

# A short note on Merlin-Arthur protocols for subset sum

Jesper Nederlof<sup>\*</sup>

In the SUBSET SUM problem we are given positive integers  $w_1, \dots, w_n$  along with a target integer  $t$ . The task is to determine whether there exists a subset  $X \subseteq \{1, \dots, n\}$  such that  $\sum_{i \in X} w_i = t$ . We refer to such an  $X$  as a *solution*. The goal of this short note is to prove the following:

**Theorem 1.** *For a given SUBSET SUM instance  $w_1, \dots, w_n, t$  there is a proof of size<sup>1</sup>  $O^*(\sqrt{t})$  of what the number of solutions is that can be constructed in  $O^*(t)$  time and can be probabilistically verified in time  $O^*(\sqrt{t})$  with at most constant error probability.*

In complexity theory, a proof system as above is commonly referred to as a *Merlin-Arthur protocol*. These protocols very recently received attention [6, 3] in the exponential time setting: Williams [6] gives very generic Merlin-Arthur protocols with verifiers more efficient than best known exponential time algorithms. By no means we claim this note is innovative in any way: both our work and [6] have a similar flavor and it is likely that Theorem 1 is also obtainable via a clever application of [6, Theorem 1.1], or at the very least our observation will be less surprising given [6, 3].

*Proof of Theorem 1.* The proof system is outlined in Algorithm 1. Note that both the prover and the verifier algorithms are simple modifications of the well known pseudo-polynomial time dynamic programming algorithm for SUBSET SUM usually attributed to Bellman.

<p><b>Algorithm</b> <math>P(w_1, \dots, w_n, t)</math></p> <p><b>Output:</b> Prime <math>p = \Theta(\sqrt{nt})</math>, integer <math>c_i</math> for <math>i \leq nt, i \equiv_p t</math> with <math>c_i =  \{X \subseteq [n] : w(X) = i\} </math>.</p> <ol style="list-style-type: none"> <li>1: Initiate <math>T[0, 0] = 1</math> and <math>T[0, i] = 0</math> for <math>0 &lt; i \leq nt</math>.</li> <li>2: <b>for</b> <math>j = 1 \rightarrow n</math> <b>do</b></li> <li>3:   <b>for</b> <math>i = 1 \rightarrow n \cdot t</math> <b>do</b></li> <li>4:     <b>if</b> <math>i &lt; w_j</math> <b>then</b> <math>T[j, i] \leftarrow T[j - 1, i]</math> <b>else</b> <math>T[j, i] \leftarrow T[j - 1, i] + T[j - 1, i - w_j]</math>.</li> <li>5: Pick smallest prime <math>p</math> satisfying <math>2\sqrt{nt} &lt; p &lt; 4\sqrt{nt}</math>.</li> <li>6: <b>for</b> <math>i \leq n \cdot t</math> such that <math>i \equiv_p t</math> <b>do</b></li> <li>7:   <math>c_i \leftarrow T[n, i]</math>.</li> </ol>	<p>Prove that the number of solutions is <math>c_t</math>.</p> <p>Brute-force and [1] is within time budget.</p>
<p><b>Algorithm</b> <math>V(w_1, \dots, w_n, t, p, \{c_i\})</math></p> <p><b>Output:</b> Yes if the proof is as output by <math>P</math>, no with constant probability if the proof is ‘incorrect’.</p> <ol style="list-style-type: none"> <li>1: Pick a prime <math>q</math> satisfying <math>2^{nt} &lt; q &lt; 2^{nt+1}</math>.</li> <li>2: Pick a random element <math>r \in \mathbb{Z}_q</math>.</li> <li>3: Initiate <math>T'[0, 0] = 1</math> and <math>T'[0, i] = 0</math> for <math>i \in \mathbb{Z}_p</math>.</li> <li>4: <b>for</b> <math>j = 1 \rightarrow n</math> <b>do</b></li> <li>5:   <b>for</b> <math>i = 1 \rightarrow n \cdot t</math> <b>do</b></li> <li>6:     <math>T'[j, i] \leftarrow (T'[j - 1, i] + r^{w_j} \cdot T'[j - 1, (i - w_j) \% p]) \% q</math>.</li> <li>7: Compute <math>\sum_i c_i r^i \% q</math>.</li> <li>8: <b>if</b> <math>\sum_i c_i r^i \equiv_q T'[n, t \% p]</math> <b>then return yes else return no</b>.</li> </ol>	<p>Verify the proof for the number of solutions.</p> <p>Sample and check primality until success.</p>

