirically been shown to have a dramatic and positive impact on the performance of local search [7,8,14,17,18,26,27,30,45,46]. Consequently, stochastic local search (SLS) algorithms are currently strong performers in several areas of automated reasoning, including the problems of satisfiability (SAT) in propositional logic [11,18,45,46] and computing MPE and MAP in Bayesian networks [22,27,30,36,37]. Previous research on stochastic local search has been predominantly experimental, and there is a need for theoretical foundations [16]. We have in this article, based on discrete Markov chain models derived from a simple but general SLS algorithm called SIMPLESLS, developed a theoretical foundation for the role of noise in stochastic local search. Analytically, we used hitting time analysis in Markov chain models to obtain our results. Curves for expected hitting times are the analytical counterpart to empirical noise response curves often reported in the literature [7,8,14,16,26]. Our analysis shows that expected hitting times, for individual instances as well as for mixtures of instances, are rational functions with noise p as the independent variable. We also emphasize the use of polynomials and convex functions. Analytically, we have found that the impact of noise is quite context-dependent, and the shape of the expected hitting time curve depends strongly on the problem instance at hand.

In order to improve our understanding of the interaction between noise and the presence of local maxima, a simple class of trap functions based on deceptive functions [6] with these characteristics was introduced: (i) one global maximum; (ii) one local (deceptive) maximum at maximal distance from the global maximum; and (iii) a slope-change location controlling the size of the region from which greedy search only is sufficient to reach the global maximum. Obviously, there are other crucial issues involved in SLS applied to NP-hard problems. Our model, from which we derive trap Markov chains and then expected hitting times, highlights the problem of how local maxima trap the search process, and how the careful application of noise helps in escaping such traps. Trap functions are closely related to search space traps—portions of the search space that are attractive to SLS but do not contain solutions [11,15].

Our results also include experiments. In this area, we used the stochastic local search algorithm SGS (stochastic greedy search) for computing MPEs in Bayesian networks [27,30]. Using SGS, we experimented with synthetic Bayesian network of varying difficulty, measured in terms of C/V-ratio [33]. In these experiments, we illustrated that the Markov chain models are relevant to real problem instances. For instance, we found good correspondence

