Theoretical Informatics and Applications ITA, Vol. 33, N° 1, 1999, p. 33–45
Informatique Théorique et Applications

ON THE AVERAGE CASE COMPLEXITY OF SOME P-COMPLETE PROBLEMS [*]

Maria Serna[1] and Fatos Xhafa[1]

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 descriptions of the problem with the underlying distribution being the uniform one. We present parallel algorithms that use a polynomial number of processors and have expected time upper bounded by $(e \ln 4 + o(1)) \log n$, asymptotically with high probability, where n is the instance size.

Résumé. Nous montrons que quelques problèmes classiques qui sont P-complets peuvent être résolus efficacement en *average* NC. Le modèle probabiliste que nous considérons est l'espace de descriptions des entrées du problème sous la distribution uniforme. Nous présentons des algorithmes parallèles qui utilisent un nombre polynomial de processeurs dont leur temps espéré est majoré par $(e \ln 4 + o(1)) \log n$, assymptotiquement avec haute probabilité, où n est la taille de l'entrée.

1. Introduction

An important topic of the theory of parallel computations is to study the possibility that certain problems are *difficult* to parallelize. As a matter of fact, hundreds of problems are known to resist finding *efficient* parallel algorithms. The study of this behavior led to the theory of P-complete problems, initiated with the work of Cook [3] (see also [8,11]). A given problem is called P-complete if there is a polynomial time algorithm that solves it, and any other problem in the class P of problems solvable in polynomial time, considered as a language, can be *logspace* reduced to the problem [11]. Many important problems have been shown P-complete, evidencing that they all appear to be inherently sequential.

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

[1] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord – C6, Jordi Girona Salgado 1-3, 08034 Barcelona, Spain;
e-mail: `fatos@lsi.upc.es`

(For more on the P-completeness theory, we suggest to the reader the book by Greenlaw *et al.* [6].)

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.* [1, 6]). 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 method to cope with P-complete problems. This approach has been initiated for P-hard problems in Calkin and Frieze [4], Coppersmith *et al.* [2] and further considered in Díaz *et al.* [5] for Circuit Value Problem (CVP). It should be pointed out that in [5] their main goal 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 a recent paper, Tsukiji and Xhafa [16] proved a structural result on the depth of circuits, namely the expected depth of circuits of fixed fan-in f, unbounded fan-out and n gates under the uniform probabilistic model is $ef \ln n$, asympotically with high probability. This was a strong probability law from which, as a side effect, the average NC complexity of CVP was completely resolved.

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 expected time bounds better than those of Díaz *et al.* 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 combinatorial arguments to show that under the uniform distribution of instances, our algorithms have expected time upper bounded by $(e \ln 4 + o(1)) \log n$, asymptotically with high probability, where n is the instance size. The bound of [16] for the problems considered here would require a deeper analysis which we currently lack.

Given a parallel algorithm to solve a P-complete problem another question of interest is whether we can compute beforehand, for a given instance, the time the algorithm runs on this instance. Our motivation for this issue comes from the following observation: if we can quickly compute the time the algorithm runs 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. We show that this problem is P-complete and furthermore that the problem of approximating it within any $\varepsilon > 0$ is P-complete. Similar results can be found for other problems addressed in [12]. Regarding this result we will refer to ε-Π problems as well as logspace ε-gap reductions [13,14] which are helpful 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 for it 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 upper bound on the expected time of the proposed algorithm is given in full detail for the Uniform Word and without proof for other cases.

1.1. Preliminaries

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 [13,14]. Given a decision problem Π, a function problem Π' and an $\varepsilon \in (0, 1)$, 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 result of Serna [12]. 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 was among the first problems shown to be P-complete. Kozen [10] gave a polynomial time algorithm for the problem and provided a logspace reduction from Monotone Circuit Value Problem.

