On the Average Case Complexity of Some P-complete Problems

Maria Serna **Fatos Xhafa**

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Mòdul C6 - Campus Nord
Jordi Girona Salgado, 1-3
08034-Barcelona, Spain

Technical Report **LSI-95-47-R**

Note: The results presented here were obtained previously to those in [TX96]. In fact this last result can be extended to the problems addresed in this paper, and we will consider it in the thesis.

On the Average Case Complexity of Some P-complete Problems *

Maria Serna† Fatos Xhafa†

Abstract

We show that some classical P-complete problems can be solved efficiently in average NC. The probabilistic model we consider is the sample space of input instances with the underlying distribution being the uniform one. The parallel algorithms use a polynomial number of processors and have expected time polylogarithmic in the instance size.

1 Introduction

Given a P-complete problem there is little hope of finding an efficient parallel algorithm to solve it, unless P=NC. Consider that a parallel algorithm is efficient if it finds feasible solutions to instances of the given problem in time polylogarithmic in instance size and uses a polynomial number of processors (see, e.g., [BDG95, GHR95]). Since the P-completeness is a worst case complexity result, it does not rule out the existence of parallel algorithms that although needing polynomial time in the worst case, perform efficiently for "almost all" instances. Finding such parallel algorithms is an alternative to cope with P-complete problems which consists in showing that the expected parallel time of such algorithm is polylogarithmic under a reasonable probabilistic model. This approach has been initiated for P-hard problems in [CRT87, CF90] and further considered in Díaz et al. [DSST94] for Circuit Value Problem. In [DSST94] the authors left open the possibility of using their model for other P-complete problems. It should be pointed out that their main interest was to give a parallel algorithm for CVP having an expected time polylogarithmically upper bounded. They do not consider the possibility for improvements on the upper bound.

In this paper our interest is, first, to extend the result of Díaz et al. for the the following classical P-complete problems: *Uniform Word*, *Unit Resolution*, *Path Systems* and *Generability*, and secondly, to give improved expected parallel time, i.e. to find parallel algorithms with better expected time bounds. Consider that a parallel algorithm is *efficient on average*, or is in average NC, if it uses a polynomial number of processors and finds the solution to the problem in polylogarithmic expected time, under an appropriate probabilistic model. The probabilistic model that we consider is the following: the sample space is the set of input descriptions, and the underlying distribution is the uniform one, that is, every object in the sample space is counted once. We present parallel algorithms for the problems mentioned above such that if the instances of the problems are chosen uniformly at random they have expected time that is polylogarithmic in the instance size. More precisely, we use Chernoff bounds (see, e.g., [HR90]) to show that such algorithms

*This research was supported by the ESPRIT BRA Program of the European Community under contract No. 7141, Project ALCOM II.

†Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Pau Gargallo 5, 08028-Barcelona.

have expected time $O(\log^3 n)$, with high probability, for large values of (instance size) n. Then, in order to provide improved expected bounds we take advantage of the particular structure of the problems at hand and show expected time $O(\log n)$.

Given a parallel algorithm to solve a P-complete problem another question of interest to ask is whether we can compute in advance, for a given instance, the time the algorithm lasts on this instance. Our motivation for this issue comes from the following observation: if we can compute quickly the time the algorithm lasts on instances then we could use the algorithm for the easy instances and decide whether to use it or not on the difficult ones. We address this problem for the proposed parallel algorithm for *Uniform Word*. We define a new problem, called *Extended Uniform Word*, in which we are given an instance of Uniform Word and we want to compute the *"depth"* of the given pair of terms in the set of axioms rather than to prove whether that pair of terms belongs to the set of axioms. We show that this problem is P-complete and furthermore that the problem of approximating it within any ratio $\varepsilon > 0$ is P-complete. Similar results can be found for the other problems addressed in [Ser90]. Regarding this result we will refer to ε-Π problem as well as logspace ε-gap reductions which are useful for proving parallel non-approximability results.

The methodology for the problems considered here is almost the same. First we show how to generate uniformly at random instances of the problem at hand and then exhibit a parallel algorithm that is in average NC. Since random generation of instances presents peculiarities for each problem, we do it for any case. On the other hand the analysis that proves the expected polylog time of the proposed algorithm is given complete for the Uniform Word and with simplified proofs for other cases.