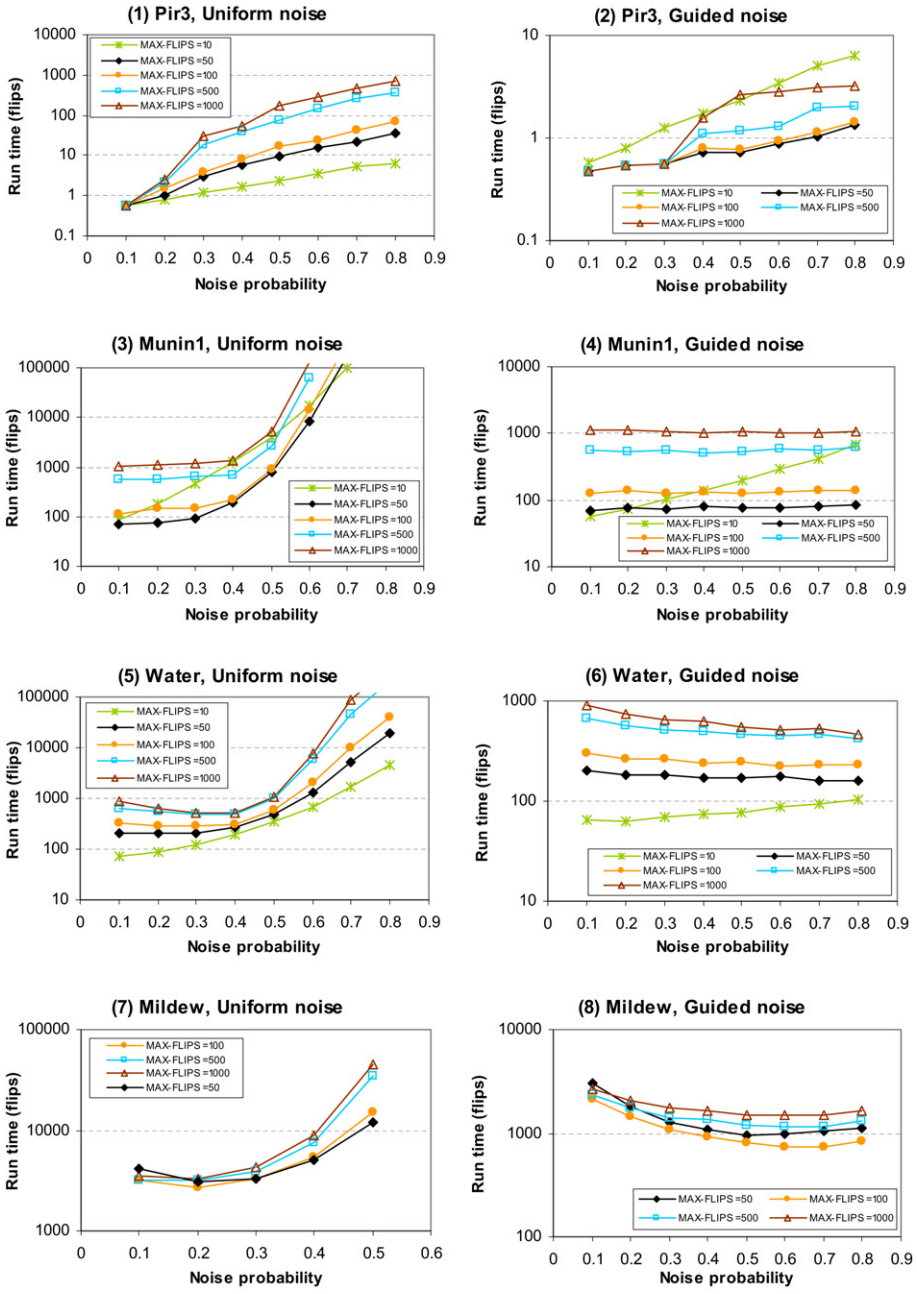

Fig. 14. Empirical results for the BNs Pir3, Munin1, Water, and Mildew under different experimental conditions. In all cases, the noise probability p, varying from $p = 0.1$ to $p = 0.8$, is shown on the x-axis. The mean run time (measured in flips) is displayed using a logarithmic scale on the y-axis. Each point represents the sample mean of 1000 runs. Two different noise mechanisms (Uniform and Guided) and five different values of MAX-FLIPS (from MAX-FLIPS = 10 to MAX-FLIPS = 1000) were used. In most cases, the value of the noise probability has a significant impact on SLS run time.

between expected hitting time results derived analytically from real problem instances, SGS's behavior on the same problem instances, and polynomial regression results. These noise response curves were, as expected, similar to but not as extreme as the bounding hitting times for the two extreme trap Markov chains defined over the same search space. We also sampled mixtures of problem instances, as defined by C/V, and found that SGS generated noise response curves consistent with our hitting time analysis. Here, we also performed extensive polynomial approximation

experiments. Finally, experiments with SGS using application Bayesian networks were also performed. Here, other input parameters of SGS—including restarts, initialization algorithms, and noise algorithms—were varied in addition to the level of noise. Results consistent with our analysis were in general found.

We conclude by outlining a few areas for future work. First, a natural extension of this research would be to analyze and optimize the joint effect of multiple SLS parameters, taking into account varying distributions of problem instance inputs (including Bayesian networks), perhaps using convex optimization and response surface techniques. Second, given the benefit of guided noise observed in experiments, it would be fruitful to study such approaches analytically. Third, this work provides a framework for improved analysis of other algorithms that can be formalized by means of Markov chains, for instance genetic and evolutionary algorithms. Fourth, the optimal noise level might vary for different sub-problems of one problem instance as well as between different problem instances and problem instance distributions. Following this line of reasoning, it would pay off to adapt the noise level for one problem instance, within a try or between different tries. While results on adaptive noise already exist [14,26], we hope that the analysis provided here will inspire further research in this area.

Acknowledgements

This material is based upon work supported by NASA under award NCC2-1426. The anonymous reviewers are acknowledged for their comments, which helped improve the article.

