Completeness Results for Counting Problems with Easy Decision

Eleni Bakali(✉), Aggeliki Chalki, Aris Pagourtzis, Petros Pantavos, and Stathis Zachos

School of Electrical and Computer Engineering, National Technical University of Athens, 15780 Athens, Greece
{mpakali,achalki}@corelab.ntua.gr, {pagour,zachos}@cs.ntua.gr, ppantavos@gmail.com

Abstract. Counting problems with easy decision are the only ones among problems in complexity class #P that are likely to be (randomly) approximable, under the assumption RP $\neq$ NP. TotP is a subclass of #P that contains many of these problems. TotP and #P share some complete problems under Cook reductions, the approximability of which does not extend to all problems in these classes (if RP $\neq$ NP); the reason is that such reductions do not preserve the function value. Therefore Cook reductions do not seem useful in obtaining (in)approximability results for counting problems in TotP and #P.

On the other hand, the existence of TotP-complete problems (apart from the generic one) under stronger reductions that preserve the function value has remained an open question thus far. In this paper we present the first such problems, the definitions of which are related to satisfiability of Boolean circuits and formulas. We also discuss implications of our results to the complexity and approximability of counting problems in general.

1 Introduction

Since Valiant introduced #P [25], the class of functions that count the number of accepting paths of a NPTM (Nondeterministic Polynomial Time Turing Machine), many counting classes arose in the literature. In [20] the class #PE was defined, as a subclass of #P that contains all functions of #P with easy decision version, that is, for a function $f \in$ #PE the problem "$f(x) \neq 0$?" is in P. #PE contains important problems such as Permanent [25], a special case of which is equivalent to counting perfect matchings in bipartite graphs. Another well known member of #PE is #DNF-Sat, i.e. the problem of counting satisfying assignments to DNF boolean formulas; for more such problems see [26]. Notably it was shown in [25,26] that Permanent, #DNF-Sat, as well as several problems presented in [26] are #P-complete, showing that counting is likely to be harder than decision (existence checking) for all these problems.

A subclass of #PE, namely TotP, was defined as the class of functions that count the total number of computation paths of the computation tree of a binary

DOI: 10.1007/978-3-319-57586-5_6

NPTM minus one [16]. Equivalently, for $f \in$ TotP, $f(x)$ is the *number of branchings* of the computation tree of a binary NPTM. TotP contains PERMANENT and #DNF-SAT, as well as all self-reducible problems of #PE under a natural notion of self-reducibility for counting problems. It is intriguing that problems in TotP have varying approximability status. In particular, TotP contains well approximable problems (e.g. #DNF-SAT), not approximable (within a polynomial factor) problems unless NP = RP (e.g. #IS that is the number of independent sets of all sizes), and also problems of yet unknown approximability status, conjectured to be "intermediate" (e.g. #BIS that is the number of independent sets of all sizes of bipartite graphs).

It is known that #PE contains TotP [16] and moreover that TotP is exactly the Karp closure of self-reducible functions of #PE [21]. There is a great number of self-reducible problems with easy decision which are therefore in TotP: counting matchings, computing the determinant of a matrix, computing the partition function of several models from statistical physics, like the Ising and the hard-core model, counting colorings of a graph with a number of colors greater than the maximum degree, counting bases of a matroid, computing the volume of a convex body, counting independent sets, and many more. TotP-completeness results can shed light to the complexity and approximability of all these problems and help treat such questions in a uniform way.

Regarding completeness results, as mentioned above, there are several #P-complete problems, which belong to TotP and therefore are TotP-complete under Cook reductions. More precisely, TotP and #P are interreducible under Cook reductions [16,17].

On the other hand, the situation is different when completeness under Karp (parsimonious) reductions is considered. In particular, there is no #P-complete problem in TotP, unless P = NP. For example, PERMANENT cannot be #P-complete under Karp reductions unless P = NP. Furthermore it also seems unlikely that PERMANENT is TotP-complete under Karp reductions. This is because such reductions preserve approximability and PERMANENT admits a FPRAS, while other problems in TotP, like #IS, are not likely to do so, as they are AP-interreducible with #SAT [9]. In this perspective completeness results in TotP and classes inside TotP may shed light on approximability of many counting problems.

In this paper, we present a first TotP-complete problem under parsimonious reductions, namely #MONOTONE-CIRCUIT-SAT: given the encoding of a monotone circuit with respect to a specific partial order (to be defined later), compute the number of inputs for which the circuit accepts. Then we reduce this to other problems, proving them to be also TotP-complete under parsimonious reductions. Finally we discuss some implications of our results in the last section.