Preliminaries

Average NC Class. Given a language L and a probability distribution p_n over words of of length n of L, it is said that L belongs to the class Average NC (with respect to the given distribution) if there is a parallel algorithm for L that uses a polynomial number of processors, runs in time $T(|x|)$, $x \in L$, and there exist constants k, γ such that, for strings of length n the following holds

$$\sum_{x \in L, |x|=n} T(x) p_n(x) \leq \gamma \log^k n.$$

Function Problem. Let Π be a given problem such that for any instance I there is a unique solution to I, and let $\Pi(I)$ denote the (unique) value corresponding to the solution. Clearly, $\Pi(I)$, as a function, is well defined. Π is called a function problem.

ε-Π **Problem.** Given a function problem Π, an instance I of Π and an ε, $\varepsilon \in (0,1)$, compute a value $V(I)$ such that $\varepsilon\Pi(I) \leq V(I) \leq \Pi(I)$.

Logspace ε-Gap Reduction [SG76, Ser91]. Given a decision problem Π, a function problem Π' and an ε, $\varepsilon \in (0,1)$, a logspace ε-gap reduction from Π to Π' is a pair of functions (f,g) computable in logspace such that (a) function f transforms instances of Π into instances of Π', (b) function g assigns a rational value $g(I)$ to instance I of Π and, finally, (c) if $\Pi(I) = 0$ then $\Pi(f(I)) < g(I)$, otherwise $\Pi(f(I)) \geq g(I)/\varepsilon$.

Logspace ε-gap reductions are useful to prove parallel non-approximablity results as stated by the following result of Serna [Ser90]. Let Π be a P-complete problem, Π' a function problem and $\varepsilon \in (0,1)$. If there is a logspace ε-gap reduction from Π to Π' then ε-Π' is P-complete.

2 The Uniform Word Problem

The Uniform Word Problem (UWP) for finitely presented algebras is defined as follows [Koz77]. Let $\langle M, \texttt{arity}\rangle$ be a ranked alphabet where M is a finite set of *symbols* and arity is a mapping $\texttt{arity} : M \to N$ that assigns a non-negative integer to each symbol. The alphabet M is partitioned into two sets: $\mathcal{G} = \{a \in M \mid \texttt{arity}(a) = 0\}$ and $\mathcal{O} = \{\theta \in M \mid \texttt{arity}(\theta) > 0\}$. The elements of $\mathcal{G}$ are called *generator symbols* and those of $\mathcal{O}$ *operator symbols*. Further, the set of *terms* over M is defined as follows:

- all elements of $\mathcal{G}$ are terms;
- if θ is m-ary and $x_1, x_2, \ldots, x_m$ are terms then $\theta(x_1, x_2, \ldots, x_m)$ is a term.

Let $\mathcal{T}$ denote the set of terms. Then, a binary relation $\approx$ in $\mathcal{T}$ is a congruence provided that: (a) $\approx$ is an equivalence relation, and (b) if $\theta \in \mathcal{O}$ is m-ary and x_i, y_i, $i = 1, \ldots, m$ are terms such that $x_i \approx y_i$ then $\theta(x_1, x_2, \ldots, x_m) \approx \theta(y_1, y_2, \ldots, y_m)$. Next, a set of *axioms* Γ is a set of unordered pairs of terms. There is possible to define on $\mathcal{T}$ an equivalence relation that satisfy all the axioms of Γ. For that we define firstly the binary relation: $\forall x, y \in \mathcal{T}$, $x \approx y$ iff $\{x, y\} \in \Gamma$, which satisfies all the axioms of Γ. Having $\approx$ we define $\equiv_\Gamma$ to be the smallest congruence in $\mathcal{T}$.

The *Uniform Word Problem* is: Given Γ, $\mathcal{T}$ and a pair of terms $\{x, y\}$ decide whether $\{x, y\}$ belongs to the closure of $\equiv_\Gamma$.

2.1 Generating Instances

Let $\langle M, \texttt{arity}\rangle$ be a given ranked alphabet. Let us fix a positive integer n. We will generate a set Γ of n axioms and an additional term to be checked, as given below. We will suppose that there is a *fictitious* operator of arity 1 in $\mathcal{O}$ such that whenever it is chosen means that a symbol from $\mathcal{G}$ is to be chosen. This is to assure that terms that are symbols from $\mathcal{G}$ can be generated as well.

