Markov Chain Analysis of Noise and Restart in Stochastic Local Search

Ole J. Mengshoel
Electrical and Computer Engineering
Carnegie Mellon University
ole.mengshoel@sv.cmu.edu

Youssef Ahres
Electrical Engineering
Stanford University
yahres@stanford.edu

Tong Yu
Electrical and Computer Engineering
Carnegie Mellon University
tong.yu@sv.cmu.edu

Abstract

Stochastic local search (SLS) algorithms have proven to be very competitive in solving hard computational problems. This paper investigates the foundations of SLS algorithms. We develop a simple SLS algorithm, MarkovSLS, with three search operators: greedy, noise, and restart. The search operators are controlled by probability parameters, leading to soft (probabilistic) rather than hard (deterministic) restarts. We consider two special cases of the MarkovSLS algorithm: SoftSLS and AdaptiveSLS. In SoftSLS, the probability parameters are fixed, enabling analysis using standard homogeneous Markov chains. We study the interaction between the restart and noise parameters in SoftSLS, and optimize them analytically in addition to the traditional empirical approach. Experimentally, we investigate the dependency of SoftSLS's performance on its noise and restart parameters, validating the analytical results. AdaptiveSLS dynamically adjusts its noise and restart parameters during search. Experimentally, on synthetic and feature selection problems, we compare AdaptiveSLS with other algorithms including an analytically optimized version of SoftSLS, and find that it performs well while not requiring prior knowledge of the search space.

1 Introduction

Stochastic local search (SLS) algorithms are greedy optimizers that also make noisy (random) search steps to improve their performance. The SLS paradigm is simple but broadly applicable [Hoos and Stützle, 2005], and underlies strong algorithms for satisfiability [Selman *et al.*, 1992; 1994; Gu *et al.*, 1997; Hoos and Stützle, 2005] as well as computing the most probable explanation [Kask and Dechter, 1999; Mengshoel, 2008; Mengshoel *et al.*, 2011b] and the maximum a posteriori hypothesis [Park and Darwiche, 2004] in Bayesian networks. Restart, often combined with powerful initialization heuristics, has proven to be essential in many SLS algorithms [Selman *et al.*, 1992; Gent and Walsh, 1993; Ruan *et al.*, 2002; Ryvchin and Strichman, 2008; Mengshoel *et al.*, 2011b; 2011a; Audemard and Simon, 2012].

Despite the empirical success of SLS algorithms, their mathematical foundation is in general under-developed [Wegener, 2001; Mengshoel, 2008]. For example, hard restart as implemented in most classic SLS algorithms [Selman *et al.*, 1992; Gent and Walsh, 1993; Selman *et al.*, 1994; Gu *et al.*, 1997; Hoos and Stützle, 2005; Mengshoel, 2008; Mengshoel *et al.*, 2011b] violates the assumption of a homogeneous Markov chain as we discuss further below. This makes a Markov chain analysis that takes classic restart into account quite complex.

Evolutionary algorithms (EAs) including genetic algorithms (GAs) have been studied using Markov chains [Goldberg and Segrest, 1987; Suzuki, 1995; Yu and Zhou, 2008]. Among EAs, the (1+1)EA [Wegener, 2001; Droste *et al.*, 2002] may appear similar to our MarkovSLS approach. These EAs, however, do not integrate the greedy, noise, and restart steps typically found in SLS algorithms. For example, the (1+1)EA does not have greedy or restart steps and can during mutation flip multiple bits per search step, assuming that the search space is a bit string. Our MarkovSLS algorithm, on the other hand, only flips one bit per search step but integrates greedy, noise, and restart steps.

There is also related research on Las Vegas algorithms, which always compute the correct answer but have a random running time. Luby et al. develop a universal restart strategy for Las Vegas algorithms [Luby *et al.*, 1993]. However, this work does not integrate noise and restart, which is our focus.

More broadly, most previous research on SLS is limited in at least one of the following ways. It (i) is typically either strictly empirical or strictly theoretical; (ii) investigates the impact of a single algorithmic operator (noise, restart, or mutation) to complement greedy search but not several; (iii) does not adapt operator parameters at run-time; or (iv) focuses on problems other than feature selection (for example SAT).

This paper seeks to jointly address all issues (i)–(iv) through a simple algorithm MarkovSLS, which has two special cases SoftSLS and AdaptiveSLS. *Issue (i):* Using deceptive problems and Markov chains, SoftSLS enables us to develop a theoretical foundation to optimize the parameters analytically by means of expected hitting times as opposed to the traditional empirical way. Experimentally, we show the combined impact of the SoftSLS and problem instance parameters on the average running time, validating our expected hitting time results. *Issue (ii):* We study a probabilistic

SLS restart approach, which we call *soft* restart, and embed it into MarkovSLS. Due to the use of soft restart, and in contrast to the classic hard restart, SoftSLS does not violate the homogeneous Markov chain property. We can therefore investigate analytically how restart and noise interact.[1] *Issue (iii):* The novel AdaptiveSLS algorithm, which dynamically adjusts its noise and restart parameters, is shown to have robust performance on several different problem instances. *Issue (iv):* In small-scale experiments, we investigate feature selection problems using SoftSLS and AdaptiveSLS.

In addition, we hope that this work will provide a foundation for future work, both on SLS algorithms and on integrating SLS more tightly with Markov decision processes (MDPs) and other techniques based on Markov chains.

2 Preliminaries and Notation

We consider problem instances as bitstrings $b \in \{0,1\}^n$ and define $N(b)$, the neighbors of b, to be all bitstrings with a Hamming distance of one to b. Without loss of generality, we study maximization problems. For simplicity, we assume a unique optimal solution b^* in our analysis. We construct a one-to-one mapping β from the original search space as follows: 1 means a correct value (relative to b^*) and 0 an incorrect value (relative to b^*). Under these assumptions, $\beta(b^*) = 1...1$ is the unique optimal solution.

