Probabilistic Analysis of Local Search and NP-Completeness Result for Constraint Satisfaction

(Extended Abstract)

Hoong Chuin LAU

Dept. of Computer Science, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152, Japan
email: hclau@cs.titech.ac.jp

1 Introduction

An instance of the binary Constraint Satisfaction Problem (CSP) is defined by a set of variables, their associated value domains and a set of binary constraints. The output is an assignment of variables to domain values which satisfies all constraints simultaneously. Many problems in Artificial Intelligence, Operations Research and Combinatorics can be formulated as CSP, such as problems in scene labelling, temporal reasoning and resource allocation. CSP can be modelled as a graph where the vertices and edges represent the variables and constraints. It is a generalization of the well-known graph-coloring problem.

The CSP is known to be NP-hard in general. One research direction to tackle NP-hard problems is to identify subclasses which are polynomial-time solvable. Recently, van Beek and Dechter [8] proved that CSP is polynomial-time solvable if the set of constraints are *row-convex* and *path-consistent* with respect to the given graph.

Another direction is to design efficient algorithms to solve random instances of the problem. Koutsoupias and Papadimitriou [3] proved that, for random satisfiable 3CNF Boolean formulas, if the initial assignment agrees with the satisfying assignment in more than half the number of variables, then local search based on flipping succeeds with high probability. Alon and Kahale [1] proved that random 3-colorable graphs can be colored efficiently with high probability when the edge probability is at least $polylog(n)/n$. Minton *et al.* [4] considered CSP instances on regular graphs. Under the assumption that instances have unique solutions and constraints are consistent with a fixed probability, they proved that local search performs well when the Hamming distance between the initial assignment and the solution is small. The Hamming distance refers to the number of variables with different assigned values.

Local search is an appealing paradigm because it is conceptually simple and efficiently implementable. While a pervasive amount of experimental results have been presented by AI researchers on applying local search to solve CSP in general (e.g. see the Constraint Satisfaction section in *AAAI* and *IJCAI* conference proceedings), little is known theoretically for restricted classes of CSP.

In this paper, we study the computational complexity of and local search algorithms for restricted classes of CSP. Our contributions are as follows.

1. We prove that the decision problem of CSP remains NP-complete even if the constraint graph is a grid-graph having *row-convex* constraints. This provides a theoretical limit to a result in [8].
2. We consider satisfiable CSP instances drawn randomly from a distribution parameterized by the edge probability q and the probability λ that a constraint is satisfied. We show that:
   - If $q \geq \frac{\log n}{(1-\lambda)n}$ and $\lambda \leq \frac{\ln k}{k}$, then almost every instance has only one solution. Furthermore, if the initial assignment is 'good', i.e. the Hamming distance between the initial assignment and the solution is less than half the number of variables, then our local search algorithm finds the solution with high probability. Our analysis is a refinement of [4] and an improvement of their result.
   - For $q \geq \log n/n$ and $0.43 \leq \lambda \leq 1$, a solution can be found with high probability after a polynomial number of local search iterations having independent random initial assignments. Selman et al. [6] proposed an iterative local search algorithm GSAT and showed empirically that it works well for random satisfiable 3CNF formulas. We mimic their algorithm and formalize its behavior on CSP.

2 Preliminaries

The Constraint Satisfaction Problem (CSP) is defined by a triple (V, D, R), where V is a set of n *variables* $\{1, 2, ..., n\}$, D is a set of *domains* $\{D_1, D_2, ..., D_n\}$, and R is a collection of *binary constraints* (or simply constraints). Each domain D_i is a finite set of values that can be assigned to variable i. Unless otherwise specified, we will assume that all domains have a fixed constant size k and are equal to the set $K = \{1, 2, \ldots, k\}$. A *constraint*

$R(i_1, i_2)$ is a relation on $K \times K$ specifying pairs of values which can be assigned to variables i_1 and i_2 simultaneously. All value pairs in $R(i_1, i_2)$ are said to be *consistent.* Given an assignment $\sigma : V \longrightarrow K$, $R(i_1, i_2)$ is *satisfied* under σ if $(\sigma_{i_1}, \sigma_{i_2}) \in R(i_1, i_2)$. Otherwise, we say that i_1 *conflicts* with i_2 under σ. A *solution* is an assignment which satisfies all constraints simultaneously. An instance is *satisfiable* iff it has at least one solution.