`procedure instance`

> $\mathcal{A} := \mathcal{G}$;
> <u>for</u> i:=1 to $n+1$ <u>do</u>
> choose uniformly at random two operators $\theta, \phi \in \mathcal{O}$;
> choose uniformly at random from $\mathcal{A}$, $\texttt{arity}(\theta)$ elements $g_1, g_2, \ldots, g_{\texttt{arity}(\theta)}$ corresp. to θ and $\texttt{arity}(\phi)$ elements $e_1, e_2, \ldots, e_{\texttt{arity}(\phi)}$ corresp. to ϕ;
> generate the pair $\{\theta(g_1, g_2, \ldots, g_{\texttt{arity}(\theta)}), \phi(e_1, e_2, \ldots, e_{\texttt{arity}(\phi)})\}$;
> update $\mathcal{A} := \mathcal{A} \cup \{\theta(g_1, g_2, \ldots, g_{\texttt{arity}(\theta)}), \phi(e_1, e_2, \ldots, e_{\texttt{arity}(\phi)}\}$
> <u>end</u>

Notice that we generate a pair of terms each step and the terms of the next one are generated from the set of all the previous terms. This procedure generates $n+1$ pairs of terms. The first n terms will be the *axioms* and the last one is the term $\{x, y\}$ we want to check. Clearly, the procedure `instance` generates any term, so we cover all the instances, and moreover, the probability distribution over the instances is uniform. The encoding of an instance of UWP is then the sequence of axioms followed by the term to be checked.

2.2 The Deduction "Tree" Algorithm and Its Expected Time

We describe an algorithm which inputs an encoding of an instance of UWP consisting of a set Γ of n axioms and a pair of terms $\{x, y\}$, and decides whether $\{x, y\}$ belongs to the closure of Γ.

First Step: Consider all pairs $\{u, v\}$, $\{v, w\}$ of Γ and all pairs $\{r, z\}$, $\{s, t\}$ with $\{z, s\} \in \Gamma$, and apply on them a deduction step accordingly to the rules:

$\mathcal{R}_1$. From pairs $u \equiv_\Gamma v$ and $v \equiv_\Gamma z$ deduce the pair $u \equiv_\Gamma z$;

$\mathcal{R}_2$. From pairs $r \equiv_\Gamma z$, $s \equiv_\Gamma q$ with $\{z, s\} \in \Gamma$ deduce $r \equiv_\Gamma q$.

General Step: Let A_t be the set of terms which belong to the closure of Γ until step t. Find all the pairs of A_t as in First Step, and all the pairs $\{\alpha, \beta\}$, $\{\delta, \varepsilon\}$ where $\beta = \theta(x_1, x_2, \ldots, x_m)$ and $\delta = \theta(y_1, y_2, \ldots, y_m)$ with $x_i \equiv_\Gamma y_i$. Then, apply on them the deduction rules $\mathcal{R}_1$, $\mathcal{R}_2$ and the following:

$\mathcal{R}_3$. From the pairs $(u, \theta(x_1, \ldots, x_m))$ and $(\theta(y_1, \ldots, y_m), z)$ with $x_1 \equiv_\Gamma y_1, \ldots, x_m \equiv_\Gamma y_m$ deduce (u, z).

Repeat until no deductions are possible.

Notice that this algorithm constructs a *deduction "tree."* The axioms are the leaves, and the depth of any node is (defined to be) one greater than the maximum depth of the nodes from which it was deduced. Let d be the expected depth of the closure of Γ. We claim that $d = \text{polylog}(n)$.

Theorem 1 *The expected time to decide whether a given pair $\{x, y\}$ belongs the closure of a set of axioms Γ considered under the uniform distribution is at most $\gamma \log^3 n$ for some constant $\gamma > 1$ and sufficiently large n, with high probability.*

Proof. Here is the idea of the proof. The time needed by the deduction algorithm is closely related to the depth of the deduction tree. We will find an recurrence equation on this last which enables us to derive a polylogarithmic upper bound. More precisely, let us suppose that $n/\log n$ pairs of axioms are added to Γ and let us estimate how much increases the depth of the deduction tree. We denote this quantity by $\Delta(n)$. We prove first the following fact.

Fact 1 *There exist constants δ and α such that $\Delta(n) \leq \alpha \log n$ with probability at least $1 - n^{-\delta}$.*

Proof (of Fact 1). Indeed, in the worst case we can suppose that all the n pairs are at the top-level d. Let K_d be this current highest level and K_{d+1} be the set of terms deduced at step $d+1$ and $|K_{d+1}^{\max}|$ be the maximum size of K_{d+1}. More generally, let $|K_{d+i}^{\max}|$ denote the maximum size of K_{d+i} corresponding to step i. Based on Chernoff bonds [HR90] we prove the following fact from which we will derive the claim.