We use Markov chains to formalize SLS algorithms.

Definition 1. *(Markov chain) A Markov chain $M = (S,V,P)$ has a set $S = \{s_1, ..., s_n\}$ of states, an initialization vector $V = (\pi_1, .., \pi_n)$, and an $n \times n$ transition probability matrix P. M is homogeneous if $P(X_{t+1} = j \mid X_t = i) = p_{ij}$ is independent of t, else it is inhomogeneous.*

In this paper, "Markov violation" or "violation of the Markov property" is an abbreviation of "violation of the assumption of a homogeneous Markov chain."

The optimal state s^* is what SLS seeks to find. We now define first passage time for s^*.

Definition 2. *(First passage time) Consider a Markov chain $M = (S,V,P)$ with a unique optimal state $s^* \in S$. The first passage time T into s^* is given by: $T = min(j \geq 0 : X_j = s^*)$. The expected value of T given an initial state $s_i \in S$ is:*

$$m_i = E[T \mid X_0 = s_i].$$

Using the definition of first passage time, the expected hitting time for a Markov chain is defined.

Definition 3. *Given a Markov chain $M = (S,V,P)$, the expected hitting time is:*

$$h = \sum_{i=1}^{n} E[T \mid X_0 = i] Pr(X_0 = i) = \sum_{i=1}^{n} m_i \pi_i. \quad (1)$$

The expected hitting time of an SLS algorithm is the analytical counterpart to average running time as measured for an implementation of that algorithm [Mengshoel, 2008].[2]

[1] While very interesting, Luby et al.'s paper, for example, leaves this issue unanswered.

[2] An alternative to using hitting times is to introduce, for optima, absorbing states in the Markov chain and study absorption times [Zhou *et al.*, 2009; Ermon *et al.*, 2014].

```
Data: $p_r$, $p_n$, $f$, $n$, $\tau$, $\alpha_r$, $\alpha_n$
Result: $s'$, $g'$, $t$
t ← 1; g' ← 0;
while $g' < \tau$ do
    $s \leftarrow$ INIT($n$); $g \leftarrow f(s)$;
    if $g \geq g'$ then $g' \leftarrow g$; $s' \leftarrow s$ ;
    $restart \leftarrow false$;
    while !$restart$ and $g' < \tau$ do
        $next \leftarrow$ NEXT_STEP($p_r, p_n$);
        if $next = o_G$ then
            $old\_s \leftarrow s$; $s \leftarrow$ GREEDY_STEP($s$);
            if $old\_s = s$ then
                $p_r = p_r + \bar{p}_r \alpha_r$;
                $p_n = p_n + \bar{p}_n \alpha_n$;
            else
                $p_r = p_r(1 - \alpha_r/2)$;
                $p_n = p_n(1 - \alpha_n/2)$;
            end
        end
        if $next = o_N$ then $s \leftarrow$ NOISE_STEP($s$);
        if $next = o_R$ then $restart \leftarrow true$;
        $g \leftarrow f(s)$;
        if $g \geq g'$ then $g' \leftarrow g$; $s' \leftarrow s$ ;
        $t \leftarrow t + 1$;
    end
end
Return $s'$, $g'$, $t$
```

Algorithm 1: MarkovSLS is an algorithm where noise probability p_n and restart probability p_r can be adapted.

Our deceptive functions, used to evaluate restart, are inspired by deceptive problems [Goldberg, 1987; Whitley, 1991; Deb and Goldberg, 1991; Mengshoel, 2008]. Deceptive problems typically contain a global optimum and a distinct local optimum, also called a deceptive attractor.

Definition 4. *(Deceptive function) Let $n, \gamma, \mu, \delta, x \in \mathbb{N}$ and $d : x \rightarrow \mathbb{R}$, where $0 \leq x \leq n$. A deceptive function $d(x; n, \gamma, \mu, \delta)$, abbreviated $d(x)$, with $0 \leq \delta < \mu < \gamma \leq n$ is defined as follows: there is a unique global maximum $d^* = d(\gamma)$, a local (but non-global) maximum $d(\delta)$, and a local minimum μ (the slope-change location). Formally, we have: $d(x+1) > d(x)$ for $x < \delta$ or $\mu \leq x < \gamma$; $d(x+1) < d(x)$ for $\delta \leq x < \mu$ or $x > \gamma$; and $d(\mu+1) > d(\mu-1)$.*

A deceptive function helps to improve the understanding of the interaction between problem instance difficulty and parameters of stochastic optimization algorithms [Mengshoel, 2008], as reflected in our Markov chain analysis of SoftSLS in Section 4. The parameters n, γ, μ, δ of these deceptive functions can easily be varied to study the restart aspect of SLS algorithms. This is important, given our focus on the soft (probabilistic) restart mechanism.

The notation $\bar{p} = (1 - p)$ is used in this paper.

3 The MarkovSLS Algorithm

Our MarkovSLS algorithm is presented in Algorithm 1, with these inputs and outputs. The inputs are: restart probability p_r; noise probability p_n; pseudo-boolean fitness function f :

$\{0,1\}^n \to \mathbb{R}$; bitstring length n; termination threshold $\tau \in \mathbb{R}$ with $\tau \leq f(s^*)$; restart adaptation parameter $\alpha_r \in [0,1]$; and noise adaptation parameter $\alpha_n \in [0,1]$. The outputs are: approximate maximum s'; approximately maximal fitness $g' = f(s') \geq \tau$; and search step counter t.

MarkovSLS has two main advantages relative to previous work: First, it allows a deeper theoretical analysis of its search by randomizing the restart procedure via p_r, so-called *soft restart*. Second, it introduces a novel *parameter adaptation* mechanism that adapts both the noise and the restart probabilities p_n and p_r based on information collected about the optimization landscape.