A constraint $R(i_1, i_2)$ is said to be *row-convex* from i_1 to i_2 iff there is an ordering of the values in D_{i_2} such that every value in D_{i_1} is consistent with only consecutive values of D_{i_2}. For example, the constraint

$$\{(1,2), (2,1), (2,2), (3,2), (3,3)\}$$

is row-convex while the 'not-equal' constraint

$$\{(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)\}$$

is not row-convex. A CSP instance is row-convex iff there exists an ordering of the variables and an ordering of the values in every domain such that for all $i_1 \leq i_2$, $R(i_1, i_2)$ is row-convex from i_1 to i_2. Let rc-CSP denote the decision problem whose input is a row-convex CSP instance and output is whether the instance is satisfiable. Note that rc-CSP does not include graph-coloring instances, because the constraints for graph coloring are all 'not-equal' constraints.

Let R-CSP denote a set of distributions on satisfiable CSP instances. An element of R-CSP is a specific distribution parameterized by the *edge probability* $0 \leq q \leq 1$ and the *consistency probability* $0 \leq \lambda \leq 1$, and a CSP instance is drawn from this distribution by the following procedure:

1. Generate a random graph G of n nodes such that an edge (i.e. constraint) exists between any two variables with probability q.
2. Generate a random assignment $\hat{\sigma}$. For each edge (i_1, i_2) in G, construct the constraint as follows. Insert the value pair $(\hat{\sigma}_{i_1}, \hat{\sigma}_{i_2})$ with probability 1 and all other $k^2 - 1$ pairs with probability λ.

This generation procedure has been used by others and an online implementation can be found in [7]. Let m denote the number of edges in G. Let $p = 1 - \lambda$ be the *inconsistency probability.* For brevity, an instance generated by the above method will be called an R-CSP instance. Clearly, every R-CSP instance is satisfiable since it has at least one solution $\hat{\sigma}$. Further, it has the following property:

Fact 1. For all constraints $R(i_1, i_2)$ and all value pairs (j_1, j_2),

$$\Pr[(j_1, j_2) \in R(i_1, i_2)] = \begin{cases} 1, \text{ if } j_1 = \hat{\sigma}_{i_1} \text{ and } j_2 = \hat{\sigma}_{i_2} \\ \lambda, \text{ otherwise.} \end{cases}$$

The expected probability that (j_1, j_2) is a consistent pair is $\phi = \left(1 - \frac{1}{k^2}\right)\lambda + \frac{1}{k^2}$.

Henceforth, the phrase *almost surely* means: with probability approaching 1 when n is sufficiently large. All log's are base 2 and 'ln' refers to the natural log.

3 NP-Completeness of rc-CSP

Recently, van Beek and Dechter [8] proved that rc-CSP is polynomial-time solvable if the constraints are *path-consistent* with respect to the given graph. In other words, for any pair of variables (i_1, i_2) assigned to consistent values j_1 and j_2 respectively, there exists a value j_3 for any variable i_3 such that these three values satisfy the constraints $R(i_1, i_3)$ and $R(i_2, i_3)$. Path-consistency is a well-studied notion in the of design of algorithms for CSP. In this section, we show that without the path-consistency property, unfortunately, rc-CSP remains NP-complete under certain restrictions.

Theorem 1. *rc-CSP is NP-complete, even for grid graphs and domain size 4.*

Proof. (sketch) We show by a many-one reduction from the 3-satisfiability problem (3-SAT). Let F be an instance of 3-SAT with n Boolean variables $\{U_1, \ldots, U_n\}$ and m clauses. Assume that for each clause, if the literal U_i occurs in the clause, then $\overline{U_i}$ will. not occur and vice versa. Construct an $n \times m$ grid such that nodes in row i represent the variable U_i and nodes in column j represent the clause C_j. Let the node on row i and column j be called $x_{i,j}$. Call a node $x_{i,j}$ *active* if U_i or $\overline{U_i}$ occurs in C_j, and *passive* otherwise.

