Empirical Support for the High-Density Subset Sum Decision Threshold

T. E. O'Neil and T. Desell
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202–9015
Email: oneil@cs.und.edu tdesell@cs.und.edu

***Abstract*—This article describes several properties of the random problem space for the Subset Sum problem, derived both empirically and analytically. Empirical results support the conjecture that Subset Sum instances always have a solution when the input set S is a set of n elements with a maximum value of m, the target sum t is between m and the sum of the smallest $n-1$ elements of S, and $n > \lfloor m/2 \rfloor + 1$. While the proof of this conjecture remains an open problem, exhaustive enumeration of problem instances has resulted in no counter-examples for values of $m \leq 49$. Sequential processing was used to generate the empirical data for values up to $m = 40$. The SubsetSum@Home volunteer computing project reproduced the results of the sequential code and extended the enumeration beyond $m = 49$.**

I. A Review of the Subset Sum and Partition Problems

Subset Sum is a well-known NP-complete problem [18] that can be defined as follows: given a set of positive integers S and a target sum t, determine whether some subset of S has sum t. The Partition problem, determining whether a set of numbers can be partitioned into two subsets that have the same sum, is also NP-complete. It can be defined to be a special case of Subset Sum. For any set of natural numbers S, let ΣS represent the sum of all elements of S. We define the Partition problem as follows: given a set of positive integers S, determine whether S has a subset with sum $\Sigma S/2$. The definitions above prohibit duplicate values in the set S. Variations on these definitions can be found in the research literature. Here we prefer the simplest possible definitions of the problems that remain NP-complete. In discussing Subset Sum or Partition, we will denote the size of the input set S as n and the maximum value in S as m, and given our assumption that S does not contain duplicates, we have $n \leq m$.

Exact algorithms for Subset Sum can be designed to search the space of all possible subsets or the space of all possible sums. Variants of backtracking and branch-and-bound are usually employed for enumerating subsets. These algorithms typically have worst-case time complexity $O(2^n)$, since the set S has 2^n subsets. Variants of dynamic programming are more appropriate for enumerating possible sums. Horowitz and Sahni provided a survey of these approaches for Subset Sum and 0/1 Knapsack in [17]. Their list-based sum enumeration algorithm remains a standard with worst-case time complexity $O(2^{n/2})$. Enumeration of all subset sums is a staightforward application of dynamic programming. For an input set S, no subset sum can exceed $\Sigma S < n \cdot m$. We use an array called *summap* of at most ΣS bits, initially all zeros. If some subset of S generates sum i, we set $summap[i] = 1$. Using this data structure, dynamic programming can be simply coded as a sequence of bitwise *shift* and *or* operations. Pseudo-code can be found in Figure 1. The algorithm has pseudo-polynomial time complexity $O(m \cdot n^2)$. This complexity may still be exponential in n, depending on the relationship between n and m.

If we use n and m as parameters to define the problem space of random instances for Subset Sum, we find that the performance of an algorithm may be very sensitive to the ratio n/m, which we will define to be the density of the input set. Combinatorial decision problems frequently have critical regions where the hardest instances reside. The critical region in a problem space is normally defined to be the region just below and above a crossover point, where random instances have a 50% chance of having a solution. And in the critical region, the probability of finding a solution changes rapidly from close to 0 (the overconstrained side of the crossover point) to almost 1 (the underconstrained side). We expect that a backtracking algorithm with appropriate bounding conditions will show the highest step counts for problems in the critical region. Studies of the critical region for the Partition problem began to appear in about 1996 (e.g. [16]). Crossover was predicted to occur at $m = 2^n$, or $n = \lg m$. S. Mertens [22, p. 5] published a more refined model a few years later. His analysis parameterized the average bit length b of the numbers in the set and predicted crossover when b was about equal to the size of the set n. For $m = 100000$, this model predicts crossover at $n = \lg m/.923$.