Explanation. We first explain the key variables and functions used in MarkovSLS in Algorithm 1. The variables s, s', and *old_s* in Algorithm 1 are bitstrings of length n. INIT(n) creates a bitstring of length n, according to an initialization method. Sampling uniformly at random is such an initialization method. GREEDY_STEP(s) and NOISE_STEP(s) flip one bit in their input s when creating their output, which is then assigned to s. NEXT_STEP(p_r, p_n) randomly decides the next search operator O according to this definition.

Definition 5. *(MarkovSLS search operators) The restart operator o_R has probability $P(O = o_R) = p_r$; the noise operator o_N has probability $P(O = o_N) = (1 - p_r)p_n = \bar{p}_r p_n$; and the greedy operator o_G has probability $P(O = o_G) = (1 - p_r)(1 - p_n) = \bar{p}_r \bar{p}_n$.*

In a GREEDY_STEP(s), the best-fit neighboring state u^* of s, among $u \in N(s)$, is picked as the next state. If there are k candidates u_1^*, u_2^*, ..., u_k^*, such that $f(u_1^*) = f(u_2^*) = \cdots = f(u_k^*)$, one of them is picked as the next step, uniformly at random. In a NOISE_STEP(s), a neighboring state $u \in N(s)$ is picked, uniformly at random, as the next step (regardless of $f(u)$). To simplify analysis, MarkovSLS runs until an achievable input threshold τ is reached.

Soft restart. In classic SLS algorithms, restart is hard (or deterministic) and happens after a fixed number, often called MAX_FLIPS, of search steps [Selman *et al.*, 1992; Gent and Walsh, 1993; Selman *et al.*, 1994; Hoos, 2002; Hoos and Stützle, 2005; Mengshoel, 2008; Mengshoel *et al.*, 2011b]. Since hard restart violates the Markov property, it is difficult to analyze the impact of restart in these classic algorithms. Instead of having a hard boundary on MAX_FLIPS and always restart once we reach it, MarkovSLS uses a probability p_r to restart at each step. This is what we refer to as probabilistic (or soft) restart. With soft restart, the effect of restart on the expected hitting time can be better understood, see Section 4.

Parameter adaptation. MarkovSLS updates p_r and p_n according to the values of the adaptation parameters α_r and α_n.[3] The increments are defined by $p_r = p_r + (1 - p_r)\alpha_r$ and $p_n = p_n + (1 - p_n)\alpha_n$, while the decrements are $p_r = p_r(1 - \alpha_r/2)$ and $p_n = p_n(1 - \alpha_n/2)$. The increase rate is more aggressive than the decrease rate: once stagnation is detected, we deduce that we are trapped in a local (but perhaps not global) maximum and increase the probabilities p_n and p_r to escape it. To detect search stagnation, we simply store the last state visited and compare it with the current state, *i.e.*, a self-loop.

[3]Similar to AdaptiveNoise [Hoos, 2002], we optimized them empirically for a small set of problem instances and then fixed them for the rest of the study. In this paper, their optimized values are: $\alpha_n = 0.2$ and $\alpha_r = 0.1$.

Depending on the values of α_r and α_n, we have different variants of MarkovSLS. In this paper we investigate SoftSLS (using $\alpha_r = \alpha_n = 0$ in MarkovSLS) and AdaptiveSLS (using $\alpha_r > 0$ and $\alpha_n > 0$ in MarkovSLS).

3.1 SoftSLS: MarkovSLS with $\alpha_r = 0$, $\alpha_n = 0$

For $\alpha_r = \alpha_n = 0$, it is easy to see that p_r and p_n remain at their respective input values in MarkovSLS. There is no adaptation. We define this MarkovSLS variant as SoftSLS in reference to the soft restart mechanism.

In extreme cases, the behavior of SoftSLS can easily be deduced. If $p_r = 0$ and $p_n = 1$, it is a random walk. If $p_r = 1$, we restart without any greedy or noisy steps, and perform random sampling according to INIT(n). However, we are most interested in the interaction between p_n and p_r in general, where $0 \leq p_n \leq 1$ and $0 \leq p_r \leq 1$, as we now discuss.

Restart in most SLS algorithms, which we will call *classic* SLS algorithm, works like this: after a number of steps defined by MAX_FLIPS, the search process deterministically restarts. Cleary, this violates the homogeneous Markov chain property. Let ClassicSLS be the same as the SoftSLS, except it has deterministic restart. The SoftSLS and ClassicSLS algorithms have, in a certain sense, similar behavior if we carefully choose their respective input parameters.

Proposition 1. *Let $p_r > 0$ and ${}^1/_{p_r} \in \mathbb{N}^+$. Consider the ClassicSLS algorithm with parameters (p_n,MAX_FLIPS), where MAX_FLIPS is the number of flips before restart, and SoftSLS with parameters (p_n, p_r). Let MAX_FLIPS = ${}^1/_{p_r}$. Then the expected time to restart is the same for ClassicSLS and SoftSLS.*

Proof. Given the MarkovSLS pseudo-code, the number of steps before restarting follows a geometric distribution with a parameter p_r. Thus, its expectation is ${}^1/_{p_r}$. Using the Weak Law of Large Number, we conclude that SoftSLS restarts on average after ${}^1/_{p_r}$ steps. Under the assumption that classic SLS has restart parameter MAX_FLIPS = ${}^1/_{p_r}$, clearly the expectations of the restart times are the same.

For SoftSLS, since p_r and p_n are fixed, we have homogeneous Markov chains, analyzed in Section 4.

3.2 AdaptiveSLS: MarkovSLS with $\alpha_r > 0$, $\alpha_n > 0$

The SoftSLS presented above can be mathematically analyzed to extract optimal parameters for a given problem. However, in practice, we have limited information about the fitness landscape, making the optimization very challenging. Thus, we propose a second special case of MarkovSLS, namely AdaptiveSLS with $\alpha_r > 0$ and $\alpha_n > 0$.