Fact 2 $|K_{d+i}^{\max}| = \Theta(n/\log^i n)$ *with high probability (at least $1 - \exp(-\varepsilon/2 \cdot n/\log^i n)$).*

Proof (of Fact 2). We prove it for then case $i = 1$ (i.e. when passing from level K_{d+1} to K_{d+2}), the general case is similar.

The level K_{d+2} will be produced if there will occur a deduction. Let p_1 be this probability. We can easily seen that this probability is at least one half, $p_1 \geq 2 \cdot (n/\log n)/(n + \log n) \geq 1/2$. That means, we can bound the process by an $n/\log n$ Bernoulli trials with probability $1/2$. Then, applying Chernoff bounds [HR90], we have that $|K_{d+1}^{\max}| = \Theta(n/\log n)$ with high probability, i.e. at least $1 - \exp(-\varepsilon/2 \cdot n/\log n))$, for any $\varepsilon \in (0, 1)$.

At the second step the only pairs that will cause deductions are those of K_{d+1}. Let $\mathcal{E}_1$ be the event:

$$\mathcal{E}_1 = \{|K_{d+1}^{\max}| = \Theta(\frac{n}{\log n}), \text{ when } n/\log n \text{ axioms are added}\}.$$

We calculate the probability p_2 that a deduction occurs at the step $d+2$, conditioning on $\mathcal{E}_1$. Using the same idea as above, a new deduction will occur with probability $p_2 = 2 \cdot |K_{d+1}|/n$ and since $|K_{d+1}^{\max}|/2 \le |K_{d+1}| \le |K_{d+1}^{\max}|$, we obtain $p_2 = \Theta(1/\log n)$. Applying again Chernoff bounds we conclude that $|K_{d+2}^{\max}| = \Theta(n/\log^2 n)$, with probability at least $1 - \exp(-\varepsilon^2/2 \cdot n/\log^2 n)$, $\varepsilon \in (0,1)$. □

Next, let $k = \Theta(\log n/\log\log n)$ and let us consider the events $\mathcal{E}_i$, $i = 1, \ldots k$,

$$\mathcal{E}_i = \{|K_{d+i}^{\max}| = \Theta(n/\log^i n), \text{ when } n/\log n \text{ axioms are added}\}.$$

Now we prove that $\Pr(\mathcal{E}_1 \cdot \mathcal{E}_2 \cdots \mathcal{E}_k) \ge 1 - n^{-\delta}$, for some $\delta > 0$. Indeed, from Fact 2, we have

$$\begin{aligned} &\Pr(\mathcal{E}_1 \cdot \mathcal{E}_2 \cdots \mathcal{E}_k) = \\ &\Pr(\mathcal{E}_1)\Pr(\mathcal{E}_2/\mathcal{E}_1) \cdots \Pr(\mathcal{E}_k/\mathcal{E}_1 \cdots \mathcal{E}_{k-1}) \ge \\ &\prod_{i=1}^{k} (1 - \exp(-\varepsilon/2 \cdot n/\log^i n)) \ge 1 - n^{-\delta}. \end{aligned}$$

After $k = \Theta(\log n/\log\log n)$ steps there will be left $a \log n$ axioms, for some constant a, which can contribute at most $a \log n$ additional levels. Therefore, we conclude that, with high probability, the increase $\Delta(n)$ when there are added $n/\log n$ new axioms is $\Delta(n) \le k + a \log n \le \alpha \log n$, for some constant α. □

Finally, the expected depth d_n for the whole closure of $n + n/\log n$ axioms satisfies the recursion

$$E[d_n] + E[\Delta(n)] = E[d_{n+n/\log n}]$$

and in combination with Fact 1 we obtain

$$E[d_{n+n/\log n}] \le E[d_n] + \alpha \log n.$$

By solving this recursion the theorem follows. □

2.3 An Algorithm which Gives Better Bound

The result on the deduction algorithm given in the previous subsection is sufficient to state that UWP is in the class Average NC. However, algorithms with better bounds are possible. Here we consider the algorithm given in [Koz77] and show a better bound on the expected deduction time. We say that x derives y in one step, denoted $x \to y$, if there is an axiom $z \equiv w$ in Γ and an occurrence of z in x such that when that occurrence of z is replaced by w then the result is y. Let $\stackrel{*}{\to}$ be the transitive closure of $\to$. Now, in terms of $\stackrel{*}{\to}$ we have that

$$x \equiv_\Gamma y \quad \text{if there exist a proof} \quad x \stackrel{*}{\to} x_1 \stackrel{*}{\to} \cdots \stackrel{*}{\to} x_l \stackrel{*}{\to} y.$$

We want to estimate the expected time of the proofs needed to decide whether $\{x, y\}$ belongs to the closure of Γ. We claim that the expected length of proofs is bounded from above by $(4e+1)\log n$, asymptotically.