We introduce 7 distinct values $\{0, \theta, 1, a, c, b, d\}$. Call the subset $\{\theta, 1, c, d\}$ the accepting set. Except the first and last rows, active nodes have domain $\{0, \theta, 1\}$ while passive nodes have domain $\{a, b, c, d\}$. In the first row, active nodes have domain $\{0, 1\}$ while passive nodes have domain $\{a, b\}$. In the last row, active nodes have domain $\{\theta, 1\}$ while passive nodes have domain $\{c, d\}$. The value 1 represents the truth value *True*, while both

0 and θ represent $False$. The values a and c are used to convey 0 or θ while b and d convey 1.

Two main issues need to be addressed. First, values of active nodes must be consistently conveyed along the rows. This is essential so that every Boolean variable carries exactly one truth value throughout all clauses. This condition is enforced by the horizontal constraints. Next, every clause must be satisfied, i.e. contains at least one true literal. This is why we need two values to represent $False$. Basically, an active node is assignable to 0 only if no active node in the same column and a smaller-number row is assigned 1. Similarly, a passive node is assignable to a or b only if no active node in the same column and a smaller-number row is assigned 1. By this approach, enforcing the satisfiability of clause C_j is equivalent to enforcing the last node of column j to have a value of the accepting set. This condition can be enforced by the vertical constraints.

All constraints can be shown to be row-convex and thus the instance is an rc-CSP instance. Due to space constraint, we illustrate how values are conveyed via horizontal constraints by an example. Suppose we have three nodes $x = x_{i,j-1}$, $y = x_{i,j}$ and $z = x_{i,j+1}$. Further suppose the literal U_i occurs in clause $j-1$ and the literal $\overline{U_i}$ occurs in clause $j+1$. Thus x and z are active while y is passive. We define the following constraints to ensure that x and z carry opposite truth values:

$$\begin{aligned} R(x,y) &= \{(0,b),(0,d),(\theta,b),(\theta,d),(1,a),(1,c)\}, \\ R(y,z) &= \{(a,0),(a,\theta),(b,1),(c,0),(c,\theta),(d,1)\}. \end{aligned}$$

If x is assigned to 1, then y must be assigned to a or c and thus z must be assigned to 0 or θ. Else if x is assigned to 0 or θ, then y must be assigned to b or d and thus z must be assigned to 1.

Suppose we have a solution for the rc-CSP instance. For every i, let j_i be the clause in which either the literal U_i or $\overline{U_i}$ first occurs. If U_i occurs, then set $t(U_i)$ to $True$ when $x_{i,j_i} = 1$, and $False$ when $x_{i,j_i} = 0$ or θ. If $\overline{U_i}$ occurs instead, do the reverse. The resulting truth assignment t satisfies F because every column has at least one '1'.

Conversely, suppose we have a satisfying assignment t. Assign the active nodes from top to bottom, left to right. For each $t(U_i) = True$, assign node $x_{i,j}$ to 1 for all j such that U_i occurs in C_j, and to 0 or θ for all j such that $\overline{U_i}$ occurs in C_j (depends on whether a '1' has been assigned to a node in the same column and previous rows). Do the reverse for each $t(U_i) = False$. Assign the passive nodes from top to bottom, left to right. Clearly, such assignment is possible. Since each clause has at least one true literal, the nodes in the last row will be assigned values in the accepting set. □

Example Figure 1 gives an example of the reduction. (a) gives the rc-CSP instance constructed from the 3-SAT instance

$$U = \{U_1, U_2, U_3, U_4, U_5\}, F = \{(U_1, U_2, U_5), (\overline{U_2}, \overline{U_3}, \overline{U_4}), (\overline{U_1}, \overline{U_3}, \overline{U_4}), (U_1, U_3, \overline{U_5})\}.$$

Active nodes are unshaded while passive nodes are shaded. From the solution (b), the corresponding satisfying assignment is $\{True, False, True, False, True\}$.

Fig. 1. Example of reduction from 3-SAT to rc-CSP.

4 Local Search Algorithm LS1

In this section, we present a simple local search algorithm LS1 and analyze its performance.