**Algorithm 1:** The prover-verifier protocol.

Recall the Prime Number Theorem, that is used in Lines 5 and 1:

**Claim 2.** *For large enough integers  $b$  there exist at least  $2^b/b$  prime numbers  $p$  in the interval  $2^b < p < 2^{b+1}$ .*

<sup>\*</sup>Department of Mathematics and Computer Science, Technical University of Eindhoven, The Netherlands, j.nederlof@tue.nl

<sup>1</sup>  $O^*(\cdot)$  omits factors polynomial in the input size.

By this result, we may sample  $2^{nt} < q < 2^{n+1}t$  and check primality until success and declare YES after  $n + \lg(t)$  unsuccessful tries (which happens with probability at most  $1/4$ ).

For the verification probability, it is straightforward to see that  $T[n, i]$  equals  $|\{X \subseteq [n] : w(X) = i\}|$ . Similarly, when considering  $r$  as indeterminate, we see that  $T'[n, t]$  is a polynomial in  $\mathbb{Z}_q[r]$  satisfying

$$T'[n, t] \equiv_q \sum_{\substack{i \leq nt \\ i \equiv_p t}} |\{X \subseteq [n] : w(X) = i\}| r^i.$$

Also note that  $|\{X \subseteq [n] : w(X) = j\}| < q$ . Thus we see that if the proof is correctly constructed as in  $\mathsf{P}$ ,  $\sum_i c_i r^i \% q$  and  $T'[n, t]$  are equivalent polynomials and  $\mathsf{V}$  accepts. On the other hand, if the proof is incorrect, e.g.  $c_i \neq T[n, i]$  for some  $i$ , we see that  $(\sum_i c_i r^i \% q) - T'[n, t]$  is not the zero polynomial and has degree at most  $n \cdot t$ . Thus, it has at most  $n \cdot t$  roots in  $\mathbb{Z}_q$ , and the probability that  $r$  is one of these roots is  $nt/q < 1/4$ . If  $r$  is not a root of the difference polynomial, the two evaluations necessarily differ and the verifier rejects. Thus in total the verifier does not reject an incorrect proof with probability at most  $1/2$ .

For the running time of  $\mathsf{P}$ , note that all involved integers are at most  $2^n$  and the running time bound follows directly. The running time of  $\mathsf{V}$  follows similarly by using taking modulus at each powering step at Line 6.  $\square$

Let us note that the above theorem also gives rise to similar results for parity versions of Set Partition, Set Cover, Hitting Set and CNF-Sat via the reductions of [5, 4] (since the reduction in [4, Theorem 4.9] from Subset Sum/ $m$  to Set Partition is parsimonious). But all of this was also shown in a stronger form in [6, 3].

Also [2, Theorem 1.6], in combination with the above theorem gives an  $O^*(2^{.499n})$ -sized proof that can be probabilistically verified in time  $O^*(2^{.499n})$  for SUBSET SUM. However, combining [6, Theorem A.1] with the methods from [2] gives a more efficient proof system leading to a proof of size  $O^*(2^{.3113n})$  that can be verified in time  $O^*(2^{.3113n})$ .

**Acknowledgements** The author would like to thank Petteri Kaski for bringing the question of whether SUBSET SUM admits a Merlin-Arthur protocol with proof size and verification time  $O^*(2^{(0.5-\epsilon)n})$  for  $\epsilon > 0$  to his attention. The author was not aware of the contents of [6, 3] while observing this note in the beginning of November'16. This work is supported by NWO VENI project 639.021.438, and was done while visiting the Simons institute in the fall of 2015.

## References

- [1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Ann. of Math*, 2:781–793, 2002.
- [2] Per Austrin, Mikko Koivisto, Petteri Kaski, and Jesper Nederlof. Dense subset sum may be the hardest. to appear at STACS'16.
- [3] Andreas Björklund and Petteri Kaski. How proofs are prepared at Camelot. *ArXiv e-prints*, February 2016.
- [4] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNFSAT. *CoRR*, abs/1112.2275, 2011.
- [5] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 74–84. IEEE, 2012.
- [6] Ryan Williams. Strong ETH Breaks With Merlin and Arthur: Short Non-Interactive Proofs of Batch Evaluation. *ArXiv e-prints*, January 2016.