1.1 Related Work

In recent years there has been great interest in classifying the approximation complexity of counting problems. This interest derives from the fact that very few counting problems have been proved to be in FP. At the same time the counting problems in #P with NP-complete decision version cannot have a polynomial time approximation, unless P = NP. Moreover in [9] it was proved that these problems are complete for #P under approximation-preserving reductions. Thus there is no FPRAS for any of them, unless NP = RP. Counting problems that could have efficient approximation algorithms (FPRAS, FPTAS) are counting problems with easy decision version. Such algorithms for counting problems can be found in [8,12,14,15]. Especially, the steady progress in determining the complexity of counting graph homomorphisms [10] contributed to the study of approximation counting complexity. Also the connection of counting problems with statistical physics have led to important results in this area [1–3,11].

Regarding subclasses of #P, counting classes like #L, SpanL [4], #PE [20], TotP [16], $\#\mathrm{R}\Sigma_2$ [23], $\#\mathrm{RH}\Pi_1$ [9] have been defined. A significant open question concerns the relation between each of these classes and the problems that admit a FPRAS. We are particularly interested in the class TotP, which is related to other subclasses of #P in the following way: $\mathrm{FP} \subseteq \mathrm{SpanL} \subseteq \mathrm{TotP} \subseteq \#\mathrm{PE} \subseteq \#\mathrm{P}$.

Furthermore, TotP is equal to $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$, the class of interval size functions defined on total p-orders with efficiently computable lexicographically nearest function [7]. This was based on [13], in which Hemaspaandra et al. defined classes of interval size functions and characterized #P in terms of such functions.

2 Preliminaries

The model of computation is the nondeterministic polynomial-time bounded Turing machine (NPTM), i.e. there is some polynomial p such that for any input x from an alphabet Σ^*, all computation paths have length at most $p(|x|)$, where $|x|$ is the length of the input. In [25] Valiant introduced the class #P:

Definition 1. *Let R be a polynomial-time decidable binary relation and p a polynomial. Let f be the function such that given $x \in \Sigma^*$, $f(x) = \big|\{y : |y| = p(|x|) \wedge R(x,y)\}\big|$. #P is the class of all these functions. Equivalently, $\#\mathrm{P} = \{acc_M : M$ is a NPTM$\}$, where $acc_M(x) = \#$accepting paths of M on input x.*

The decision version of a function $f \in \#\mathrm{P}$ is the following problem: Given x, is $f(x)$ nonzero? Equivalently, is there at least one accepting path of M on input x? For each function f the related language $L_f = \{x : f(x) > 0\}$ can be defined. 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 of a solution.

Definition 2. *$TotP = \{tot_M : M$ is a NPTM$\}$, where $tot_M(x) = \#($all computation paths of M on input $x) - 1$.*

A second class of functions with similar properties was introduced in [20]. #PE is the class that contains all the functions in #P such that their decision version is polynomial-time decidable.

Definition 3. $\#\mathrm{PE} = \{f : f \in \#\mathrm{P}$ *and L_f is polynomial time computable*$\}$.

Reductions between functions can be defined in a similar manner to the Cook/Turing and Karp/many-one reductions between languages. The latter kind of reduction is often called parsimonious, when referring to functions that count the number of solutions to NP problems. We use the terms "Cook" and "Karp", as shortcuts for "poly-time Turing" and "poly-time many-one" respectively:

Definition 4. *Polynomial-time reductions between functions:*

- *Cook (poly-time Turing)* $f \leq_T^p g$: $f \in \mathrm{FP}^g$.
- *Karp (poly-time many-one)* $f \leq_m^p g$: $\exists h \in \mathrm{FP}$, $\forall x \in \Sigma^*$ $f(x) = g(h(x))$.

The relations among #P, #PE and TotP were explored in [21]. The notion of self-reducibility is crucial for this investigation.