Given an assignment σ, let $\Delta(\sigma, i, j)$ denote the assignment obtained by replacing the assigned value of variable i (i.e. σ_i) with another value j. The logic of our local search algorithm LS1 is simple. Begin with a random assignment and iterate until we reach a solution or all variables have been reset. Let σ denote the current assignment and S denote the set of variables which have not been reset. Define a potential function $c : S \times K \longrightarrow \mathbb{N}$:

$$c(i, j) = \#\{v \in V - \{i\} \mid (j, \sigma_v) \notin R(i, v)\}$$

That is, $c(i,j)$ counts the number of variables which conflict with the variable i under the assignment $\varDelta(\sigma,i,j)$. Among all variables in S, the algorithm selects a variable-value pair (i,j) such that the potential function is minimized and reset i to the value j. LS1 is codified as follows:

```
procedure LS1:
begin
 set $S = V$;
 generate a random initial assignment $\sigma$;
 while $S \neq \emptyset$ and $\sigma$ is not a solution do
   select $(i,j)$ such that $i \in S$, $j \neq \sigma_i$ and $c(i,j)$ is minimized;
   set $\sigma = \varDelta(\sigma,i,j)$ and $S = S - \{i\}$;
 endwhile
 output $\sigma$;
end.
```

LS1 is similar to Minton *et al.*'s algorithm [4] with the following exceptions:

1. Instead of arbitrarily picking a variable in conflict, we pick a variable-value pair which has the minimum number of conflicts, ties broken arbitrarily. We call this the *min-conflict* heuristic.
2. The algorithm terminates after $O(n)$ iterations since we do not allow variables to be reset more than once. Hence, the worst-case time complexity of LS1 in terms of number of comparisons is $O(n^2k)$.

We proceed to analyze the performance of LS1. Let $B(N,\theta)$ be a binomial random variable representing the number of successes in N independent Bernoulli trials when the probability of success in each trial is θ. The following lemma can be derived using Chernoff bounds [2] (equations (6) and (7)). A similar lemma was given in [3].

Lemma 1. *Let $X = B(N_1,\theta)$ and $Y = B(N_2,\theta)$ where $N_2 \leq N_1 \leq n$ and $\frac{\log n}{n} \leq \theta \leq 1$. Then, $Pr[X \leq Y] \leq 2\left(\frac{1}{n}\right)^{\left(\sqrt{N_1}-\sqrt{N_2}\right)^2/2.08n}$.*

Lemma 2. *Let x be an instance of R-CSP with n variables, domain size k, edge probability $q \geq \log n/n$, and consistency probability $0 \leq \lambda \leq \ln k/k$. Then x has one solution almost surely.*

Proof. The instance x has at least one solution, namely $\hat{\sigma}$ (see section 2). Consider any assignment $\sigma \neq \hat{\sigma}$. The probability that σ is a solution is ϕ^m (see Fact 1) because it must satisfy all constraints. Thus, the probability that there exists a solution other than $\hat{\sigma}$ is no more than $(k^n - 1)\phi^m \leq k^n\phi^m$. Since $q \geq \log n/n$, m is expected to be at least $\frac{n-1}{2}\log n$. Substituting ϕ and the bounds for m and λ, we get

$$k^n\phi^m \quad \leq \quad k^n\left[\frac{\ln k}{k}\left(1-\frac{1}{k^2}\right)+\frac{1}{k^2}\right]^{n\log n} \quad \leq \quad \frac{(\ln k+\frac{1}{k})^{n\log n}}{k^{n(\log n-1)}} \quad \ll \quad 1$$

for sufficiently large n. □

From now, consider an R-CSP instance which contains exactly one solution $\hat{\sigma}$. Observe that for each variable, only one value is correct while the other $k-1$ values are wrong. Each variable-value pair (i,j) is *correct* if j is the correct value for i. Our aim is to show that, if the initial assignment is *good*, (i.e. the number of variables that differ from $\hat{\sigma}$ is less than half the total), then LS1 will pick a correct (i,j) pair almost surely thus bringing us one step closer to the solution. Let D denote the Hamming distance between the initial assignment and $\hat{\sigma}$. Let the local search iterations be numbered $D, D-1, \ldots, 1$. Define a binomial random variable $X_d[i,j]$ for the value of the potential function $c(i,j)$ at iteration d, given that we have chosen correct pairs from iterations D to $d+1$.