References

[1] A.M. Abdelbar, S.M. Hedetnieme, Approximating MAPs for belief networks is NP-hard and other theorems, Artificial Intelligence 102 (1998) 21–38.
[2] S. Andreassen, M. Woldbye, B. Falck, S.K. Andersen, MUNIN—A causal probabilistic network for interpretation of electromyographic findings. In: Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milan, Italy, August 1987, pp. 366–372.
[3] E. Cantu-Paz, Markov chain models of parallel genetic algorithms, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 216–226.
[4] F.G. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artificial Intelligence 42 (1990) 393–405.
[5] A.P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, Statistics and Computing 2 (1992) 25–36.
[6] K. Deb, D.E. Goldberg, Analyzing deception in trap functions, in: D. Whitley (Ed.), Foundations of Genetic Algorithms II, Morgan Kaufmann, San Mateo, CA, 1993, pp. 93–108.
[7] A. Fukunaga, G. Rabideau, S. Chien, Robust local search for spacecraft operations using adaptive noise, in: Proceedings of the 4th International Workshop on Planning and Scheduling for Space (IWPSS-04), Darmstadt, Germany, 2004.
[8] A. Gerevini, A. Saetti, I. Serina, An empirical analysis of some heuristic features for local search in LPG, in: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), Whistler, British Columbia, Canada, 2004, pp. 171–180.
[9] D.E. Goldberg, P. Segrest, Finite Markov chain analysis of genetic algorithms, in: J.J. Grefenstette (Ed.), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Erlbaum, Hillsdale, NJ, 1987, pp. 1–8.
[10] C.P. Gomes, B. Selman, H. Kautz, Boosting combinatorial search through randomization, in: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI, 1998, pp. 431–437.
[11] P.W. Gu, J. Purdom, J. Franco, B.W. Wah, Algorithms for the satisfiability SAT problem: A survey, in: Satisfiability Problem: Theory and Applications, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1997, pp. 19–152.
[12] G. Harik, E. Cantu-Paz, D.E. Goldberg, B.L. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, in: Proceedings of the IEEE Conference on Evolutionary Computation, Indianapolis, IN, 1997, pp. 7–12.
[13] M. Henrion, Propagating uncertainty in Bayesian networks by probabilistic logic sampling, in: Uncertainty in Artificial Intelligence 2, Elsevier, Amsterdam, 1988, pp. 149–163.
[14] H.H. Hoos, An adaptive noise mechanism for WalkSAT, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Edmonton, Alberta, Canada, 2002, pp. 655–660.
[15] H.H. Hoos, A mixture-model for the behaviour of SLS algorithms for SAT, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02), Edmonton, Alberta, Canada, 2002, pp. 661–667.
[16] H.H. Hoos, T. Stützle, Towards a characterisation of the behaviour of stochastic local search algorithms for SAT, Artificial Intelligence 112 (1–2) (1999) 213–232.
[17] H.H. Hoos, T. Stützle, Local search algorithms for SAT: An empirical evaluation, Journal of Automated Reasoning 24 (4) (2000) 421–481.
[18] H.H. Hoos, T. Stützle, Stochastic Local Search: Foundations and Applications, Morgan Kaufmann, San Francisco, CA, 2005.
[19] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, D. Chickering, A Bayesian approach to tackling hard computational problems, in: Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01), Seattle, WA, 2001, pp. 235–244.
[20] F. Hutter, H.H. Hoos, T. Stützle, Efficient stochastic local search for MPE solving, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, 2005, pp. 169–174.