Definition 5. *A function* $f : \Sigma^* \to \mathbb{N}$ *is called poly-time self-reducible if there exist polynomials* r *and* q*, and polynomial time computable functions* $h : \Sigma^* \times \mathbb{N} \to \Sigma^*$, $g : \Sigma^* \times \mathbb{N} \to \mathbb{N}$, *and* $t : \Sigma^* \to \mathbb{N}$ *such that for all* $x \in \Sigma^*$*:*

(a) $f(x) = t(x) + \sum_{i=0}^{r(|x|)} g(x,i) f(h(x,i))$, *that is,* f *can be processed recursively by reducing* x *to* $h(x,i)$ $(0 \leq i \leq r(|x|))$, *and*
(b) the recursion terminates after at most polynomial depth (that is, $f\big(h(...h(h(x,i_1),i_2)...,i_{q(|x|)})\big)$ *can be computed in polynomial time).*
(c) $|h(...h(h(x,i_1),i_2)...,i_{q(|x|)}| \in O\big(poly(|x|)\big)$.

Note that if $|h(x,i)| < |x|$ for every x and i, $0 \leq i \leq r(|x|)$, then requirement (b) holds trivially. Moreover, (c) requires that f must be computed only on inputs of polynomial length in $|x|$, which also holds if h is of decreasing length.

Theorem 1 [21]. *(a)* $\mathrm{FP} \subseteq$ *TotP* $\subseteq \#\mathrm{PE} \subseteq \#\mathrm{P}$. *The inclusions are proper unless* $\mathrm{P} = \mathrm{NP}$.
(b) TotP is the Karp closure of self-reducible #PE *functions.*

Although, TotP, #PE and #P are Cook-equivalent, they are not Karp equivalent unless P = NP. This means that:

- Under Karp reductions, #P-complete, #PE-complete and TotP-complete problems constitute disjoint classes, unless P = NP.
- Under Cook reductions, TotP-complete problems are contained in #PE-complete problems which are contained in #P-complete problems.

In order to fully classify a problem, we need to prove that it is complete for a class under Karp reductions. As it can be easily observed, the fact that a problem is #P-complete under Cook reductions does not give enough information about its complexity, since it could belong in TotP. Cook reductions blur structural differences between classes.

In the rest of this section, we present definitions and observations useful for the main proof of this paper.

A tree is called (a) *binary* if every node has at most two children, (b) *full binary* if every node has either zero or two children, (c) *perfect binary* if it is binary, all interior nodes have two children and all leaves have the same depth.

Let M be a NPTM. We can modify M, without changing the total number of its paths, so that it has at most two nondeterministic choices at each step. Therefore, the computation of M on input x can be seen as a binary tree $T_{M(x)}$, i.e. a branching is created in the computation tree whenever M has to select between two choices. If there is only one choice for some state-symbol combination, we consider that the tree has no branching at this point. We conclude that we can restrict ourselves to full binary computation trees. So, it is not hard to see that $tot_M(x) = \#(\text{all paths of } M \text{ on input } x) - 1 = \#\text{branchings of } T_{M(x)}$.

Furthermore, the nondeterministic choices of the computation of M can be represented as a binary string y (a left branching corresponds to "0" and a right branching to "1"). When we write $M(x, y)$ we refer to the output of the Turing machine M on input x and nondeterministic choices y. Specifically, $M(x, y) = 1$ if M accepts x with nondeterministic choices y, and $M(x, y) = 0$ otherwise.

In the following sections we make use of two mappings from natural numbers to binary strings, as well as a special partial order on natural numbers.

Definition 6. *We define the* tree *partial order, denoted by* $\leq_{\text{tree}}$*, of* $\mathbb{N}$ *as follows. It is reflexive and transitive and, if* $y = 2x+1$ *or* $y = 2x+2$ *then* $x \leq_{\text{tree}} y$*.*

Note that the graph of this partial order is an infinite perfect binary tree denoted by T, the nodes of which are labeled with natural numbers, in such a way that the left to right BFS traversal of this tree yields the natural order of $\mathbb{N}$ (assuming that the left child of x is $2x + 1$ and the right one is $2x + 2$). Its root is labeled with 0, and $x \leq_{\text{tree}} y$ if and only if y is a descendant of x on this tree. The structure of T is illustrated in Fig. 1.

Using the notion of the infinite tree T we can define mappings between natural numbers and strings:

Definition 7. *1.* $\text{path} : \mathbb{N} \to \{0, 1\}^*$*. It maps* n *to the binary string that describes the path that starts from the root of* T *and ends at the node with label* n*. For example,* $\text{path}(3) = 00$*,* $\text{path}(9) = 010$*,* $\text{path}(0) = \varepsilon$*, where* ε *is the empty string.*

2. $\text{num} : \{0, 1\}^* \to \mathbb{N}$*. It is defined as the inverse mapping of* path.