The Uniform Word Problem is defined as follows. Let $\langle M, \texttt{arity}\rangle$ be a ranked alphabet where M is a finite set of *symbols* and $\texttt{arity}$: $M \to N$ is a mapping 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. It is possible to define on $\mathcal{T}$ an equivalence relation that satisfy all the axioms of Γ. For that we first define 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. Extended uniform word

We define a new problem, called Extended Uniform Word (EUW), 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 closure of the set of axioms rather than to prove whether that pair of terms belongs to the closure. Given an instance $(\Gamma, \mathcal{T}, \{x, y\})$ of UWP, the axiom $\{x, y\}$ belongs to the closure of Γ if there is a proof $x \to x_1 \to x_2 \to \cdots \to x_l \to y$. Let us define

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

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. That is,

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

We define the *depth* of a pair $\{x, y\}$ in the 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 $(\Gamma, \mathcal{T}, \{x, y\})$, compute the depth $D(\Gamma, \{x, y\})$.

We prove that ε-EUW is P-complete for any value of ε, $\varepsilon \in (0, 1)$, *i.e.* EUW problem is non-approximable in NC, unless P = NC. This result is obtained by providing a logspace ε-gap reduction.

Proposition 1. *ε-EUW is P-complete.*

Proof. We present 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

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

For the new instance we have:

- if the axiom $\{x, y\}$ belongs to the closure of Γ then

$$D(\Gamma, \{x, y\}) = D(\Gamma', \{x, y\}) < m^2,$$

- if the axiom $\{x, y\}$ is not in the closure of Γ then

$$D(\Gamma', \{x, y\}) = l = \lceil m^2/\varepsilon \rceil,$$

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

2.2. Generating instances of uniform word

Let $\langle M, \mathtt{arity} \rangle$ 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 this means that a symbol from $\mathcal{G}$ is to be chosen. This is to assure that terms, which are symbols from $\mathcal{G}$, can be generated as well.

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

Notice that we generate a pair of terms in 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 $\{x, y\}$ that 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.

2.3. The algorithm and its expected time

The following deductive system for proving congruence of terms is given in [10]. 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. Recall that, a proof of $x \equiv_\Gamma y$ is a sequence $x_1, \ldots, x_n$ of terms such that

$$x = x_1 \to x_2 \to \cdots \to x_n = y.$$

Notice that a *single* step $x_i \to x_{i+1}$ can be efficiently computed in parallel (there are a polynomial number of axioms as well as substitutions to be checked for $x_i \to x_{i+1}$) hence the total parallel time for the whole proof of $x \to y$ depends on the length of the proof itself. We want to estimate the expected length 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 $(e \ln 4 + o(1)) \log n$ asymptotically with high probability, where n is the total number of terms of the given instance.

Theorem 1. *Assuming uniform distribution over descriptions of* $(\Gamma, \mathcal{T}, \{x, y\})$, *where* $\mathcal{T}$ *has* n *terms, then the expected length of proofs that decide whether a given pair of elements* $\{x, y\}$ *belongs to the closure of* Γ *is upper bounded by* $(e \ln 4 + o(1)) \log n$, *asympotically with high probability.*

Proof. Given a description of $(\Gamma, \mathcal{T}, \{x, y\})$, with $\mathcal{T}$ having a total of $n = 2^k$ terms, we construct a directed acyclic graph as follows. Let $\mathcal{P}$ be the set of all the proofs for $x \equiv_\Gamma y$. We can consider a proof as a directed path from x to y. Different proofs may have common terms other than x and y. So, let $\mathcal{Q}$ be the corresponding graph of $\mathcal{P}$ whose vertices are terms from $\mathcal{T}$ and there is an edge from vertex u to vertex v if $u \to v$. Further, we consider a topological order on $\mathcal{Q}$ (the order in which the terms are deduced) and classify the vertices of $\mathcal{Q}$ in the sets

$$N_i = \{T_j \ : \ 2^i \le j < 2^{i+1}\}$$

for $i = 0, \ldots, k-1$ where $T_0 = x, \ldots, T_{n-1} = y$.