[21] P. Jones, C. Hayes, D. Wilkins, R. Bargar, J. Sniezek, P. Asaro, O.J. Mengshoel, D. Kessler, M. Lucenti, I. Choi, N. Tu, J. Schlabach, CoRAVEN: Modeling and design of a multimedia intelligent infrastructure for collaborative intelligence analysis, in: Proceedings of the International Conference on Systems, Man, and Cybernetics, San Diego, CA, October 1998, pp. 914–919.
[22] K. Kask, R. Dechter, Stochastic local search for Bayesian networks, in: Proceedings Seventh International Workshop on Artificial Intelligence and Statistics, Fort Lauderdale, FL, January 1999, Morgan Kaufmann, 1999.
[23] V.G. Kulkarni, Modeling, Analysis, Design, and Control of Stochastic Systems, Springer, New York, 2005.
[24] S. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems (with discussion), Journal of the Royal Statistical Society series B 50 (2) (1988) 157–224.
[25] D.J.C. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, Cambridge, UK, 2002.
[26] D. McAllester, B. Selman, H. Kautz, Evidence for invariants in local search, in: Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97), Providence, RI, 1997, pp. 321–326.
[27] O.J. Mengshoel, Efficient Bayesian network inference: Genetic algorithms, stochastic local search, and abstraction, PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, April 1999.
[28] O.J. Mengshoel, Designing resource-bounded reasoners using Bayesian networks: System health monitoring and diagnosis, in: Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07), Nashville, TN, 2007, pp. 330–337.
[29] O.J. Mengshoel, Macroscopic models of clique tree growth for Bayesian networks, in: Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07), Vancouver, British Columbia, 2007, pp. 1256–1262.
[30] O.J. Mengshoel, D. Roth, D.C. Wilkins, Stochastic greedy search: Computing the most probable explanation in Bayesian networks, Technical Report UIUCDCS-R-2000-2150, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, February 2000.
[31] O.J. Mengshoel, D. Roth, D.C. Wilkins, Initialization and restart in stochastic local search: Computing a most probable explanation in Bayesian networks, IEEE Transactions on Knowledge and Data Engineering (2007), submitted for publication.
[32] O.J. Mengshoel, D.C. Wilkins, Raven: Bayesian networks for human-computer intelligent interaction, in: M.S. Vassiliou, T.S. Huang (Eds.), Computer Science Handbook for Displays, Rockwell Scientific Company, 2001, pp. 209–219.
[33] O.J. Mengshoel, D.C. Wilkins, D. Roth, Controlled generation of hard and easy Bayesian networks: Impact on maximal clique tree in tree clustering, Artificial Intelligence 170 (16–17) (2006) 1137–1174.
[34] D. Mitchell, B. Selman, H.J. Levesque, Hard and easy distributions of SAT problems, in: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, CA, 1992, pp. 459–465.
[35] J. Park, Using weighted MAX-SAT engines to solve MPE, in: Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-04), Edmonton, Alberta, Canada, 2004, pp. 682–687.
[36] J.D. Park, A. Darwiche, Approximating MAP using local search, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01), Seattle, WA, 2001, pp. 403–410.
[37] J.D. Park, A. Darwiche, Complexity results and approximation strategies for MAP explanations, Journal of Artificial Intelligence Research (JAIR) 21 (2004) 101–133.
[38] A.J. Parkes, J.P. Walser, Tuning local search for satisfiability testing, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, OR, 1996, pp. 356–362.
[39] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[40] D. Roth, On the hardness of approximate reasoning, Artificial Intelligence 82 (1996) 273–302.
[41] Y. Ruan, E. Horvitz, H. Kautz, Restart policies with dependence among runs: A dynamic programming approach, in: Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming, Ithaca, NY, 2002, pp. 573–586.
[42] Y. Ruan, E. Horvitz, H. Kautz, Hardness-aware restart policies, in: IJCAI-03 Workshop on Stochastic Search Algorithms, Acapulco, Mexico, 2003.
[43] D. Schuurmans, F. Southey, Local search characteristics of incomplete SAT procedures, Artificial Intelligence 132 (2) (2001) 121–150.
[44] B. Selman, H. Kautz, Domain-independent extensions to GSAT: Solving large structured satisfiability problems, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France, 1993, pp. 290–295.
[45] B. Selman, H.A. Kautz, B. Cohen, Noise strategies for improving local search, in: Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, WA, 1994, pp. 337–343.
[46] B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, CA, 1992, pp. 440–446.
[47] E. Shimony, Finding MAPs for belief networks is NP-hard, Artificial Intelligence 68 (1994) 399–410.
[48] J. Suzuki, A Markov chain analysis on a genetic algorithm, in: S. Forrest (Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA, 1993, pp. 146–153.
[49] A.J. Viterbi, Error bounds for convolutional codes and an asymptotically optimal decoding algorithm, IEEE Transactions on Information Theory 13 (1967) 260–269.
[50] C. Yanover, Y. Weiss, Finding the m most probable configurations in arbitrary graphical models, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, Cambridge, MA, 2004.
[51] M. Yokoo, Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs, in: Proceedings of the Third International Conference on Principles and Practice of Constraint Programming, in: LNCS, vol. 1330, Springer Verlag, 1997, pp. 357–370.