3. $\text{bin}_k : \{0, 1, \dots, 2^k - 1\} \to \{0, 1\}^k$*. It maps* n *to its binary representation padded with leading zeros, so as to have length* k*. For example,* $\text{bin}_6(3) = 000011$*,* $\text{bin}_4(9) = 1001$*, and* $\text{bin}_3(9)$ *is not defined.*

In addition, bin_k^{-1} is the inverse of bin_k. For simplicity, we slightly abuse notation and use bin and bin^{-1}, when the length of the binary representation is clear from the context. The functions path, num, bin_k and bin_k^{-1} are polynomial-time computable.

Definition 8. *If we restrict* $\leq_{\text{tree}}$ *on* $\{0, 1, \dots, 2^k - 1\}$ *and apply* bin_k*, we obtain a partial order of* $\{0, 1\}^k$*, which, abusing notation, we also denote by* $\leq_{\text{tree}}$*.*

Let T^k denote the complete binary tree representing $\leq_{\text{tree}}$ on $\{0, 1\}^k$; an illustration of T^3 is given in Fig. 2.

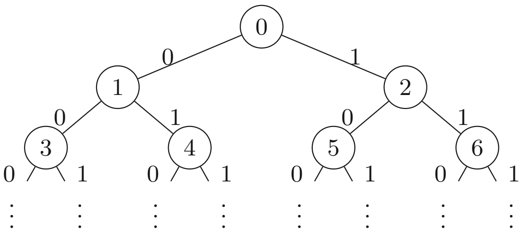

Fig. 1. The infinite perfect binary tree T.

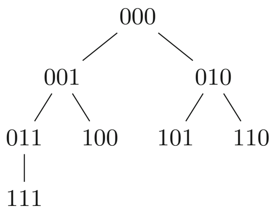

Fig. 2. Tree T^3.

3 #Monotone-Circuit-Sat is TotP-complete Under Karp Reductions

In this section we define a new counting problem and we prove that it is TotP-complete. Let C_n denote a Boolean circuit (see [5]) with n input gates, and let $C_n(z)$ be the output of C_n on input $z \in \{0,1\}^n$.

Definition 9. *We call a Boolean circuit C_n non-increasing with respect to $\leq_{\text{tree}}$ if for every $x, y \in \{0,1\}^n$, $x \leq_{\text{tree}} y$ implies that $C_n(x) \geq C_n(y)$.*

Definition 10. #MONOTONE-CIRCUIT-SAT, *denoted also by $f_{\#MC}$*
Input: A Boolean circuit C_n, non-increasing with respect to $\leq_{\text{tree}}$.
Output: $f_{\#MC}(C_n) := |\{y \in \{0,1\}^n : C_n(y) = 1\}|$, i.e. the number of satisfying assignments for C_n.

3.1 #Monotone-Circuit-Sat is TotP-hard

We prove that the function $f_{\#MC}$ is TotP-hard by reducing the computation of any function $h \in$ TotP to $f_{\#MC}$.

The key observation is the following. There is a NPTM M such that for any input x, $h(x) = tot_M(x)$; let $T_{M(x)}$ denote the corresponding computation tree. Consider extending $T_{M(x)}$ to a perfect binary tree $S_{M(x)}$ with the same height, so that all leaves of the original $T_{M(x)}$ tree and all their descendants are labeled "halting". Therefore $h(x) = \#(\text{branching nodes of } T_{M(x)}) = \#(\text{non-“halting” nodes of } S_{M(x)})$.

We construct a circuit C non-increasing w.r.t. $\leq_{\text{tree}}$, such that the number of accepting inputs of C equals $h(x)$. The idea is to describe a bijection between inputs of C and paths from the root to nodes of $S_{M(x)}$. C accepts an input if and only if the corresponding path ends at a non-"halting" node of $S_{M(x)}$, which in turn corresponds to a branching node of $T_{M(x)}$.

Theorem 2. *If $h \in$ TotP then $h \leq_m^p f_{\#MC}$.*

Proof. Let $h \in$ TotP, and M the corresponding binary NPTM. Recall that for every input x, $h(x) = tot_M(x) = \#\text{branchings of } T_{M(x)}$, where $T_{M(x)}$ is the computation tree of $M(x)$. Let p be a polynomial bounding the running time of M, thus the height of $T_{M(x)}$ is at most $p(|x|)$. Given the description of M we can construct a NPTM M' such that for every input x of M:

(i) $T_{M'(x)}$ is a perfect binary tree of height $p(|x|)+1$.
(ii) $\#(\text{accepting paths of } M'(x)) = \#(\text{branchings of } T_{M(x)})$.
(iii) For $y_1, y_2 \in \{0,1\}^{p(|x|)+1}$, if $y_1 \leq_{\text{tree}} y_2$, then $M'(x, y_1) \geq M'(x, y_2)$.