Let d_{xy} denote the length of the (longest) proof for $x \equiv_\Gamma y$. Notice that when there is a proof of length d for $x \equiv_\Gamma y$, there must be a sequence of terms having length d, *i.e.*,

$$\alpha_0 = x \to \alpha_1 \to \cdots \to \alpha_d = y \tag{1}$$

and we will denote by A_d the event:

$$A_d = \{x \equiv_\Gamma y \text{ has a proof of length } d\}\cdot$$

Our first objective is to upper bound $\Pr[A_d]$. To this aim, we will express the event A_d as union of a set of disjoint events and compute their probabilities from which we will deduce the bound on $\Pr[A_d]$. We split the path (1) into the following sets:

$$L_i = \{\alpha_j : 1 \le j \le d \ , \ \alpha_j \in N_i\}$$

and let $l_i = |L_i|$. Note that L_i can be seen as an intersection of path (1) with the set N_i. For some intuition, we will use later on this splitting to bound the length of possible paths in any N_i which will be used, in turn, to bound the whole length.

The events expressing A_d are defined as follows: given i, $1 \le i \le k-1$ and numbers $m_i, \ldots, m_{k-1}$ such that $m_i \ne 0$ and $\sum_{j=i}^{k-1} m_j = d$ we let $A_{i, m_i, \ldots, m_{k-1}}$

be the event:

$$A_{i,m_i,\ldots,m_{k-1}} = \{\text{There is a proof } \alpha_0, \alpha_1, \ldots, \alpha_d \text{ s.t. } \alpha_1 \in N_i \wedge l_j = m_j \ \forall j \geq i\}\cdot$$

Clearly, A_d is the union of all $A_{i,m_i,\ldots,m_{k-1}}$ over all possible values of i and $m_i, \ldots, m_{k-1}$, as specified previously, and the events $A_{i,m_i,\ldots,m_{k-1}}$ are disjoint ones. Here is some intuition behind this splitting of A_d. Any path (*i.e.*, proof) starts by x. It is the position of the second term α_1 and the values of the parameters l_j which vary for different paths hence our event should *cover* all these cases. From the expression of A_d we can write

$$\Pr[A_d] = \sum_{i,m_i,\ldots,m_{k-1}} \Pr[A_{i,m_i,\ldots,m_{k-1}}]. \tag{2}$$

Next, we upper bound the $\Pr[A_{i,m_i,\ldots,m_{k-1}}]$ by

$$\Pr[A_{i,m_i,\ldots,m_{k-1}}] \leq \prod_{j=i}^{k-1} \Pr[\text{There is a path of length } m_j \text{ in } N_i]. \tag{3}$$

The following holds:

$$\Pr[A_{i,m_i,\ldots,m_{k-1}}] \leq 2^i (\ln 4)^d \prod_{j=i}^{k-1} \frac{1}{m_j!}\cdot \tag{4}$$

Indeed, we can express the set L_j as $L_j = \{\alpha_{2^j+n_1}, \ldots, \alpha_{2^j+n_{m_j}}\}$ for some values $n_1, \ldots, n_{m_j}$ satisfying $1 \leq n_1 < \ldots < n_{m_j} \leq 2^j$. Next, N_i has cardinality 2^i, we are considering paths of length m_j and each vertex in the path appears with the same probability, therefore:

$$\Pr[\text{There is a path of length } m_j \text{ in } N_i] \leq 2^i \sum_{n_1,\ldots,n_{m_j}} 2^{m_j} \left(\frac{1}{2^j+n_1}\right) \cdots \left(\frac{1}{2^j+n_{m_j}}\right)\cdot \tag{5}$$

In order to estimate the last sum above we remove the condition $1 \leq n_1 < \ldots < n_{m_j} \leq 2^j$ on $n_1, \ldots, n_{m_j}$ and use the following fact.