Theorem 2 *For sufficiently large n, the expected length of proofs that decide whether a given pair of elements $\{x,y\}$ belongs to the closure of a set of n axioms Γ is upper bounded by $(4e+1)\log n$, with high probability.*

Proof. Let T be the set of all the proofs, with a total of $n = 2^k$ terms. We can consider a proof as a directed path from x to y. Different proofs may have common vertices (terms) other than x and y. So, let Q be the corresponding graph of T. Suppose that the expected length proof is d, therefore there is a proof of length d,

$$x \xrightarrow{*} \alpha_1 \xrightarrow{*} \cdots \xrightarrow{*} \alpha_r \xrightarrow{*} y \tag{1}$$

where the length of the partial proof $\alpha_i \xrightarrow{*} \alpha_{i+1}$ $(\alpha_0 = x,\ \alpha_{r+1} = y)$ is l_i, $\sum_{i=0}^{r} l_i = d$. Let us estimate $\Pr(\texttt{length}(\texttt{proof}) = d)$. We must consider all proofs of length d, where the length of the partial proofs $l_0, \ldots, l_{r-1}$ vary over $m_0, \ldots, m_{r-1}$ with $\sum_{i=0}^{r-1} m_i = d$. Therefore

$$\Pr(\texttt{length}(\texttt{proof}) = d) = \sum_{m_0,\ldots,m_{r-1}} \Pr(l_0 = m_0, \ldots, l_{r-1} = m_{r-1}). \tag{2}$$

We consider a topological order on Q (in which the terms are deduced) and classify the vertices of Q in the sets $N_i = \{T_j \ :\ 2^i \leq j < 2^{i+1}\}$ for $i = 0, \ldots, k-1$ where $T_0 = x, \ldots, T_{n-1} = y$. In order to estimate the last sum in (2) we note that the first term α_1 in (1) may belong to one of the sets N_i so, we sum over i and over l, where l is the number of partial proofs of the whole proof $(0 \leq l \leq d, l = 0$ when $x = y$, and $l = d$ when all the partial proofs are of one step). A sequence of l partial proofs is viewed as a strict increasing function $\phi : \{0, \ldots, l\} \rightarrow \{0, \ldots, k-1\}$ since such a function defines exactly l partial proofs with length $l_j > 0$ for $j \in range(\phi)$. Thus (2) is written as

$$\sum_1 \sum_2 \sum_3 \Pr(\alpha_1 \in N_i \textit{ and } (l_j > 0 \ \text{ if } j \in range(\phi)) \text{ and } \ \text{all } l_{\phi(j)} = m_j)$$

where the first sum is over all $0 \leq i < k$, $0 \leq l < d$, the second one over $\phi \in M_l$ (M_l is the set of functions ϕ as above) and the third one over sequences $m_1, \ldots, m_l$ with $\sum m_i = d$. Now, we can write

$$E(\texttt{length}(\texttt{proof})) = \sum_d k \cdot \Pr(\texttt{length}(\texttt{proof}) = d) = E_1 + E_2$$

where

$$E_1 = \sum_{d \leq (4e+1)k} d \cdot \Pr(\texttt{length}(\texttt{proof}) = d), \quad E_2 = \sum_{d \geq (4e+1)k} d \cdot \Pr(\texttt{length}(\texttt{proof}) = d).$$

The following hold [Tat95]:

Fact 3 *(a) $E_1 = O((4e+1)\log n)$, and (b) E_2 tends to 0 when k tends to ∞.*

From Fact 3 the theorem follows. □

2.4 Extended Uniform Word (EUW)

Recall that, given an instance $(\Gamma, T, \{x, y\})$ of UWP, the axiom $\{x, y\}$ belongs to the closure of Γ if there is a proof $x \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_l \rightarrow y$. Let us define

$$R(\Gamma_i) = \{w \mid \exists\ u \in \Gamma_{i-1}\ ,\ u \rightarrow w\}$$

where $R(\Gamma_0) = \Gamma$ and $\Gamma_i = R(\Gamma_{i-1})$, $i \geq 1$. Let $L(\Gamma)$ denote the length of the longest proof, i.e.,

$$L(\Gamma) = \min\ \{k \mid \Gamma_k = \Gamma_{k+1}\}.$$

We define the *depth* of a pair $\{x,y\}$ in a set of axioms Γ as

$$D(\Gamma,\{x,y\}) = \begin{cases} \max\{k \mid \{x,y\} \in \Gamma_k \wedge 0 \leq k \leq L(\Gamma)\}, & \text{if} \quad x \equiv_\Gamma y, \\ L(\Gamma), & \text{otherwise.} \end{cases}$$

The Extended Uniform Word (EUW) is: Given an instance $(\Gamma, T, \{x,y\})$, compute the depth $D(\Gamma,\{x,y\})$.