Lemma 3. *For all $i \in S$ and $j \in K$, if (i,j) is correct then $X_d[i,j] = B(d, qp)$, else $X_d[i,j] = B(n-1, qp)$.*

Proof. Let σ be the current assignment at the beginning of iteration d. In σ, d variables have wrong values. If (i,j) is a correct pair, only these d variables can conflict with i under the assignment $\Delta(\sigma, i, j)$. Conversely, if (i,j) is wrong, then all $n-1$ variables can potentially conflict with i. Any variable i' will conflict with i iff there is an edge between i and i' and the value pair $(j, \sigma_{i'})$ is inconsistent. The probability that an edge exists between i and i' is q. By Fact 1, the probability that $(j, \sigma_{i'})$ is inconsistent is p. □

Let (i_{min}, j_{min}) denote the variable-value pair chosen by the min-conflict heuristic at iteration d and E_d, the event that (i_{min}, j_{min}) is a wrong pair.

Lemma 4. *If $qp \geq \frac{\log n}{n}$ then $Pr[E_d] \leq 2\left(\frac{1}{n}\right)^{\frac{d}{2.08n}(\sqrt{n-1}-\sqrt{d})^2}$.*

Proof. Since (i_{min}, j_{min}) is wrong, by the min-conflict heuristic, there exist d correct pairs examined at iteration d whose potential function values are no less than that of (i_{min}, j_{min}) (otherwise, one of those pairs would have been picked instead of the wrong pair). Therefore, the probability that E_d occurs is the probability that $X_d[i,j] \geq X_d[i_{min}, j_{min}]$, for all d correct pairs (i,j). Since the $d+1$ random variables are pairwise independent,

$$\begin{array}{rcll} \Pr[\mathrm{E}_d] & \leq & (\Pr[X_d[i_{min}, j_{min}] \leq X_d[i,j]])^d & \\ & \leq & (\Pr[B(n-1, qp) \leq B(d, qp)])^d & \text{(by Lemma 3)} \\ & \leq & 2\left(\frac{1}{n}\right)^{\frac{d}{2.08n}\left(\sqrt{n-1}-\sqrt{d}\right)^2} & \text{(by Lemma 1)} \end{array}$$

□

Putting Lemmas 2 and 4 together, we get the following result:

Theorem 2. *Let x be an instance of R-CSP with n variables, domain size k, consistency probability $0 \leq \lambda \leq \frac{\ln k}{k}$ and edge probability $q \geq \frac{\log n}{(1-\lambda)n}$. Then LS1 returns a solution for x almost surely if the initial assignment is good.*

Proof. By Lemma 2, x has exactly one solution almost surely. Suppose the initial assignment is good, i.e. $D \leq N$ ($N = \lfloor \frac{n-1}{2} \rfloor$). The probability that LS1 fails to return the solution is the probability that at least one event E_d occurs along the way, for some $2 \leq d \leq D$. Notice that E_1 cannot occur because the algorithm always correctly reset the last wrongly-set variable. Thus,

$$\begin{array}{rcl} \Pr[\text{LS1 fails}] & = & \Pr[\mathrm{E}_D] + \Pr[\mathrm{E}_{D-1} \mid \overline{\mathrm{E}_D}] + \cdots + \Pr[\mathrm{E}_2 \mid \wedge_{d=3}^{D} \overline{\mathrm{E}_d}] \\ & \leq & \Pr[\mathrm{E}_D] + \frac{\Pr[\mathrm{E}_{D-1}]}{1-\Pr[\mathrm{E}_D]} + \cdots + \frac{\Pr[\mathrm{E}_2]}{1-\sum_{t=3}^{D}\Pr[\mathrm{E}_t]} \\ & \leq & \sum_{d=2}^{D} \frac{e_d}{1-\sum_{t=d+1}^{D} e_t} \quad \text{(by Lemma 4)} \end{array}$$

