Self-Reducibility of Hard Counting Problems with Decision Version in P*

Aris Pagourtzis[1] and Stathis Zachos[1,2]

[1] Department of Computer Science
School of Electrical and Computer Engineering
National Technical University of Athens
{pagour,zachos}@cs.ntua.gr
[2] CIS Department, Brooklyn College, CUNY

Abstract. Many NP-complete problems have counting versions which are #P-complete. On the other hand, #Perfect Matchings is also Cook-complete for #P, which is surprising as Perfect Matching is actually in P (which implies that #Perfect Matchings cannot be Karp-complete for #P).
Here, we study the complexity class #PE (functions of #P with easy decision version). The inclusion #PE $\subseteq$ #P is proper unless P = NP. Several natural #PE problems (e.g., #Perfect Matchings, #DNF-Sat, #NonCliques) are shown to possess a specific self-reducibility property. This implies membership in class TotP [KPSZ98,PZ05]. We conjecture that all non-trivial problems of #PE share this self-reducibility property.

1 Introduction

Traditional complexity classes (P, NP, PH, ...) contain decision problems, computed by TM acceptors. TMs can, of course, compute functions as well (in complexity classes such as FP, FNP, ...). Here, we discuss counting functions: their values are numbers (#) of paths of polynomial-time bounded nondeterministic TM acceptors (PNTMs).

Valiant [Val79] has introduced #P as the class of functions that count the number of accepting paths of a PNTM. This class contains several interesting counting problems, in particular counting variants of classical NP-search problems. Some complete problems for this class, and thus hard to compute, are #Sat, #HamiltonCycles, etc. A remarkable fact is that, while for these problems the decision (existence) version is NP-complete, there are other #P-complete problems which have easy (polynomial-time) decision versions, e.g., #Perfect Matchings, #DNF-Sat, etc.

In this paper we investigate the complexity of such "hard-to-count-easy-to-decide" problems (term borrowed from [DHK00]). A key observation is that

* Research supported in part by "Pythagoras" grant of the Hellenic Ministry of Education, co-funded by the European Social Fund (75%) and National Resources (25%). Operational Programme "Education and Initial Vocational Training" (EPEAEK II).

these latter problems are #P-complete under Cook reductions — but cannot be #P-complete under Karp reductions. It turns out that Cook reductions blur structural differences between counting problems and complexity classes (see also [KPZ99]). The reason is that #P as well as several other classes is not closed under Cook reductions (under reasonable assumptions). Therefore, in an effort to characterize these problems, we look for subclasses of #P — closed under Karp reduction — for which there is a possibility that (some of) these problems are Karp-complete.

#PE was introduced in [Pag01] as the subclass of #P that consists of functions f such that for any input x the question '$f(x) > 0$?' is polynomial-time decidable. By definition, #PE contains #Perfect Matchings and #DNF-Sat but not #Sat, unless P=NP.

In this paper we explore #PE problems with respect to having a specific self-reducibility property (SRP). Among others we show that #Perfect Matchings, #DNF-Sat, and #NonCliques possess the SRP. Moreover we exhibit the existence of artificial #PE problems that cannot have the SRP unless P = NP.

TotP was introduced in [KPSZ98] as the class of functions that count the total number of computational paths of a PNTM; it was proved that this class is equivalent to #P under Cook reductions. On the other hand, it was shown in [KPZ99] that #P does not reduce to TotP under Karp reductions, i.e., TotP is a proper subclass of #P unless P = NP.

In [PZ05] we show that TotP contains those problems of #PE that possess the SRP.

2 Preliminaries and Definitions

Our model of computation is the polynomial-time bounded nondeterministic Turing machine (PNTM), i.e., there is some polynomial p so that for any input x, all computation paths have length at most $p(|x|)$, where $|x|$ is the length of the input (for a detailed definition of a PNTM see, e.g., [BDG88] or [Pap94]). A counting function $\Sigma^* \rightarrow \mathbb{N}$ is associated with a PNTM M:

$$\mathrm{acc}_M(x) = \#\text{accepting paths of } M \text{ on input } x$$

In his seminal paper [Val79] Valiant introduced the class of counting functions #P:

$$\#\mathsf{P} = \{\mathrm{acc}_M \mid M \text{ is a PNTM}\}$$

Equivalently, #P can be seen as the class of functions that count the number of solutions of NP-search problems:

$$\#\mathsf{P} = \{f \mid \exists \text{ predicate } Q \in \mathsf{P},\ \forall x:\ f(x) = |\{y \mid Q(x,y)\}|\}$$

#P is a class of very high complexity. Toda showed [Tod91] that $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}[1]}$, where PH is the whole Polynomial-time Hierarchy [Sto76].