We prove that ε-EUW is P-complete, i.e. EUW problem is non-approximable in NC, unless P=NC. This result is obtained by showing a logspace ε-gap reduction [SG76, Ser91].

Proposition 1 *ε-EUW is P-complete.*

Proof. We show a logspace ε-gap reduction from UWP to EUW. Given a set of m axioms Γ, we let $l = \lceil m^2/\varepsilon \rceil$, for some $\varepsilon \in (0,1)$ and construct the following set of axioms

$$\Gamma' = \Gamma \cup \{\{x,x_1\},\{x_1,x_2\},\ldots,\{x_l,y\}\}.$$

For the new instance we have:

- if the axiom $\{x,y\}$ is in the closure of Γ then $D(\Gamma,\{x,y\}) = D(\Gamma') < m^2$
- if the axiom $\{x,y\}$ is not in the closure of Γ then $D(\Gamma') = l = \lceil m^2/\varepsilon \rceil$

implying that the above reduction is a logspace ε-gap reduction. The proposition thus follows. □

3 Unit Resolution

In any instance of Unit Resolution [JL77] we are given a set of clauses $\mathcal{F}$ in CNF, a clause C, and we want to determine whether C belongs to the closure of $\mathcal{F}$ under unit resolution. When $C = \Box$ the problem is to decide whether we can derive a contradiction.

Let $V = \{x_1,\ldots,x_n\}$ be a set of literals i.e. propositional variables or their negations and suppose that in V appears either a literal or its negation but not both of them. We will generate uniformly at random instances of 3-Unit Resolution whose clauses have at most three literals. Let $L_1 = \{x_{i_1},\ldots,x_{i_k}\}$ be a subset of literals of V for some chosen fixed $k \leq n$, where x_{i_j} , $j = 1,\ldots,k$ are chosen uniformly at random from V. These literals will serve to generate the clauses of any instance. Let L_t be the set of literals which appear in the instance at step t, $t \geq 1$. In the step $t+1$, we do the following:

- choose uniformly at random $x_i, x_j \in L_t$ and $x_l \in V$;
- with probability one half generate the clause $x_l + \neg x_i + \neg x_j$ and with the same probability generate two clauses: $x_l + \neg x_i$ and $x_l + \neg x_j$;
- Set $L_t := L_t \cup \{x_l\}$.

Repeat the process until N clauses are generated.

This model generates a sample space $\mathcal{C}(k, N, 1/2)$ of instances with N clauses. It is clear that in such manner we generate all the instances since clauses of length one (literals), of length two and three are generated and moreover, we can easily see that the probability distribution over the instances is the uniform one.

3.1 The Algorithm and Its Expected Resolution Time

Let $\mathcal{F} = \{C_1, \ldots, C_m\}$ be the set of clauses in CNF. We define in $\mathcal{F}$, a unit step resolution as follows: whenever we have clauses x and $\neg x + C$, deduce C. We say that C is obtained by a unit step resolution over $(x, \neg x + C)$. More generally, a clause C is obtained by an m-step unit resolution iff there exist a sequence

$$U_1, U_2, \ldots, U_m \equiv C \tag{3}$$

where U_j is (x_{i_j}, C_j). In other terms, to achieve C we start from a pair (x_{i_1}, C_1) and apply step by step the unit resolution. So, the sequence (3) can be written as:

$$(x_{i_1}, C_1),\ (x_{i_2}, C_2), \ldots, (x_{i_j}, C_j), \ldots,\ (x_{i_m}, C_m) \equiv\ C$$

where C_{j+1} is the result of unit step resolution over (x_{i_j}, C_j). Further, consider a clause *saturated* iff it is not possible to reduce it anymore by a unit step resolution. In general, a clause is possible to be reduced from different sequences of unit step resolution, which of them beginning with a different clause. So we can define *the depth of a clause* C to be the longest sequence from which C is reduced and *the depth of* $\mathcal{F}$ to be the maximum of depth of the clauses of $\mathcal{F}$:

$$D(\mathcal{F}) = \max_{C \in \mathcal{F}} \{D(C, \mathcal{F})\}.$$

Now, given an instance $\mathcal{F}$ of clauses, in order to decide in parallel whether the clause C belongs to the closure of $\mathcal{F}$ the algorithm proceeds as follows: detect and perform level by level all the unit step resolution, where each level is done in parallel. Here a level should be understood as the set of clauses of the same depth. Of course, the expected running time of this algorithm depends on the depth of the closure of $\mathcal{F}$. We prove that the expected depth of the closure of $\mathcal{F}$ is polylog in the number of the clauses.