For $\alpha_r > 0, \alpha_n > 0$, p_r and p_n are being adapted in MarkovSLS; we define this MarkovSLS variant as AdaptiveSLS. AdaptiveSLS initializes MarkovSLS with parameters p_n and p_r; MarkovSLS increases them when search stagnates, and decreases them otherwise. The adaptive mechanism in AdaptiveSLS, while relatively simple, appears to be highly competitive with optimized SoftSLS as we will show experimentally.

To our knowledge, previous work has not adapted both noise and restart like AdaptiveSLS. Hoos explored adaptive noise [Hoos, 2002] but not adaptive restart. Luby, Sinclair, and Zuckerman, in contrast, vary restart in their universal strategy but do not investigate adaptive noise [Luby *et al.*, 1993]. Ryvchin and Strichman advocate a related concept, namely localized restarts [Ryvchin and Strichman, 2008].

For AdaptiveSLS, the homogeneous Markov chain property is violated, since p_r and p_n are changing. Consequently, the analysis in Section 4 refers to SoftSLS, where p_r and p_n are fixed (giving a homogeneous Markov chain) and not to AdaptiveSLS (giving an inhomogeneous Markov chain).

4 Markov Chain Analysis of SoftSLS

SLS has been analyzed within a Markov chain framework [Mengshoel, 2008]. However, a main constraint of this previous analysis is that the process never restarts: MAX_FLIPS $= \infty$. In practice, however, restart typically plays an essential role in SLS. The soft restart in SoftSLS enables us to analyze SLS using homogeneous Markov Chains, taking into consideration the whole SLS process including restart.

4.1 Exact Markov Chain Model

We now analyze SoftSLS, and in particular find the transition probabilities in its Markov chain.

Proposition 2. *SoftSLS induces a Markov chain $M = (S,V,P)$ where $S = \{s | s \in \{0,1\}^n\}$, V is given by INIT(n), and P is as follows. Let o_G define a greedy operator, o_R a restart operator, and o_N a noise operator. Let $P(O=o)$ be the prior, where $o \in \{o_G, o_R, o_N\}$. Let SoftSLS be in state m at iteration i, and suppose that $m,k \in \{1,2,\ldots,2^n\}$. The probability of a transition to state k is:*

$$\begin{aligned} P(X_{i+1}=k \mid X_i=m) = \bar{p}_n\bar{p}_r P(X_{i+1}=k \mid X_i=m, O=o_G) \\ + p_n\bar{p}_r P(X_{i+1}=k \mid X_i=m, O=o_N) \\ + p_r P(X_{i+1}=k \mid X_i=m, O=o_R). \end{aligned}$$

Proof. S and V are obvious, and we turn to P. The three possible SoftSLS operators (restart o_R, greedy o_G, and noise o_N) are mutually exclusive. Thus, using the law of total probability, we calculate the probability $P(X_{i+1}=k \mid X_i=m)$ of a new state k by adding up the products of the probability $P(X_{i+1}=k \mid X_i=m, O=o)$ for $o \in \{o_G, o_R, o_N\}$ and their respective priors $P(O=o_G)$, $P(O=o_R)$, and $P(O=o_N)$.

An exact Markov chain analysis of SoftSLS would be most accurate, since it is performed over all $\{0,1\}^n$ states. However, this state space grows exponentially in n, which makes the analysis difficult even for small problem instances.

4.2 Deceptive Markov Chain Model

We now introduce approximate Markov chains, giving a better opportunity to analyze and optimize the SoftSLS algorithm. We introduce deceptive Markov chains, used to evaluate the behavior of SoftSLS as its parameters vary.

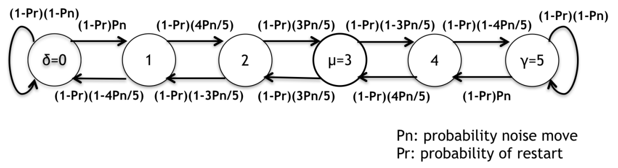

Figure 1: Deceptive Markov Chain: DMC$(p_n, p_r; 5,3,0)$ with transition probabilities.

Definition 6. *(Deceptive Markov chain) An n-bit deceptive Markov chain (DMC) with local optimum δ, global optimum $\gamma = n$, local minimum μ, probability of noise step p_n and probability of restart p_r, abbreviated DMC$(p_n, p_r; \gamma, \mu, \delta)$, is defined as follows. It has $n+1$ states, $S = \{0,..,n\}$, and the initial distribution V is defined for $i \in S$, as $\pi_i = \binom{n}{i}/2^n$. State transition probabilities are for $0 \le x < \delta$ or $\mu \le x < n$ defined as:*

$$P(X_{i+1} = x+1 \mid X_i = x) = \bar{p}_r(1 - \frac{x}{n}p_n) + p_r\pi_{x+1},$$

for $\delta \le x < \mu$:

$$P(X_{i+1} = x+1 \mid X_i = x) = \bar{p}_r(\frac{n-x}{n}p_n) + p_r\pi_{x+1},$$

for $0 < x \le \delta$ or $\mu \le x \le n$:

$$P(X_{i+1} = x-1 \mid X_i = x) = \bar{p}_r(\frac{x}{n}p_n) + p_r\pi_{x-1},$$

for $\delta < x < \mu$:

$$P(X_{i+1} = x-1 \mid X_i = x) = \bar{p}_r(1 - \frac{n-x}{n}p_n) + p_r\pi_{x-1},$$

and for $x = 0$ or $x = n$:

$$P(X_{i+1} = x \mid X_i = x) = \bar{p}_r\bar{p}_n.$$

The above equations divide the transitions into regions: the first two regions consist of increasing slopes between 0 and the local optimum δ and between local minimum μ and the global maximum $\gamma = n$. (We have generally assumed, in this paper, that $\gamma = n$ for simplicity.) The third region consists of a decreasing slope between δ, the deceptive optimum, and the local minimum μ. Within each of these two regions, SLS moves forward (toward γ) or backward (away from γ) according to the above equations.