3 Classes of "Hard-to-Count-Easy-to-Decide" Problems: #PE and TotP

In [KPZ99] it was shown that while #P and TotP are equivalent under Cook reductions, this is not so under Karp reductions unless P = NP. The difference between Karp and Cook reductions is well reflected in the behavior of #P-complete problems. While there exist problems which are #P-complete under both reductions (namely, the counting versions of most NP-complete problems), there also exist problems which are #P-complete only under Cook reduction (unless P=NP). These latter problems usually have a decision version in P and are sometimes called "hard-to-count-easy-to-decide" problems. Some examples: #Perfect Matchings, #DNF-Sat, #NonCliques.

For each function $f \in \Sigma^* \to \mathbb{N}$ we define a related language:

$$L_f = \{x \mid f(x) > 0\}$$

Note that for function problems this language represents a natural decision version of the problem. In particular, if a function f corresponds to the counting version of a search problem (i.e., f counts how many solutions are there for a given instance) then L_f corresponds to the existence version.

The class #PE contains functions of #P whose related language is in P (i.e., the question "$f(x) > 0$?" is polynomial-time decidable). In other words, #PE by definition contains all *hard-to-count-easy-to-decide* problems.

The definition of TotP involves a function associated with every PNTM M:

$$\mathrm{tot}_M(x) = \#\text{paths of } M \text{ on input } x - 1$$

Remark 1. The 'minus one' in the definition of tot_M above was introduced so that the function can have a zero value.

TotP is defined as the class of all tot_M functions:

$$\mathsf{TotP} = \{\mathrm{tot}_M \mid M \text{ is a PNTM}\}$$

TotP and #PE have several similarities — and some differences.

In the remaining of the paper we discuss some #PE problems that possess the following *Self-Reducibility Property (SRP)*: An instance of the problem can be reduced to two simpler instances of the same problem. Thus, computing the number of solutions can be represented by a recursive tree. In other words, there exists a non-deterministic algorithm, the computation tree of which has as many paths as solutions (plus one: this technicality is necessary for the no-solution case).

4 #PE-complete Problems that Possess the SRP

In this section we show that important #PE problems possess the SRP.

Theorem 1. *#Perfect Matchings possesses the SRP.*

Proof. Given a graph $G = (V, E)$ let $e = (u, v)$ be an edge of G. Let also

$$G_0 := (V - \{u, v\},\ E - \{\text{all edges incident to } u \text{ or } v\}), \quad G_1 := (V,\ E - \{e\}).$$

Clearly, there is a 1-1 correspondence between perfect matchings of G that contain e and perfect matchings of G_0; moreover, each perfect matching of G that does not contain e is also a perfect matching in G_1, and vice versa. Hence:

$$\#\text{PM}(G) = \#\text{PM}(G_0) + \#\text{PM}(G_1)$$

□

The SRP gives rise to a recursive algorithm for #Perfect Matchings that works in non-deterministic polynomial time and has $\#\text{PM}(G) + 1$ computation paths, because the question "$\#\text{PM}(G_i) > 0$?" can be decided in polynomial time. In other words, #Perfect Matchings is in TotP (see also [PZ05]). More specifically:

If G has no perfect matchings, this can be tested in polynomial time and the recursion can be terminated (one computation path). It remains to show that for any graph G with *at least one perfect matching*, the recursion leads to exactly $\#\text{PM}(G)$ computation paths. We show this by induction on the number of edges. Note that the extra computation path required by the definition of class TotP can be easily added as a "dummy" computation.

- If G has only one edge $e = (u, v)$, and $\#\text{PM}(G) > 0$, then G contains no other nodes except u and v and $\#\text{PM}(G) = 1$. Therefore, G_0 is the empty graph while G_1 contains only nodes u and v with no edge between them. Hence, $\#\text{PM}(G_0) = 1$ (the empty matching) and $\#\text{PM}(G_1) = 0$. The algorithm can terminate having executed a unique computation path.
- Assume that the claim is true for all graphs with $\leq k$ edges for some $k > 0$. Consider a graph G with $k + 1$ edges and $\#\text{PM}(G) > 0$. Select an edge and construct two subgraphs of G as explained above, namely G_0 and G_1, both having $\leq k$ edges; i.e. $\#\text{PM}(G) = \#\text{PM}(G_0) + \#\text{PM}(G_1)$. If both $\#\text{PM}(G_0) > 0$ and $\#\text{PM}(G_1) > 0$, the algorithm continues recursively executing itself for G_0 and for G_1. Due to the induction hypothesis, the former leads to a computation tree with exactly $\#\text{PM}(G_0)$ leaves while the latter leads to a tree with exactly $\#\text{PM}(G_1)$ leaves. Hence, the whole computation tree has $\#\text{PM}(G)$ leaves. On the other hand, if only one of the G_i's has a perfect matching, then this can be tested in polynomial time and only the corresponding call is performed, leading to a computation tree with the correct number of leaves. Note also that it cannot happen that both $\#\text{PM}(G_0) = 0$ and $\#\text{PM}(G_1) = 0$ since we have assumed that G has at least one perfect matching.