Fact 1. *Let $n_1, \ldots, n_k$ be independent and uniform numbers in $\{1, \ldots 2^j\}$. Then*

$$E\left[\frac{1}{2^j+n_1} \cdots \frac{1}{2^j+n_k}\right] \leq \left(\frac{\ln 2}{2^j}\right)^k \cdot$$

Proof. (of Fact 1) We have that

$$E\left[\frac{1}{2^j+n_1}\cdots\frac{1}{2^j+n_k}\right] = \sum_{n_1,\ldots,n_k}\frac{1}{2^j+n_1}\cdots\frac{1}{2^j+n_k}\Pr[\{n_1,\ldots,n_k\}]$$
$$\leq \frac{1}{2^{jk}}\int_0^1\cdots\int_0^1\frac{1}{1+x_1}\cdots\frac{1}{1+x_k}dx_1\cdots dx_k = \left(\frac{\ln 2}{2^j}\right)^k$$

thus the fact follows. □

Now, we apply this fact to bound the summation in (5) as follows: we bound the total sum of terms by the expected value of the summation term (where $n_1,\ldots,n_{m_j}$ are m_j independent random numbers in $\{1,\ldots,2^j\}$) multiplied by the total number of the terms in the summation. Therefore,

$$\begin{aligned}
\Pr[A_{i,m_i,\ldots,m_{k-1}}] &\leq 2^i\prod_{j=i}^{k-1}2^{m_j}\binom{2^j}{m_j}\left(\frac{\ln 2}{2^j}\right)^{m_j}\\
&\leq 2^i(\ln 2)^{m_i+\cdots+m_{k-1}}\prod_{j=i}^{k-1}2^{m_j}\frac{2^j!}{m_j!(2^j-m_j)!}\frac{1}{2^{jm_j}}\\
&\leq 2^i(\ln 2)^d\prod_{j=i}^{k-1}2^{m_j}\frac{1}{m_j!}\leq 2^i(\ln 2)^d\cdot 2^d\prod_{j=i}^{k-1}\frac{1}{m_j!}\\
= &\qquad 2^i(\ln 4)^d\prod_{j=i}^{k-1}\frac{1}{m_j!}
\end{aligned}$$

and thus proving (4). For the A_d, we have

$$\Pr[A_d]\leq(\ln 4)^d\sum_{i=0}^{k-1}2^i\sum_{m_i,\ldots,m_{k-1}}\prod_{j=i}^{k-1}\frac{1}{m_j!}\leq(\ln 4)^d2^k\sum_{i=0}^{k-1}\sum_{m_i,\ldots,m_{k-1}}\prod_{j=i}^{k-1}\frac{1}{m_j!} \quad (6)$$

where, again, $m_i,\ldots,m_{k-1}$ satisfy $m_j=0$ for $j<i$ and $\sum_{j=i}^{k-1}m_j=d$. Since summing $\prod_{j=i}^{k-1}\frac{1}{m_j!}$ up for all non-negative $m_i,\ldots,m_{k-1}$ satisfying $\sum_{j=i}^{k-1}m_j=d$ is equal to $\frac{k^d}{d!}$ (see, *e.g.* [9], page 64), we deduce

$$\Pr[A_d]\leq(\ln 4)^d2^k\,\frac{k^d}{d!}=2^k\left(\frac{k\ln 4}{(d!)^{1/d}}\right)^d. \quad (7)$$

Our final step for the proof of the theorem is to use (7) to derive the upper bound on $E[d_{xy}]$. Let us write $E[d_{xy}]=E_1+E_2$ where

$$E_1=\sum_{d\leq(e\ln 4+o(1))k}d\cdot\mathbf{Pr}[d_{xy}=d],\quad E_2=\sum_{d\geq(e\ln 4+o(1))k}d\cdot\mathbf{Pr}[d_{xy}=d].$$