Proposition 3. *If INIT(n) in SoftSLS(p_r,p_n,f,n τ) is initialization uniformly at random and f is a deceptive function $d(x; n, \gamma, \mu, \delta)$, then the Markov chain induced by SoftSLS for $\tau = f(n)$ is a DMC, DMC$(p_n, p_r; \gamma, \mu, \delta)$.*

Proof Sketch. This follows from the definitions of SoftSLS and of a deceptive function. We start by formulating a Markov chain, then using the fact that f is a deceptive function, we divide the Markov chain into regions: two ascending slope regions and a descending slope region as explained above. It is then easy to deduce the probability to go to a given state using p_n and p_r, resulting in *DMC$(p_n, p_r; \gamma, \mu, \delta)$*.

Figure 1 shows as an example DMC$(p_n, p_r; 5,3,0)$, where each directed edge is a probabilistic transition. When $p_r > 0$ in SoftSLS, each node should be connected to all other nodes, since restart can lead the search from a node to any other node. To improve readability, we omit these restart edges.

Generally, we are interested in the influence that the problem difficulty parameters μ and δ in a DMC have on the expected hitting time, and their interaction with the SoftSLS parameters (p_n, p_r).

Proposition 4. *Let M be a vector of expected first passage times for a DMC$(p_n, p_r; \gamma, \mu, \delta)$; m_i for $0 \leq i \leq n$. Clearly, $m_n = 0$ as we assumed $\gamma = n$. We obtain the following. For $i = \delta$:*

$$m_i = 1 + \bar{p_r}(\frac{n-\delta}{n} p_n m_{\delta+1} + \frac{\delta}{n} p_n m_{\delta-1} + \bar{p_n} m_\delta) + p_r \times MV^t,$$

for $\delta < i < \mu$:

$$m_i = 1 + \bar{p_r}(\frac{n-i}{n} p_n m_{i+1} + (1 - \frac{n-i}{n} p_n) m_{i-1}) + p_r \times MV^t,$$

for $0 < i < \delta$ or $\mu \leq i < n$:

$$m_i = 1 + \bar{p_r}((1 - \frac{i}{n} p_n) m_{i+1} + \frac{i}{n} p_n m_{i-1}) + p_r \times MV^t,$$

and if $i = 0$ and $\delta \neq 0$:

$$m_i = 1 + \bar{p_r} m_{i+1} + p_r \times MV^t.$$

For the same reason as in Definition 5, the above equations are divided into two regions. We see that the expressions for m_i greatly depend on μ, δ, p_r, and p_n in addition to the current state i. Thus, we use the notation $m_{i,\mu,\delta}(p_n, p_r)$ to more fully characterize the equations.

Following (1), the expected hitting time for a DMC problem of size n and difficulty parameters μ and δ is:

$$h_{n,\mu,\delta}(p_n, p_r) = \sum_{i=0}^{n} \pi_i m_{i,\mu,\delta}(p_n, p_r). \qquad (2)$$

From (2), the expected hitting time for a particular deceptive Markov chain, formally denoted as DMC$(p_n, p_r; \gamma, \mu, \delta)$, can be computed. An example is provided in Section 4.3.

4.3 Markov Chain Optimization

The above formulation establishes a framework to analyze the performance of SoftSLS on a given search landscape. In this section, we show how we can utilize it to optimize the SoftSLS parameters on that particular landscape. Then, we propose a strategy to generalize this optimization approach to non-trivial problems and unknown landscapes.

An Optimization Example

Let us consider how to solve a DMC and find an optimal pair (p_n^*, p_r^*). We use a small DMC, with $n = 5$.

Consider the DMC$(p_n, p_r; 5, 3, 0)$ as shown in Figure 1 along with its transition probabilities. We will now derive $h_{5,3,0}(p_n, p_r)$. The Markov chains initialization vector is $V = (\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5)$ and we define the vector $M = (m_0, m_1, m_2, m_3, m_4, m_5)$ where $m_5 = 0$ and:

$$m_0 = 1 + \bar{p_r}((1 - p_n) m_0 + p_n m_1) + p_r \times MV^t$$
$$m_1 = 1 + \bar{p_r}((1 - (4/5) p_n) m_0 + (4/5) p_n m_2) + p_r \times MV^t$$
$$m_2 = 1 + \bar{p_r}((1 - (3/5) p_n) m_1 + (3/5) p_n m_3) + p_r \times MV^t$$
$$m_3 = 1 + \bar{p_r}((1 - (3/5) p_n) m_4 + (3/5) p_n m_2) + p_r \times MV^t$$
$$m_4 = 1 + \bar{p_r}((4/5) p_n m_3 + (1 - (4/5) p_n) m_5) + p_r \times MV^t.$$

Table 1: Datasets for feature selection experiments.

Dataset	# features	# instances
Diabetes	8	768
Breast Cancer	9	700
Wine	13	178
Australian	14	690

Assuming uniform restart in SoftSLS, we have $\pi_i = \binom{5}{i}/2^5$. By solving, we obtain m_i for $i \in \{0, 1, 2, 3, 4, 5\}$ as a function of p_n and p_r. Then, we compute $\sum_0^5 E[T \mid X_0 = i] P[X_0 = i] = \sum_0^5 m_i \pi_i$ to obtain the expected hitting time $h_{5,3,0}(p_n, p_r)$. Given that they are bivariate, the expression for this solution is quite space-consuming, even for this small 5-bit problem. It is thus omitted here.

Using the solution expression for $h_{5,3,0}(p_n, p_r)$ and the active-set algorithm [Nocedal and Wright, 2006], we can deduce an optimal pair $(p_n^*, p_r^*) = (0, 0.346)$ that minimizes the expected hitting time of the Markov chain.