where $e_d = 2\left(\frac{1}{n}\right)^{\frac{d}{2.08n}\left(\sqrt{n-1}-\sqrt{d}\right)^2}$. Consider the sequence $E = (e_2, \ldots, e_N)$. By simple calculus, one may verify that E is a bitonic sequence $e_2 \geq e_3 \geq \cdots \geq e_{(n-1)/4} \leq e_{(n-1)/4+1} \leq \cdots \leq e_N$. For sufficiently large $n \geq 70$, this sequence has the interesting property that

$$\sum_{t=3}^{N} e_t \leq e_2.$$

This may be hard to prove analytically, but a simple program can be written to verify this property. It follows that

$$\frac{e_d}{1-\sum_{t=d+1}^{D} e_t} \leq \frac{e_d}{1-e_2}, \quad \forall\, 2 \leq d \leq D.$$

Therefore,

$$\Pr[\text{LS1 fails}] \quad \leq \quad \frac{1}{1-e_2}\sum_{d=2}^{D} e_d \quad \leq \quad \frac{2e_2}{1-e_2}.$$

Since e_2 tends to 0 for sufficiently large n, the lemma follows. □

5 Iterative Local Search Algorithm LS2

The previous section shows that LS1 performs well when λ is reasonably small. When λ becomes large, our analysis of LS1 collapses because we cannot assume that instances have unique solutions. In this section, we turn our attention to iterative local search, i.e. performing local search iteratively with independent random initial assignments.

We propose the following iterative local search algorithm LS2. Let LS denote any local search which begins with a random assignment and performs hill-climbing (i.e. no downhill moves) to increase the number of satisfied constraints until it reaches a local optimum. Note that LS1 is not an example of LS. Execute LS for a maximum of $O(n^{k+1}\ln n)$ times, where n is the number of variables and k is the domain size.

Consider an R-CSP instance generated with edge probability q and consistency probability λ. We say that an assignment is *α-near* iff it satisfies at least αm constraints (recall m is total number of constraints). Let σ^* be the output computed by LS2. Given $0 \leq \alpha \leq 1$, what is the probability that σ^* is α-near? Clearly, if $\alpha \leq \phi$ (recall ϕ is the expected consistency probability), the probability is close to 1, since a random assignment would satisfy ϕm constraints. If $\alpha > \lambda$ however, it turns out that the probability is still close to 1 for a significantly large range of α. The reason (which we will show) is that the assignment obtained by one local search iteration is α-near with not-so-small probability. Hence, with enough independent random initial assignments, we can expect to find an α-near assignment almost surely.

We need the following theorem from analytic inequalities:

Proposition 1. *[5]. For all $x > 0$,*

$$\int_x^\infty e^{-\frac{z^2}{2}}dz \quad > \quad \frac{1}{2}\left(\sqrt{x^2+4}-x\right)e^{-\frac{x^2}{2}}.$$

Define $\pi(\alpha)$ to be the probability that one invocation of LS returns an α-near assignment. Let $c_0 = \frac{\alpha-\lambda}{\sqrt{\lambda(1-\lambda)}}$ and $a_0 = \frac{1}{2}\left(\sqrt{c_0^2qnk+4}-c_0\sqrt{qnk}\right)$.

Lemma 5. *For sufficiently large n, $\pi(\alpha) \geq \frac{a_0}{\sqrt{2\pi}} e^{-\frac{c_0^2 qnk}{2}}$.*

Proof. Let σ be the assignment obtained by LS. The probability that σ is *not* α-near is the probability that we randomly pick a variable i and it is incident on less than αm_i (m_i is degree of i) satisfied constraints. For counting purpose only, re-label all domain values so that each domain has a distinct set of values. Since σ is locally optimal, the total number of consistent value pairs occurring in those incident constraints, which is a binomial random variable $B(m_i k, \phi)$, must be less than $\alpha m_i k$. Therefore,

$$1 - \pi(\alpha) \quad = \quad \Pr\left[B(m_i k, \phi) < \alpha m_i k\right].$$

Since the instance is generated with parameters q and $\lambda \leq \phi$, this implies that

$$\pi(\alpha) \quad \geq \quad \Pr\left[B(qnk, \lambda) \geq \alpha qnk\right].$$