For E_1 we have $E_1 \leq (e \ln 4 + o(1))k$ and for the E_2:

$$E_2 \leq \sum_{d \geq (e \ln 4 + o(1))k} d \cdot 2^k \left(\frac{k \ln 4}{(d!)^{1/d}} \right)^d . \tag{8}$$

By applying Striling's formula for $d!$, *i.e.*,

$$d! = \sqrt{2\pi d}\, \frac{d^d}{e^d} \left(1 + O\left(\frac{1}{d} \right) \right)$$

we observe that for $d \geq (e \ln 4 + o(1))k$

$$\frac{k \ln 4}{(d!)^1/d} = \frac{1}{1 + o(1)} . \tag{9}$$

Notice that when we fix a value of k in (8), the dominating term in the sumation is

$$\left(\frac{k \ln 4}{(d!)^{1/d}} \right)^d$$

hence, together with (9) we derive $\lim\limits_{k \to \infty} E_2 = 0$ and thus the thorem follows. $\square$

From Theorem 1 we have the following corollary.

Corollary 1. *The Uniform Word Problem is in the class Average NC.*

3. Unit resolution

The Unit Resolution is another classical P-complete problem. Jones and Laaser [8] gave a polynomial time algorithm for the problem together with a proof of P-completeness. A different proof of P-completeness, using a reduction from CVP, can be found in [7] (see also page 167 of Ref. [6]). Serna and Spirakis [15] showed that approximating this problem, under an appropriate definition of the optimization version, is also P-complete.

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

We will consider the 3Unit Resolution, the case when clauses have up to r literals follows along the same line of reasoning. The instances of 3Unit Resolution are generated in the standard way, namely by choosing at random 3SAT formulae. We will denote by N the total number of clauses of the instance.

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

$$U_1, U_2, \dots, U_m \equiv C \tag{10}$$

where U_j is (x_{i_j}, C_j). In other words, to achieve C we start from a pair (x_{i_1}, C_1) and apply step by step the unit resolution. So, the sequence (10) can be written as:

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

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, it is possible to reduce a clause from different sequences of unit step resolution, each of them beginning with a different clause. So we can define *the depth of a clause* C in $\mathcal{F}$ to be the longest sequence from which C is reduced.

Now, given an instance consisting of a set of clauses $\mathcal{F}$ and a clause C, 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 resolutions, where each level is done in parallel. Here a level is the set of clauses at the same depth. Note that, as in the case of Uniform Word, we can define a topological order in the closure of $\mathcal{F}$, namely the order in which the clauses are deduced. Of course, the expected running time of this algorithm depends on the depth of the closure of $\mathcal{F}$. We prove that under the uniform distribution of instances, the expected depth of the closure of $\mathcal{F}$ of N clauses has an upper bound of $(e \ln 4 + o(1)) \log N$, asymptotically with high probability. (The proof follows as in the case of Uniform Word and is omitted.)

Theorem 2. *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, chosen under the uniform distribution, is bounded by $(e \ln 4 + o(1)) \log N$, with high probability.*

4. Solvable path systems

Path Systems was defined by Cook [3]. This is the first problem shown to be P-complete, though it was not expressed in terms of completeness, as it is given in [8]. Serna and Spirakis [15] showed that approximating this problem, under an appropriate definition of the optimization version, is also P-complete. A path system 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 $x \in T$ or there are $y, z \in X$ such that $(x, y, z) \in R$ and both y and z are accessible. The *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 $X = \{x_1, \dots, x_n\}$. 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 $S = \{s_1, \dots, s_k\}$ and $T = \{t_1, \dots, t_l\}$. And now, generate the relation R:

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

Repeat this process until N elements of R are generated.

Clearly, every relation R in $X \times X \times X$ is generated, 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 proceeds 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 x's found in such a 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 order between the accessible elements. The execution 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 previous cases. Here we formulate just the theorem which states the result of that analysis.