An optimal value of $p_n^* = 0$ may seem surprising, since it goes against the idea of escaping from local optima via noise. However, $p_n^* = 0$ is a direct consequence of the deceptive function studied, which mainly stresses the restart aspect of MarkovSLS as discussed in Section 2. In fact, $p_n^* \neq 0$ for many real-world and synthetic problems.[4]

Towards Optimization in General

Above, we formulated a Markov chain and found the optimal (p_n^*, p_r^*) for a known 5-bit problem. However, in practice, we may not know the shape of the search space or its characteristics such as the number of local optima or their locations. Therefore, a procedure to generalize the analysis presented above is necessary for it to inform applications. A very simple, yet potentially powerful, strategy consists of using minimal knowledge of the problem and optimize the parameters over a set of *DMC*s. For instance, by knowing problem size n, we can solve the *DMC*$(p_n, p_r; \gamma, \mu, \delta)$ for combinations of (γ, μ, δ). This makes a strong assumption about the nature of the problem instance, which may or may not be appropriate. However, the approach has a cubic complexity and there is opportunity for pre-processing.

5 Experimental Results

In experiments with synthetic and real-world problems, we compare SoftSLS, AdaptiveSLS, AdaptiveNoise [Hoos, 2002], and Simulated Annealing [Kirkpatrick *et al.*, 1983].

5.1 Methods and Data

We use both synthetic (deceptive Markov chains) and real-world datasets (feature selection) in our experiments. Table 1

[4]For real-world examples, we refer to feature selection problems in Section 5.4. There are also synthetic problems where $p_n^* \neq 0$ (in the optimal pair). In fact, for deceptive functions with $n \geq 10$, this is usually the case. For a family of deceptive functions and their corresponding DMCs, we found an average of $p_n = 0.075$ and $p_r = 0.0388$ over optimal pairs, when averaging over a range of values of δ and μ.

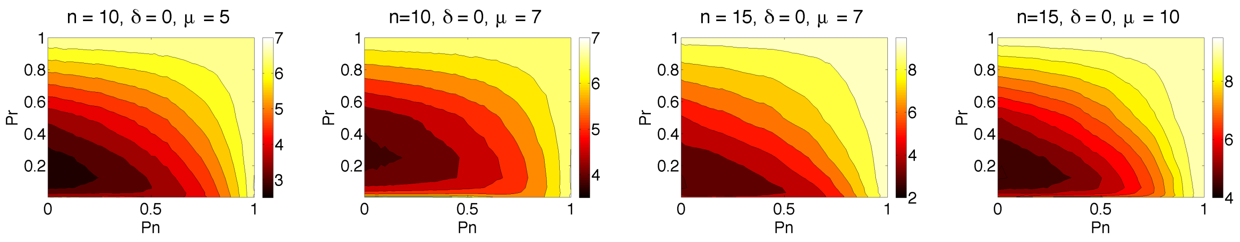

Figure 2: Experimental contour plots of the average running time on a *logarithmic* scale for varying noise and restart probabilities p_n and p_r for synthetic DMC problems. Each contour plot shows p_n on the x-axis and p_r on the y-axis.

Table 2: Average running time results for optimized SoftSLS, AdaptiveSLS, AdaptiveNoise (AN), and Simulated Annealing (SA) on synthetic DMC problems.

(δ, μ)	**SoftSLS**	**AdaptiveSLS**	**AN**	**SA**
$(0,3)$	7.80	6.57	6.78	984.31
$(0,6)$	23.65	23.14	21.02	1036.12
$(2,5)$	14.42	11.36	12.47	1004.67
$(2,7)$	45.19	46.70	37.51	956.78
$(5,6)$	45.64	44.24	46.56	966.13
$(5,7)$	44.70	45.32	48.91	989.48
$(7,9)$	339.37	385.57	298.30	1022.22

Table 3: Average running time results for optimized SoftSLS, AdaptiveSLS, and AdaptiveNoise (AN) on real-world feature selection problems.

Dataset	**SoftSLS**	**AdaptiveSLS**	**AN**
Diabetes	18.36	22.92	23.03
Breast Cancer	19.72	26.45	21.85
Wine	29.31	33.62	43.05
Australian	34.43	43.55	52.93

presents the real-world datasets used for feature selection.[5]

In experiments we use $\tau = f(s^*)$ as input to MarkovSLS. Consequently, both average running time and expected hitting time refer to a global optimum, and fitness or accuracy when terminating does not vary between experiments. Difference in fitness or accuracy is not a confounding factor here.

For apples-to-apples comparisons with analytical results, we measure the average running time, via MarkovSLS's output t. An average over 500 experiments is computed, giving an empirical estimate of the expected hitting time. We also include experimental results for AdaptiveNoise (AN) [Hoos, 2002] and Simulated Annealing (SA) [Kirkpatrick *et al.*, 1983].

5.2 SoftSLS on Synthetic Problems

We now study deceptive Markov chains DMC$(p_n, p_r; n, \mu, \delta)$ with $\delta = 0$. Using different values for p_r, p_n, and μ, we study the performance of SoftSLS. Figure 2 reflects the behavior of SoftSLS for varying p_n on the x-axis and varying p_r on the y-axis.

The two experimental plots shown leftmost in Figure 2 (for $n = 10$) correspond well to the expected hitting time results found analytically.[6] This validates our Markov chain analysis. The rightmost two plots show similar behavior for $n = 15$. Another interesting result is the behavior of the average running time for small values of p_n. For very small and very large values of p_r, SoftSLS performs poorly. However, by carefully choosing p_r, we can drastically reduce the running time. The contour plots in Figure 2 highlight this behavior. This shows how crucial the restart parameter can be to optimize the running time of SLS algorithm.

5.3 AdaptiveSLS on Synthetic Problems

We investigate the performance of AdaptiveSLS, including its response to varying synthetic problem parameters. The DMC problem size is $n = 10$, and results are averaged over 500 experiments. In order to perform a stringent comparison, we optimized SoftSLS mathematically using its Markov chain analysis.