Theorem 3 *For sufficiently large N, the expected resolution time to decide whether a given clause C belongs to the closure of a set of clauses $\mathcal{F}$ in CNF is bounded by $(4e + 1)\log N$, with high probability.*

Proof. Note that we have an order between the clauses with respect to their depth. Suppose that the closure of $\mathcal{F}$ has $N = 2^k$ clauses. Consider as an input coding for the algorithm the sequence $C_1, \ldots, C_N$ where C_j has no predecessor or two predecessors $C_i = x_r$ and C_l with $i, l < j$ from which it was deduced. Partition the clauses into the following sets:

$$T_i = \{C_j \mid 2^i \leq j < 2^{i+1}\}.$$

Now, we want to estimate the expected resolution time needed to deduce C. Let us suppose that it equals d and let $U_1, U_2, \ldots, U_d \equiv C$ be a resolution sequence of length d which is a subsequence of $\langle C_1, \ldots, C_N \rangle$. Put $L_i = \{H_j : 1 \leq j \leq d\ ,\ H_j \in T_i\}$ and let $l_i = |L_i|$. We have that $\sum_{0 \leq j \leq k} l_j = d$. Then,

$$\Pr(depth(F) = d) = \sum_{m_0, \ldots, m_{k-1}} \Pr(l_0 = m_0, \ldots, l_{k-1} = m_{k-1}).$$

Note that the clause C_1 may belong to any T_i and that a unit resolution sequence can be viewed as an increasing function of $\{1, \ldots, l\}$ to $\{1, \ldots, k\}$. Thus, we can write the above sum as

$$\sum_1 \sum_2 \sum_3 \Pr(C_1 \in T_i \ and\ (l_j > 0 \ \text{ if } j \in range(\phi)) \text{ and } \text{ all } l_{\phi(j)} = m_j),$$

where the first sum is over $1 \leq i, l \leq k$, the second over $\phi \in M_l$ defined as above, and the third over $m_1, \ldots, m_l$. The rest of the proof follows that of Uniform Word. $\square$

4 Solvable Path Systems

A path system [Coo85] is defined as a quadruple $P = \langle X, R, S, T\rangle$, where X is a finite set, $S \subseteq X$, $T \subseteq X$ and $R \subseteq X \times X \times X$. An element x is called accessible iff x is in T or there are $y, z \in X$ such that $(x, y, z) \in R$ and both y and z are accessible. *Path Accessible Problem* is: Given a path system $P = \langle X, R, S, T\rangle$, determine whether there is an accessible element in S.

We *generate* instances of this problem as follows. Let $X = \{x_1, \ldots, x_n\}$. Generate uniformly at random two subsets S, T of X by choosing first their cardinalities k and l and then their elements from X. Let $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_l\}$. And now, generate the relation R:

- choose uniformly at random two elements $y, z \in X$;
- choose uniformly at random an element $x \in X$;
- set (x, y, z) in R.

Repeat this process until N elements of R are generated.

Clearly, every relation R in $X \times X \times X$ is generated and furthermore the probability distribution over them is uniform. The encoding of an instance is the sequence of elements of T and the triples of R. A parallel algorithm for a given instance provides by levels as follows: at level 0 the set of accessible elements is T. Find all the pairs (y, z) of T and the elements $x \in X$ such that $(x, y, z) \in R$. The elements x's found in such way are accessible. Let A_t be the set of accessible elements until step t. In the next step find all the pairs (y, z) of A_t and the elements $x \in X$ such that $(x, y, z) \in R$. Repeat this process until no accessible elements are left. Let N be the total number of accessible elements. Note that this algorithm provides a natural oder between the accessible elements. The executed time is the length of the longest sequence from any element of T to any accessible element. We can estimate the expected time of this algorithm as in the cases before. Here we formulate only the theorem which states the result of that analysis.

Theorem 4 *For sufficiently large N the expected time to decide whether a given set S has some accessible element is bounded by $(4e + 1)\log N$, with high probability.*

5 Generability Problem

This problem is defined as follows [JL77]: Given $\langle X, T, x, \bullet\rangle$ where X is a set, T a subset of X, x is an element of X and $\bullet$ a binary (commutative) operation, decide whether x belongs to the closure of T.