Theorem 3. *For sufficiently large N the expected time to decide whether a given set S of a path system $P = \langle X, R, S, T \rangle$ chosen under uniform distribution, has some accessible element is bounded by $(e \ln 4 + o(1)) \log N$, with high probability.*

5. Generability problem

This problem was shown P-complete by Jones and Laaser [8] by reducing from Unit Resolution. Serna and Spirakis [15] showed that approximating this problem, under an appropriate definition of the optimization version, is also P-complete. It is defined as follows. 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 $X = \{x_1, \dots, x_m\}$ be a given set of m elements. Generate firstly, uniformly at random a set T of k elements from X, $T = \{t_1, \dots, t_k\}$ 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:= $\emptyset$;
    for i:=1 to $m^2$ do
        choose uniformly at random $x \in X$ and $y \in X$;
        if $(x, y) \notin$ tab then
            choose uniformly at random $z \in X$;
        set $x \bullet y = z$ and tab := tab $\cup \{(x,y)\}$;
    end
```

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 an operation are generated and the probability distribution over them is uniform.

The algorithm to find the closure of T proceeds as follows. Initially, we let `closure`(T):= T. In the 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 among the elements as they were inserted to `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 4. *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, assuming uniform distribution over discriptions of $\langle X, T, x, \bullet\rangle$, is bounded by $(e \ln 4 + o(1)) \log N$ with high probability.*

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. We are grateful to an anonymous referee for several useful comments.

References

[1] J.L. Balcázar, J. Díaz and J. Gabarró, *Structural complexity II*, Springer Verlag (1995).
[2] N. Calkin and A. Frieze, Probabilistic analysis of a parallel algorithm for finding a maximal independent set. *Random Structures and Algorithms* **1** (1990) 39–50.
[3] S.A. Cook, An Observation on time-storage trade-offs. *J. Comp. Syst. Sci.* **9** (1974) 308–316.
[4] D. Coppersmith, P. Raghavan and M. Tompa, Parallel graph algorithms that are efficient in average, in *Proc. of 28th IEEE Symposium on Foundations of Computer Science* (1987) 260–269.
[5] 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).
[6] R. Greenlaw, H.J. Hoover and W.L. Ruzzo, *Limits to parallel computation: P-completeness theory.* Oxford University Press (1995).
[7] J. Hoover and W. Ruzzo, A compendium of problems complete for P. *Manuscript* (1984).
[8] N.D. Jones and T. Laaser, Complete problems for deterministic polynomial time. *Theoret. Comput. Sci.* **3** (1977) 105–117.
[9] D.E. Knuth, *The art of computer programming*, Vol. 1. Addison-Wesley (1973).
[10] D. Kozen, Complexity of finitely presented algebras; in *Proc. of 9th ACM STOC* (1977) 164–167.

[11] R.E. Ladner, The circuit value problem is logspace complete for P. *SIGACT News* **7** (1975) 18–20.

[12] M. Serna, *The parallel approximbility of P-complete problems.* PhD thesis, Universitat Politècnica de Catalunya (1990).

[13] M. Serna, Approximating linear programming is logspace complete for P. *Inform. Proc. Lett.* **37** (1991) 233–236.

[14] S. Sahni and T. Gonzalez, P-complete approximation problems. *J. Assoc. Comput. Mach.* **23** (1976) 555–565.

[15] M. Serna and P.G. Spirakis, The approximability of problems complete for P, in *International Symposium on Optimal Algorithms*, Springer-Verlag, *Lecture Notes in Computer Science* **401** (1989) 193–204.

[16] T. Tsukiji and F. Xhafa, On the expected depth of randomly generated circuits, J. Díaz and M. Serna Eds., in *Proc. of Fourth European Symposium on Algorithms, ESA'96*, Springer-Verlag, *Lecture Notes in Computer Science* **1136** (1996) 208–220.

Communicated by J. Gabarro.
Received July, 1997. Accepted July, 1998.