Table 2 shows a summary of our results. Perhaps surprisingly, AdaptiveSLS sometimes performs better than an optimized SoftSLS. The intuition behind this is that AdaptiveSLS adapts its p_r and p_n parameters given the current state of search, whether stagnated or not, without knowing the globally optimal values of the parameters. On the other hand, SoftSLS keeps its globally optimized parameters (p_n^*, p_r^*) independently of the current state, which may cause poor performance if it is stuck searching close to a local optimum.

5.4 Use Case: Feature Selection Problems

We now study the performance of SoftSLS and AdaptiveSLS on the real-world problem of feature selection. The goal of the experiment is to investigate, in a randomized wrapper approach to feature selection [Stracuzzi, 2007; Kohavi and John, 1997], whether AdaptiveSLS can achieve comparable results to SoftSLS, without intensive parameter optimization. For each MarkovSLS bitstring s we have a corresponding feature subset. A 0-bit in s means that the corresponding feature is not included in the feature subset; a 1-bit means that it is.

We trained a Decision Tree, on varying feature subsets and computed the cross-validation accuracy. In feature selection, the goal is to find a bitstring s^* with the maximal cross-validation accuracy. Thus, we set a bitstring s's fitness $f(s)$ as

[5]Please see https://archive.ics.uci.edu/ml and http://www.liacc.up.pt

[6]Due to space restrictions, we do not show the analytical contour plots corresponding to Figure 2 here. The analysis was done as discussed in Section 4.3.

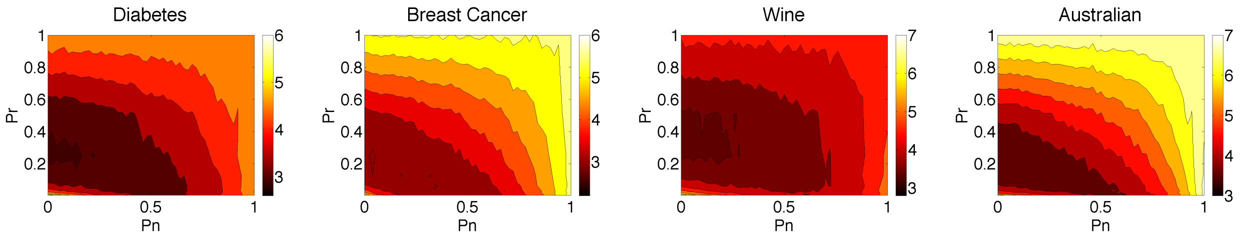

Figure 3: Experimental contour plots of the average running time on a *logarithmic* scale for varying noise and restart probabilities p_n and p_r for real-world feature selection problems, from left to right: Diabetes, Breast Cancer, Wine, and Australian. Each contour plot shows p_n on the x-axis and p_r on the y-axis.

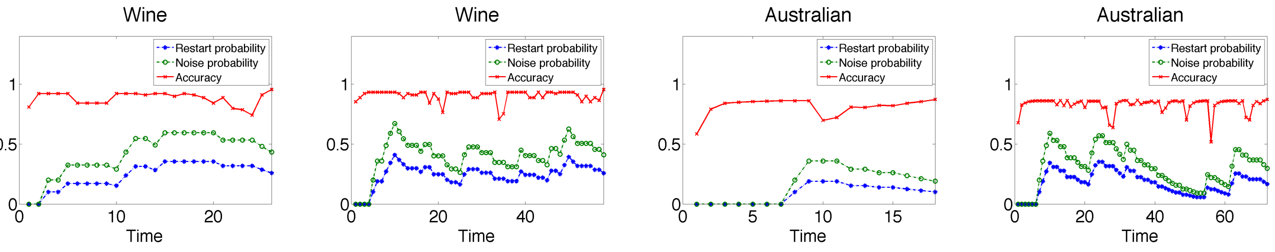

Figure 4: Typical behaviors of AdaptiveSLS on Wine and Australian data in four experiments. The x-axis shows the number of AdaptiveSLS search steps (t in Algorithm 1), and the y-axis shows p_r, p_n and cross validation accuracy.

its cross-validation accuracy, using 2-fold cross-validation.[7] For each dataset, we searched exhaustively to predetermine the globally optimal feature subset s^*, and passed $\tau = f(s^*)$ to both SoftSLS and AdaptiveSLS. Both algorithms terminate when finding a globally optimal feature subset.

In the comparison between AdaptiveSLS and SoftSLS in the feature selection problems, the results of SoftSLS with empirically optimized parameters are reported. We report the average results of SoftSLS and AdaptiveSLS over 500 experimental runs, to reduce noise.

Table 3 and Figure 3 summarize the average running time results. In general, when the dataset's dimension is larger, the running time is larger as reflected in Table 3. As shown in Figure 3, the approximately optimal pairs are: Diabetes: $p_r^* = {}^1/_4$, $p_n^* = {}^2/_{100}$; Australian: $p_r^* = {}^1/_4$, $p_n^* = 0$; Wine: $p_r^* = {}^1/_4$, $p_n^* = {}^6/_{100}$; and Breast Cancer: $p_r^* = {}^1/_4$, $p_n^* = {}^4/_{100}$. These pairs were found by grid search over $9 \times 51 = 459$ points (51 values $p_n \in \{0, {}^2/_{100}, {}^4/_{100}, \ldots\}$ and 9 values $p_r \in \{1, {}^1/_2, {}^1/_4, {}^1/_8, {}^1/_{16}, \ldots\}$). As we can see in Figure 3, the near-optimal pairs of p_n and p_r form a region. In this region, we have (p_n, p_r) giving either exactly the same or very similar performance to the optimal pair (p_n^*, p_r^*). For Wine, a relatively high p_r is necessary in order to minimize average running time, even after considering this region.

Note that AdaptiveSLS does not rely on prior knowledge about p_r^* and p_n^*, which can save substantial computing time. On Diabetes, Wine and Australian, AdaptiveSLS can also beat AdaptiveNoise [Hoos, 2002] where only noise p_n is adapted, in terms of average running time. This demonstrates the potential of adapting both the restart and noise probabilities as done in AdaptiveSLS.