Generating Instances. Let $X = \{x_1, \ldots, x_m\}$ be a given set of m elements. Generate firstly, uniformly at random a set T of k elements from X, $T = \{t_1, \ldots, t_k\}$ and the element x to be checked. Further, we generate the table of the binary operation $\bullet$ in X by the following procedure:

`procedure table`

tab:= $\emptyset$;
<u>`for`</u> i:=1 to m^2 <u>`do`</u>
    choose uniformly at random $x \in X$ and $y \in X$;
    <u>`if`</u> $(x, y) \notin$ tab <u>`then`</u>
        choose uniformly at random $z \in X$;

set $x \bullet y = z$ and tab := tab $\cup \{(x, y)\}$;

<u>end</u>

Notice that the binary operation $\bullet$ given by `table` is defined correctly. We consider the pair (x, y) unordered so this operation is commutative. Furthermore, all the tables that define such operation are generated and the probability distribution over them is uniform.

The algorithm to find the closure of T proceeds as follows. Let $closure(T) := T$. At first step, we find all the pairs $(x, y) \in T$ such that $z = x \bullet y$, $z \notin T$ and set $closure(T) := closure(T) \cup \{z\}$. At step t, find all the pairs $(x, y) \in closure(T)$ such that $z = x \bullet y$, $z \notin closure(T)$ and set $closure(T) := closure(T) \cup \{z\}$. Repeat until no new elements can be added. Note that we have an order between the elements as they were inserted in $closure(T)$. We can define the depth of an element z in the $closure(T)$ as one greater than the depth of x and y from which it was obtained, and the $depth(closure(T))$ as the maximum of the depth of its elements. The maximum depth corresponds to the longest sequence beginning with an element of T to any element of the closure. Let $N = 2^k$ be the number of the elements of the closure. We can analyze the *expected* depth of the $closure(T)$ in terms of N as in the cases above, therefore we can formulate the analogue theorem for Generability.

Theorem 5 *For sufficiently large N the expected time to decide whether a given element $x \in X$ belongs to the closure of a given subset T of X is bounded by $(4e+1)\log N$ with high probability.*

Acknowledgment

Fatos thanks Josep Díaz for support from the ALCOM Project while conducting this research. We thank Ray Greenlaw for many helpful comments that significantly improved the paper.

References

[BDG95] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer Verlag, 1995.

[CF90] N. Calkin and A. Frieze. Probabilistic Analysis of a Parallel Algorithm for Finding a Maximal Independent Set. *Random Structures and Algorithms*, 1:39–50, 1990.

[Coo85] S.A. Cook. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control*, 64:2–22, 1985.

[CRT87] D. Copersmith, P. Raghavan, and M. Tompa. Parallel Graph Algorithms that are Efficients in Average. In *Proceedings of 28th IEEE Symposium on Foundations of Computer Science*, pages 260–269, 1987.

[DSST94] J. Díaz, M. Serna, P. Spirakis, and J. Torán. On the Expected Depth of Boolean Circuits. Technical Report LSI-94-7-R, Universitat Politècnica de Catalunya, Dept. de LSI, 1994.

[GHR95] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory.* Oxford University Press, 1995.

[HR90] T. Hagerup and C. Rub. A Guided Tour to Chernoff Bounds. *Information Processing Letters*, 37:305–308, 1990.

[JL77] N.D. Jones and T. Laaser. Complete Problems for Deterministic Polynomial Time. *Theoritical Computer Science*, 3:105–117, 1977.

[Koz77] D. Kozen. Complexity of Finitely Presented Algebras. *Proceedings of 9th ACM STOC*, pages 164–167, 1977.

[Ser90] M. Serna. *The Parallel APproximbility of P-complete Problems.* PhD thesis, Universitat Politècnica de Catalunya, 1990.

[Ser91] M. Serna. Approximating Linear Programming is Logspace Complete for P. *Information Processing Letters*, 37:233–236, 1991.

[SG76] S. Sahni and T. Gonzalez. P-complete Approximation Problems. *Journal of ACM*, 23:555–565, 1976.

[Tat95] T. Tatsuie. Bounds on the Depth of Circuits. Manuscript, 1995.

[TX96] T. Tsukiji and F. Xhafa. On the Expected Depth of Randomly Generated Circuits. In *Proceedings of Fourth European Symposium on Algorithms*, Lecture Notes in Computer Science, pages 208–220. Springer-Verlag, 1996.