The second problem that we examine is #DNF-Sat: a Boolean formula in DNF is given and we want to count the number of satisfying assignments.

Theorem 2. *#DNF-Sat possesses the SRP.*

Proof. This can be done because it is possible to select a variable x and construct formulae $\phi_0 := \phi|_{x=0}$ and $\phi_1 := \phi|_{x=1}$. Clearly:

$$\#\text{DNF-Sat}(\phi) = \#\text{DNF-Sat}(\phi_0) + \#\text{DNF-Sat}(\phi_1)$$

□

As before, we can show that the SRP — together with the fact that "#DNF-Sat(ϕ_i) > 0?" can be decided in polynomial time — makes it possible to describe a polynomial time non-deterministic algorithm that on input ϕ has #DNF-Sat(ϕ) $+ 1$ computation paths (see [PZ05]).

Finally, we show that the problem #NonCliques also possesses the SRP. The problem is defined as follows: given a graph $G = (V, E)$ and an integer $k > 1$, find the number of subsets of V of size k which are not cliques. Note that the problem is in #PE since #NonCl$(G, k) > 0$ if and only if G is not a complete graph. Besides, it is #P-complete in the Cook sense because one call to the #NonCliques oracle suffices in order to compute the number of cliques of a graph. More specifically for a graph G with n nodes:

$$\#\text{Cliques}(G, k) = \binom{n}{k} - \#\text{NonCl}(G, k)$$

Theorem 3. *#NonCliques possesses the SRP.*

Proof. For a graph $G = (V, E)$, let #NonCl(G, k) denote the number of non-cliques of size k in G. Select a node v and let $u_1, \ldots, u_l$ be the nodes that are not adjacent to v. Let also

$$G_0 := (V - \{v\},\ E - \{\text{all edges incident to } v\}),$$
$$G_1 := (V - \{v, u_1, \ldots, u_l\},\ E - \{\text{all edges incident to } v, u_1, \ldots u_l\}).$$

A careful counting reveals that #NonCl(G_0, k) is equal to the number of non-cliques of size k in G that do not contain v; #NonCl$(G_1, k-1)$ is equal to the number of non-cliques of size k in G that contain v but none of the u_i's; finally, $\sum_{i=2}^{l} \binom{n-i}{k-2}$ is equal to the number of non-cliques of size k in G that contain v and at least one of the u_i's. Hence:

$$\#\text{NonCl}(G, k) = \#\text{NonCl}(G_0, k) + \#\text{NonCl}(G_1, k-1) + \sum_{i=2}^{l} \binom{n-i}{k-2}$$

Besides, for any graph G, #NonCl$(G, 1) = 0$ and #NonCl$(G, k) = 0$ if $k > n$.

□

Again, a polynomial time non-deterministic algorithm with number of computation paths equal to #NonCl$(G, k) + 1$ exists because the question #NonCl$(G, k) > 0$ can be decided in polynomial time.

Using similar arguments it can be shown that several other #PE-complete problems possess the SRP.

5 Self-reducibility: Generalization and Formalization

In the previous section we saw that several important #PE problems have the SRP. Recall that SRP for all these problems is the possibility of reducing them to other instances of the same problem. Let us now formalize this notion:

Definition 1. *A function* $f : \Sigma^* \to \mathbb{N}$ *is called* poly-time self-reducible *if there exist polynomial time computable functions* $h_0, h_1 : \Sigma^* \to \Sigma^*$ *and* $g : \Sigma^* \to \mathbb{N}$ *such that for all* $x \in \Sigma^*$*:*

(a) $f(x) = f(h_0(x)) + f(h_1(x)) + g(x)$*,*
(b) there exists polynomial q *such that* $f(y) = 0$ *for* $y = (h_{i_{q(|x|)}} \circ \ldots \circ h_{i_1})(x)$*,* $i_j \in \{0, 1\}$*.*

In the above definition, (a) states that f can be processed recursively by reducing x to $h_0(x)$ and $h_1(x)$, and (b) states that the recursion terminates after at most polynomial depth.

Remark 2. A broader notion of self-reducibility could have been used instead of (a) in Definition 1: $f(x) = g(x) + \sum_{i=0}^{r(|x|)} g_i(x) f(h_i(x))$, where r is a polynomial and $h_i, g_i \in \mathsf{FP}, 0 \leq i \leq q(|x|)$ and $g \in \mathsf{FP}$. We have chosen to use the simpler notion since it suffices for our purposes.