In order to describe M' we make use of the functions path and bin defined in Definition 7. The operation of M' on input x proceeds as follows:

1. Guess a binary string y of length $p(|x|)+1$. Let $n_y = \text{bin}^{-1}(y)$.
2. Compute $z = \text{path}(n_y)$.
3. Simulate M on input x and nondeterministic choices z.
   – If the simulation reaches a halting state of M (possibly using only a prefix of z), then output 0.
   – If the simulation uses all bits of z without reaching a halting state of M, then output 1.

We now show that properties (i), (ii), (iii) hold:

(i) The computation tree of M' is a perfect binary tree of height $p(|x|)+1$, since the only nondeterministic choices are made in Step 1 (Step 3 is deterministic).
(ii) The number of accepting paths of M' equals the number of branchings of M, since M' outputs 1 if and only if z corresponds to a computation path of M ending at a branching; recall that bin and path are bijective.
(iii) To prove the third property, it suffices to show that for all y_1, y_2 such that $y_1 \leq_{\text{tree}} y_2$ we have $M'(x, y_1) = 0 \Rightarrow M'(x, y_2) = 0$. If $y_1 \leq_{\text{tree}} y_2$, then $z_1 = \text{path}(\text{bin}^{-1}(y_1))$ is a prefix of $z_2 = \text{path}(\text{bin}^{-1}(y_2))$. This means that whenever M' simulates M with nondeterministic choices determined by z_2, it first passes through the same states as when it simulates M with nondeterministic choices determined by z_1. So, $M'(x, y_1) = 0$ means that the simulation of M reaches a halting state using (some of) the bits of z_1. Thus the remaining bits of z_2 are ignored and 0 is returned, therefore $M'(x, y_2) = 0$.

In order to complete the proof, we have to construct for each input x of h a circuit C_n^x with $n = p(|x|)+1$ input gates, that simulates the computation of M' on input x, i.e. for all $y \in \{0,1\}^n$, $C_n^x(y) = M'(x, y)$. It is well known that such a construction can be done in polynomial time (see e.g. [22, pp. 171–172]). C_n^x is non-increasing w.r.t. $\leq_{\text{tree}}$ since M' has this property (due to (iii)). Thus, we have that $|\{y \in \{0,1\}^n : C_n^x(y) = 1\}| = \#acc_{M'}(x) = tot_M(x)$, i.e. $f_{\#MC}(C_n^x) = h(x)$ so the reduction is parsimonious. □

3.2 #Monotone-Circuit-Sat Is in TotP

By Theorem 1(b), it suffices to prove that $f_{\#MC}$ is a self-reducible #PE function.

Proposition 1. $f_{\#MC} \in \#\mathrm{PE}$.

Proof sketch. It is not difficult to see that $f_{\#MC} \in \#\mathrm{P}$. Moreover, the decision version is easy since it suffices to simulate C_n on input 0^n. $\square$

Proposition 2. *$f_{\#MC}$ is self-reducible.*

Proof sketch. For proving that $f_{\#MC}$ is self-reducible, the intuition is that the number of satisfying assignments of a circuit C_n non-increasing w.r.t. $\leq_{\text{tree}}$, equals 0 iff 0^n is not a satisfying assignment. Otherwise it is equal to (the number of satisfying assignments that lie on the left subtree of T^n) + (the number of satisfying assignments that lie on the right subtree of T^n) +1. The proof consists of showing that we can efficiently construct two circuits non-increasing w.r.t. $\leq_{\text{tree}}$: C^0_{n-1}, with values compatible with the values of C_n on the left subtree of T^n, and C^1_{n-1} the corresponding for the right. $\square$

Corollary 1. *$f_{\#MC} \in$ TotP.*

Remark. Note that $f_{\#MC}$ is a "promise" problem, since it is not known how to check if a circuit is non-increasing w.r.t. $\leq_{\text{tree}}$. This is not an essential issue, as we can extend the function $f_{\#MC}$ on non-valid inputs to be equal to $tot_M(x)$, where M is the NPTM implied by the membership of $f_{\#MC}$ in TotP (on valid inputs).