By the Central Limit Theorem, $B(qnk, \lambda)$ can be approximated closely by a Gaussian random variable Z with mean $qnk\lambda$ and variance $qnk\lambda(1-\lambda)$. Thus,

$$\pi(\alpha) \quad \approx \quad \Pr\left[Z \geq \alpha qnk\right] \quad = \quad \frac{1}{\sqrt{2\pi}} \int_{c_0\sqrt{qnk}}^{\infty} e^{-\frac{z^2}{2}} dz$$

and the lemma follows by Proposition 1. □

Lemma 6. *For sufficiently large n, if $q \geq \log n/n$ and $c_0 \leq \sqrt{2\ln 2}$, then $\pi(\alpha) \geq \frac{1}{\sqrt{2\pi}} n^{-(k+1)}$.*

Proof. For sufficiently large n,

$$a_0 \;=\; \frac{1}{2}\left(\sqrt{c_0^2 qnk + 4} - c_0\sqrt{qnk}\right) \geq n^{-1}.$$

Since $q \geq \log n/n$ and $c_0 \leq \sqrt{2\ln 2}$, by Lemma 5,

$$\pi(\alpha) \;>\; \frac{a_0}{\sqrt{2\pi}} e^{-\frac{c_0^2 qnk}{2}} \;\geq\; \frac{n^{-1}}{\sqrt{2\pi}} \cdot \left(e^{-\log n}\right)^{c_0^2 k/2} \;\geq\; \frac{1}{\sqrt{2\pi}} n^{-(k+1)}.$$

□

This non-trivial probability of success allows us to conclude that:

Theorem 3. *Let x be an instance of R-CSP with n variables, fixed domain size k, edge probability $\geq \log n/n$, and consistency probability λ. LS2 almost surely returns an α-near assignment after $O(n^{k+1}\ln n)$ iterations, where $0 \leq \alpha \leq \min(\lambda + 1.17\sqrt{\lambda(1-\lambda)}, 1)$.*

Proof. By Lemma 6, after $cn^{k+1}\ln n$ (constant c) independent iterations of LS, the probability of finding an α-near assignment is at least

$$1-\left(1-\frac{1}{\sqrt{2\pi}}n^{-(k+1)}\right)^{cn^{k+1}\ln n} \quad\geq\quad 1-e^{-\ln n} \quad=\quad 1-\frac{1}{n}$$

which tends to 1 for sufficiently large n. □

While the above theorem gives a good approximation result, our main concern is to find an exact solution. Substituting the upper bound of α by 1:

Corollary 1. *For* $0.43 \leq \lambda \leq 1$ *and edge probability* $\geq \log n/n$*, LS2 almost surely returns a solution of an R-CSP instance in polynomial time.*

Acknowledgements

I wish to thank O. Watanabe for stimulating discussions and P. van Beek for comments on an early version of this paper.

References

1. Noga Alon and Nabil Kahale. A spectral technique for coloring random 3-colorable graphs. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 346–355, 1994.
2. Torben Hagerup and Christine Rub. A guided tour of Chernoff bounds. *Inf. Process. Lett.*, 33:305–308, 1989.
3. Elias Koutsoupias and Christos H. Papadimitriou. On the greedy algorithm for satisfiability. *Inf. Process. Lett.*, 43:53–55, 1992.
4. Steve Minton, Mark D. Johnson, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58:161–205, 1992.
5. D. S. Mitrinovic. *Analytic Inequalities.* Springer Verlag, Heidelberg, 1970.
6. Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proc. Nat'l Conf. on Artif. Intell. (AAAI-92)*, pages 440–446, St Jose, CA, 1992.
7. Peter van Beek. On-line C-programs available at `ftp.cs.ualberta.ca` under directory `/pub/ai/csp`.
8. Peter van Beek and Rina Dechter. On the minimality and global consistency of row-convex constraint networks. *J. Assoc. Comput. Mach.*, 42(3):543–561, 1995.
9. Nobuhiro Yugami, Yuiko Ohta, and Hirotaka Hara. Improving repair-based constraint satisfaction methods by value propagation. In *Proc. Nat'l Conf. on Artif. Intell. (AAAI-94)*, Seattle, WA, 1994.