We have recently proven the following:

Theorem 4 ([PZ05]). TotP *is exactly the closure under Karp reductions* ($\leq^p_{\mathrm{m}}$) *of the class of* #PE *problems that possess the SRP.*

Therefore, results of the previous section imply that all three problems #Perfect Matchings, #DNF-Sat and #NonCliques belong to TotP. Moreover, several other counting problems can now be easily shown to be in TotP. For example, consider the problem #NonIndependentSets: given a graph $G = (V, E)$ and an integer $k > 1$, find the number of subsets of V of size k which are not independent. Note that the problem is in #PE since the number of non-independent subsets of V of size k is non-zero if and only if $|E| \neq 0$, i.e. G has at least one edge. This problem is also #PE-complete under Cook reductions using an analogous argument as that used for #NonCliques.

Corollary 1. #NonIndependentSets *is in* TotP*.*

Proof. The number of non-independent sets of size k in G is equal to the number of non-cliques of size k in the complement graph $\bar{G}$. This implies a Karp reduction of #NonIndependentSets to #NonCliques. Using Theorem 4 the claim follows.

Of course, it is not too hard to show directly that #NonIndependentSets has the SRP and therefore is in TotP, again using Theorem 4; the above example illustrates an alternative, often simpler technique for obtaining membership in TotP.

On the other hand, not all #PE problems possess the SRP, unless P = NP. Consider for example the problem $\#\text{SAT}_{+1}$: for any input CNF formula ϕ, return the number of satisfying assignments plus one. Problem $\#\text{SAT}_{+1}$ is trivially in #PE since it is in #P (simulate a PNTM for SAT and add one computation path) and for any Boolean formula ϕ, $\#\text{SAT}_{+1}(\phi) > 0$. If $\#\text{SAT}_{+1}$ possesses the SRP then by Theorem 4 it is in TotP, hence $\exists$ PNTM M:

$$\#\text{SAT}(\phi) + 1 = \#\text{paths of } M(\phi) - 1$$

i.e., it would be decidable in polynomial time whether $\#\text{SAT}(\phi) > 0$ by simply checking whether M's computation on input ϕ has more than two computation paths or not; this would therefore imply P = NP. However, $\#\text{SAT}_{+1}$ is an artificial and a trivial problem, i.e., the corresponding decision version has always a 'yes' answer. It is an interesting open question whether there exist non-trivial #PE problems which do not possess the SRP under some reasonable complexity assumption.

6 Conclusions

In this paper we have investigated the *Self-Reducibility Property* (SRP) of counting functions with easy decision version (class #PE [Pag01]). We have shown that important counting problems such as #PERFECT MATCHINGS, #DNF-SAT, and #NONCLIQUES, have the SRP. An interesting consequence of our results is that all these functions — and several other, e.g. #NONINDEPENDENTSETS — belong to the complexity class TotP. This in turn implies that they can be described by means of an alternative computational model; namely, for each of them there is a PNTM that on a given input, executes as many computation paths as the function value on that input (see also [PZ05]). We expect that this characterization will help discover new properties of such problems.

We have also shown that there exist problems in #PE that do not possess the SRP unless P = NP; however, such problems are *trivial*, that is, the corresponding decision version has always a 'yes' answer. We conjecture that all non-trivial problems of #PE possess the SRP.

References

[BDG88] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I.* Springer-Verlag, 1988.

[DHK00] Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis. Subtractive Reductions and Complete Problems for Counting Complexity Classes. In Proc. *MFCS 2000*: 323-332.

[KPSZ98] Aggelos Kiayias, Aris Pagourtzis, Kiron Sharma, and Stathis Zachos. The complexity of determining the order of solutions. In *Proceedings of the First Southern Symposium on Computing*, Hattiesburg, Mississippi, 4–5 December 1998.

[KPZ99] Aggelos Kiayias, Aris Pagourtzis, and Stathis Zachos. Cook Reductions Blur Structural Differences Between Functional Complexity Classes. In *Proceedings of the 2nd Panhellenic Logic Symposium*, pp. 132-137, Delphi, 13–17 July 1999.

[Pag01] Aris Pagourtzis. On the complexity of Hard Counting Problems with Easy Decision Version. In *Proceedings of the 3rd Panhellenic Logic Symposium*, Anogia, Crete, 17–21 July 2001.

[PZ05] Aris Pagourtzis and Stathis Zachos. The complexity of Counting Functions with Easy Decision Version. Submitted, 2005.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Sto76] L. J. Stockmeyer. The Polynomial-time Hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[TO92] Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[Val79] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, April 1979.