4 More TotP-complete Problems

In this section we will show several problems to be TotP-complete. The proofs, omitted due to space limitations, will appear in the full version of the paper.

Definition 11. *Let U be a partially ordered set. A subset $V \subseteq U$ is called a lower-set (downwards closed) if for all $y, x \in U$, ($y \in V$ and $x < y$) $\Rightarrow x \in V$.*

Definition 12. *Let a circuit C_n with n input gates. We will call a subset V of $\{0,1\}^n$ accepting for C_n if for all $x \in V$, $C_n(x) = 1$.*

Definition 13. *We define the problem* MAX-LOWER-SET-SIZE.
Input: A circuit C_n with n input gates.
Output: The size of the maximum lower set w.r.t. $\leq_{\text{tree}}$, that is accepting for C_n.

Theorem 3. *The problem* MAX-LOWER-SET-SIZE *is TotP-complete.*

In the following we assume that each $n \in \mathbb{N}$ is encoded by path(n), and let T be the infinite perfect binary tree representing $\leq_{\text{tree}}$ on $\mathbb{N}$ (Fig. 1).

The next problem is intuitively the problem of counting the number of nodes of a subtree S of T, where S is given in a succinct way, i.e. not explicitly, but rather by a predicate that tells us whether a node v of T belongs to S.

Definition 14. Size-of-Subtree, *denoted by* f_{ss}
Input: $(M_A, u \in \mathbb{N}, 1^k, 1^t)$ *where* M_A *is a deterministic TM computing a predicate* $A : \mathbb{N} \to \{0,1\}$ *and* $t \in \mathbb{N}$.
Output: The size of the maximal subtree of S *with root* u*, where* $S = \{v \in T \mid \text{distance}(u,v) \leq k, A(v) = 1$ *and* $A(v)$ *is computed by* M_A *in at most* t *steps*$\}$.

Theorem 4. f_{ss} *is TotP-complete.*

We next show another TotP-complete problem which is a special case of #Sat. Namely, the valid input formulas have the following special properties based on a clustering of the space of solutions $\{0,1\}^n$, where each cluster contains all assignments with their first k variables fixed to some values: (a) there is at most one satisfying assignment in each cluster, and it is easy to decide whether such an assignment exists and, if so, easy to find it, and (b) if we label each cluster according to their fixed values, then there is a certain kind of monotonicity among the clusters, described below.

Definition 15. *1. For a 3-CNF formula* ϕ *and* $k \in \mathbb{N}$ *we define* $f_\phi^k : \{0,1\}^k \to \mathbb{N}$ *such that* $f_\phi^k(a) = \#($*satisfying assignments of* ϕ *with prefix* $a)$ *for* $a \in \{0,1\}^k$.
2. A 3-CNF formula ϕ *with* n *variables is called* (k,n)*-clustered-monotone for some* $k \leq n$*, if for every* $a, b \in \{0,1\}^k$ *such that* $a \leq_{\text{tree}} b$*,* $f_\phi^k(a) = 0$ *implies* $f_\phi^k(b) = 0$.

Definition 16. *1.* $Y = \{(1^k, 1^n, \phi, M, 1^t) \mid k, n, t \in \mathbb{N},\ \phi \in \Phi$*, deterministic TM* $M : \{0,1\}^k \times \Phi \to \mathbb{N}\}$*,where* Φ *is the set of 3-CNF formulas on* n *variables.*
2. $U \subset Y$ *is the set of tuples* $(1^k, 1^n, \phi, M, 1^t)$ *where* ϕ *is* (k,n)*-clustered monotone, and* M *is a deterministic TM s.t.* $\forall a \in \{0,1\}^k$, $M(a,\phi) = \#($*satisfying assignments of* ϕ *with prefix* $a)$*, and* t *is an upper bound for the running time of* M *on every* a*, and on the given* ϕ.

Note that in the above definition, the operation of the TM M is differentiated w.r.t. whether the instance is on U or $Y \setminus U$. In U we have the promise that ϕ is clustered monotone and that M counts the number of satisfying assignments in each clusters. In Y, both ϕ and M can be arbitrary.