Figure 4 shows the evolution of p_r and p_n in AdaptiveSLS during search, after initialization $p_r = p_n = 0$. Generally, we can divide the search process into two phases: In the first phase, the algorithm often gets stuck in a local optimum. Thus, p_r and p_n are increased. Due to these high probabilities for p_r and p_n, we are able to get out of the trap. Then, a second phase starts where we converge towards an optimum while the probabilities decrease. The two phases may repeat, as shown in the rightmost plot of Figure 4. At the end of search, the found p_r and p_n for Wine and Australian match well with the optimal regions shown in Figure 3.

6 Conclusion

This paper presents a simple SLS algorithm, MarkovSLS. We study two special cases of the algorithm: SoftSLS and AdaptiveSLS. SoftSLS avoids violating the homogeneous Markov chain property by modifying the deterministic restart approach of most SLS algorithms, and using instead a probabilistic restart. This probabilistic restart enables a broad Markov chain and hitting time analysis, including finding the global optima of the noise and restart parameters, for synthetic problems. Experimentally, we study the impact of varying the use of the greedy, noise, and restart operators on the time it takes SoftSLS to find an optimal solution. In contrast to SoftSLS, AdaptiveSLS, dynamically adjusts the probability parameters. Experimentally, we find that the AdaptiveSLS algorithm behaves very well compared to optimized SoftSLS, both for synthetic deceptive problems and for real-world feature selection problems.

[7] In multi-class problems, like the Wine dataset, if the number of folds is too large, validation performance will degrade due to imbalance between different folds.

References

[Audemard and Simon, 2012] G. Audemard and L. Simon. Refining restarts strategies for SAT and UNSAT. In *Principles and Practice of Constraint Programming*, pages 118–126. 2012.

[Deb and Goldberg, 1991] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In *Foundations of Genetic Algorithms*, pages 93–108. Morgan Kaufmann, 1991.

[Droste *et al.*, 2002] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(12):51 – 81, 2002.

[Ermon *et al.*, 2014] S. Ermon, C. P. Gomes, A. Sabharwal, and B. Selman. Designing fast absorbing Markov chains. In *Proc. of AAAI-14*, pages 849–855, 2014.

[Gent and Walsh, 1993] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proc. of AAAI-93*, pages 28–33, 1993.

[Goldberg and Segrest, 1987] D. E. Goldberg and P. Segrest. Finite Markov chain analysis of genetic algorithms. In *Proc. of the Second International Conference on Genetic Algorithms and Their Application*, pages 1–8, 1987.

[Goldberg, 1987] D. E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 74–88. Pitman, 1987.

[Gu *et al.*, 1997] P. W. Gu, J. Purdom, J. Franco, and B. W. Wah. *Satisfiability Problem: Theory and Applications*, chapter Algorithms for the Satisfiability SAT Problem: A Survey, pages 19–152. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1997.

[Hoos and Stützle, 2005] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, 2005.

[Hoos, 2002] H. H. Hoos. An adaptive noise mechanism for WalkSAT. In *Proc. of AAAI-02*, pages 655–660, 2002.

[Kask and Dechter, 1999] K. Kask and R. Dechter. Stochastic local search for Bayesian networks. In *Proc. of AISTATS-99*, pages 113–122, 1999.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Kohavi and John, 1997] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[Luby *et al.*, 1993] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

[Mengshoel *et al.*, 2011a] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Portfolios in stochastic local search: Efficiently computing most probable explanations in Bayesian networks. *Journal of Automated Reasoning*, 46(2):103–160, 2011.

[Mengshoel *et al.*, 2011b] O. J. Mengshoel, D. C. Wilkins, and D. Roth. Initialization and restart in stochastic local search: Computing a most probable explanation in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(2):235–247, 2011.

[Mengshoel, 2008] O. J. Mengshoel. Understanding the role of noise in stochastic local search: Analysis and experiments. *Artificial Intelligence*, 172(8-9):955–990, 2008.

[Nocedal and Wright, 2006] J. Nocedal and S. J. Wright. *Numerical Optimization (2nd Edition)*. Springer, 2006.

[Park and Darwiche, 2004] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research (JAIR)*, 21:101–133, 2004.

[Ruan *et al.*, 2002] Y. Ruan, E. Horvitz, and H. Kautz. Restart policies with dependence among runs: A dynamic programming approach. In *Proc. of the Eighth International Conference on Principles and Practice of Constraint Programming*, pages 573–586, 2002.

[Ryvchin and Strichman, 2008] V. Ryvchin and O. Strichman. Local restarts in SAT. *Constraint Programming Letters (CPL)*, 4:3–13, 2008.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of AAAI-92*, pages 440–446, 1992.

[Selman *et al.*, 1994] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proc. of AAAI-94*, pages 337–343, 1994.

[Stracuzzi, 2007] David J Stracuzzi. Randomized feature selection. In *Computational Methods of Feature Selection*, pages 41–62. CRC Press, 2007.

[Suzuki, 1995] J. Suzuki. A Markov chain analysis on simple genetic algorithms. *IEEE Trans. Systems, Man, and Cybernetics*, 25(4):655–659, 1995.

[Wegener, 2001] Ingo Wegener. Theoretical aspects of evolutionary algorithms. In *Proc. of ICALP-01*, pages 64–78, 2001.

[Whitley, 1991] L. D. Whitley. Fundamental principles of deception in genetic search. In *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, 1991.

[Yu and Zhou, 2008] Y. Yu and Z.-H. Zhou. A new approach to estimating the expected first hitting time of evolutionary algorithms. *Artificial Intelligence*, 172(15):1809 – 1832, 2008.

[Zhou *et al.*, 2009] Y. Zhou, J. He, and Q. Nie. A comparative runtime analysis of heuristic algorithms for satisfiability problems. *Artificial Intelligence*, 173(2):240–257, 2009.