Definition 17. #Clustered-Monotone-Sat*, denoted by* $f_{\#CMS}$
Input: $y = (1^k, 1^n, \phi, M, 1^t) \in Y$
Output: $f_{\#CMS}(y) = \begin{cases} \#\textit{satisfying assignments of } \phi & \textit{, if } y \in U \\ \sum_{a \in S} M(a,\phi) & \textit{, if } y \in Y \setminus U \end{cases}$
where $S \subseteq T^k$ *is the largest subtree of* T^k *containing* 0^k *s.t.* $\forall a \in S$ $[M(a,\phi) > 0$ *and* $M(a,\phi)$ *is computed within* t *steps*$]$.

Theorem 5. $f_{\#CMS}$ *is TotP-complete.*

By introducing #Clustered-Monotone-Sat, which is a special case of #Sat, the intuition we want to capture is the following. Every problem in TotP is reduced, as made clear from the above proof, to a 3-CNF formula that is clustered monotone, and for all formulas created in this way we have an efficient algorithm that returns the number of satisfying assignments in each cluster. So this TotP-complete special case of #Sat is much more structured than the #P-complete version. This fact, combined with other known results concerning the approximability of counting problems, may have interesting consequences, as we will discuss in the next section.

It is also worth noting that #Clustered-Monotone-Sat is a special case of another #Sat variant which is SpanP-complete: given a formula ϕ on n variables, and a number $k \leq n$, compute the number of satisfying assignments that are different in the first k variables [19].

5 Discussion on Approximability Implications

On the Approximability of TotP. It is known that there are problems in TotP, e.g. #IS (as shown in [21]), that do not admit a FPRAS unless NP = RP [9], not even a polynomial factor approximation. This follows from the fact that for self-reducible problems a polynomial factor approximation would yield a FPRAS [24].

However, it turns out that the class TotP admits some kind of polynomial time approximation: the problem Size-of-Subtree is a special case of the backtracking-tree problem, studied in [18]; in that paper Knuth proposed a randomized algorithm. By appropriate adaptation we can use it to approximate Size-of-Subtree. The expected output value of the algorithm is exactly the desired value, but the variance can be exponential in the worst case. Thus this algorithm would not yield a FPRAS. Approximation algorithms under other notions of approximability for Size-of-Subtree were studied in [6].

The TotP-completeness of Size-of-Subtree under Karp reductions implies that the above algorithmic results can be applied to every problem in TotP. Recall that TotP contains self-reducible hard counting problems with easy decision version [21]. On the other hand these simple algorithms are essentially the best we can hope for, unless NP = RP, since #IS belongs to TotP.

On the Approximability of #P and Connections to Statistical Physics. Another interesting implication comes from the TotP-completeness of the problem #Clustered-Monotone-Sat (Definition 17), i.e. the problem of counting the number of satisfying assignments of formulas such that: (a) a solution can easily be found if one exists, and (b) their set of solutions is connected in a specific way as described before Definition 15. Combining this completeness result with the fact that #Sat can be reduced to #IS $\in$ TotP (i.e. counting independent sets) by a reduction that preserves approximability [9], we get that approximating the number of satisfying assignments of an arbitrary formula is as difficult as approximating the number of satisfying assignments of a formula with the above properties.

This is particularly interesting since there is a series of papers that relate counting complexity to statistical physics [1–3], from which we know that, for the "difficult" instances of Sat, the set of satisfying assignments is widely scattered in the space of all assignments (i.e. the boolean hypercube of n dimensions), and this scattering might be responsible for the hardness of Sat. Our results show that we can reduce (with an approximation-preserving reduction) an arbitrary instance with a set of solutions that are disconnected and for which it is hard to find even one solution, to an instance with a set of solutions that are connected in a way that we have described explicitly, and for which we can easily find one solution.

This can be viewed in two ways: For an optimist it shows that approximating #Sat may be not so difficult after all (e.g. perhaps NP = RP). On the other hand, a pessimist may conclude that #Sat is not only hard in general, but also (by such "hardness amplification") even seemingly easy (e.g. structured) cases would possess the same hardness.

6 Conclusion and Open Problems

We have made an important step towards a better understanding of the complexity class TotP by presenting problems that are TotP-complete under parsimonious reductions. However, these problems are not among the well-studied problems in TotP such as #IS, Permanent, etc. The completeness of such problems constitutes an intriguing open question. Note that, if Permanent is TotP-complete under parsimonious reductions, then NP = RP.

Another interesting direction would be to explore the approximability status of the problems presented in this paper. The positive approximability of these problems would transfer to every problem in TotP.

Acknowledgments. We would like to thank Antonis Antonopoulos for many useful discussions as well as the anonymous reviewers for their observations and corrections.

References

1. Achlioptas, D.: Random Satisfiability. In: Biere, A., et al. (eds.) Handbook of Satisfiability, pp. 245–270. IOS Press, Amsterdam (2009)
2. Achlioptas, D., Coja-Oghlan, A., Ricci-Tersenghi, F.: On the solution-space geometry of random constraint satisfaction problems. Random Struct. Algorithms **38**(3), 251–268 (2011)
3. Achlioptas, D., Ricci-Tersenghi, F.: Random formulas have frozen variables. SIAM J. Comput. **39**(1), 260–280 (2009)
4. Àlvarez, C., Jenner, B.: A very hard log-space counting class. Theoret. Comput. Sci. **107**(1), 3–30 (1993)
5. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press, New York (2009)
6. Bakali, E.: Self-reducible with easy decision version counting problems admit additive error approximation. Connections to counting complexity, exponential time complexity, and circuit lower bounds. CoRR abs/1611.01706 (2016)

7. Bampas, E., Gobel, A., Pagourtzis, A., Tentes, A.: On the connection between interval size functions and path counting. Comput. Complex., 1–47 (2016). doi:10.1007/s00037-016-0137-8. Springer
8. Dyer, M.: Approximate counting by dynamic programming. In: Proceedings of 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 693–699 (2003)
9. Dyer, M.E., Goldberg, L.A., Greenhill, C.S., Jerrum, M.: The relative complexity of approximate counting problems. Algorithmica **38**(3), 471–500 (2003)
10. Galanis, A., Goldberg, L.A., Jerrum, M.: Approximately counting H-colourings is #BIS-hard. SIAM J. Comput. **45**(3), 680–711 (2016)
11. Goldberg, L.A., Jerrum, M.: The complexity of ferromagnetic ising with local fields. Comb. Probab. Comput. **16**(1), 43–61 (2007)
12. Gopalan, P., Klivans, A., Meka, R., Štefankovič, D., Vempala, S., Vigoda, E.: An FPTAS for #knapsack and related counting problems. In: Proceedings of 52nd Annual Symposium on Foundations of Computer Science (FOCS), pp. 817–826 (2011)
13. Hemaspaandra, L.A., Homan, C.M., Kosub, S., Wagner, K.W.: The complexity of computing the size of an interval. SIAM J. Comput. **36**(5), 1264–1300 (2007)
14. Jerrum, M., Sinclair, A.: The Markov chain Monte Carlo method: an approach to approximate counting and integration. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-hard Problems, pp. 482–520. PWS, Boston (1996)
15. Karp, R.M., Luby, M., Madras, N.: Monte-Carlo approximation algorithms for enumeration problems. J. Algorithms **10**(3), 429–448 (1989)
16. Kiayias, A., Pagourtzis, A., Sharma, K., Zachos, S.: Acceptor-definable counting classes. In: Manolopoulos, Y., Evripidou, S., Kakas, A.C. (eds.) PCI 2001. LNCS, vol. 2563, pp. 453–463. Springer, Heidelberg (2003). doi:10.1007/3-540-38076-0_29
17. Kiayias, A., Pagourtzis, A., Zachos, S.: Cook reductions blur structural differences between functional complexity classes. In: Proceedings of 2nd Panhellenic Logic Symposium, pp. 132–137 (1999)
18. Knuth, D.E.: Estimating the efficiency of backtrack programs. Math. Comput. **29**(129), 121–136 (1975)
19. Köbler, J., Schöning, U., Toran, J.: On counting and approximation. Acta Inform. **26**, 363–379 (1989)
20. Pagourtzis, A.: On the complexity of hard counting problems with easy decision version. In: Proceedings of 3rd Panhellenic Logic Symposium, Anogia, Crete (2001)
21. Pagourtzis, A., Zachos, S.: The complexity of counting functions with easy decision version. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 741–752. Springer, Heidelberg (2006). doi:10.1007/11821069_64
22. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, New York (1994)
23. Saluja, S., Subrahmanyam, K.V., Thakur, M.: Descriptive complexity of #P functions. J. Comput. Syst. Sci. **50**(3), 169–184 (1992)
24. Sinclair, A.J., Jerrum, M.R.: Approximate counting, uniform generation and rapidly mixing Markov chains. Inf. Comput. **82**, 93–133 (1989)
25. Valiant, L.G.: The complexity of computing the permanent. Theoret. Comput. Sci. **8**(2), 189–201 (1979)
26. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**(3), 410–421 (1979)