The current upper bound on the complexity of exact algorithms for the Subset Sum problem using n as the complexity parameter is apparently $O(n2^{n/2})$ [17]. The hardness of a Subset Sum instance, however, is highly dependent on the density of the input set. For very dense instances, where $m = O(n)$, a dynamic programming algorithm operates in

```
BitMap function findSums(set $S$)
// computes all sums of subsets of S
1)    BitMap $summap$: $summap[0] \Leftarrow 1$
2)    for each $num$ in $S$ from high to low
3)        BitMap $newmap \Leftarrow summap >> num$;
4)        $summap \Leftarrow summap$ or $newmap$;
5)    return $summap$;
```

Fig. 1. A dynamic programming algorithm for Subset Sum

polynomial time. Chaimovich, Frieman, and Galil [6] define an algorithm that has time complexity $O(m^2/n^2 + n \lg n)$, which is better than dynamic programming's $O(m \cdot n^2)$. The theorem can apparently be applied for any problem instances where $m \leq n^c$ where c is a constant. In [14], similar methods are applied to achieve an improved algorithm with time complexity $O(m \lg m)$. A few more recent articles describe specialized algorithms designed for high, medium, or low density instances of the Random Modular Subset Sum problem (RMSS), which has applications in cryptology. In these papers, density is more likely to be defined as the ratio of n to $\lg m$. The input is a set of n numbers with maximum value less than a modulus m and a target sum t, and the task is to find a subset that has sum $t(\text{mod } m)$. An expected polynomial-time algorithm can be found in [13] for so-called medium-density instances, where $\lg m$ falls between $\lg n$ and $(\lg n)^2$. In [21], a RMSS algorithm is given that has complexity $2^{O(\frac{n^\epsilon}{\log n})}$ for instances with $m = 2^{n^\epsilon}$ for arbitrarily small constants $0 < \epsilon < 1$. This is better than dynamic programming, which would yield $2^{O(n^\epsilon)}$ at this density.

If we use the total bit length x of the input as the complexity parameter instead of the number of elements n, we can find algorithms with strongly subexponential time complexity ($2^{O(x^c)}$ for a constant $c < 1$). Stearns and Hunt [29] defined such an algorithm for Partition with time complexity $2^{O(\sqrt{x})}$. Their algorithm divides the input set into lower and upper subsets, applies dynamic programming to the lower subset and backtracking to the upper subset. O'Neil and Kerlin [26] showed that a variant of dynamic programming called DDP can also achieve this result for Subset Sum without explicitly splitting the input list and without using backtracking. DDP has a strongly sub-exponential time complexity of $O(x^3 \cdot 2^{\sqrt{x}})$.

II. A Density-Based Decision Threshold

As mentioned in the previous section, the problem space for many combinatorial problems contains a critical region, an underconstrained region, and an overconstrained region. It is also possible that a solution is guaranteed in some subrange of the unconstrained region, and that a solution is impossible in some subrange of the overconstrained region. Knowledge of these thresholds can be translated to heuristics that make decision algorithms more efficient. A small-scale empirical study of Subset Sum was conducted in [24] to search for a density threshold beyond which solutions are guaranteed to be achievable. The results indicated a distinct density threshold, formally described in the conjecture below.

Definition 2.1: Let S be a set of natural numbers with maximum element m, that is, $S \subseteq \{1, \ldots, m\}$. We call a sum t *central* with respect to S if $m < t < \Sigma S - m$.

Conjecture 2.1 (The Decision Threshold Conjecture): Let S be a subset of $\{1, 2, \ldots, m\}$. If $|S| > \lfloor m/2 \rfloor + 1$, then for every $m \leq t \leq \Sigma S - m$, some subset of S has sum t.

Continuing empirical work to test the conjecture has grown into the SubsetSum@Home project, which is briefly summarized in [25] and described in detail in the sections below. As discussed in [25], proof of the conjecture would rigorously establish that the Subset Sum problem has strongly sub-exponential time complexity ($2^{O(\sqrt{x})}$ where x is the length of the input) even under symmetric represenation schemes. The empirical analysis has shown that most randomly generated problem instances well below the threshold (but still outside the critical region) have no missing central sums. There are, however, some interesting special cases that have missing central sums just below the threshold. If all numbers in a set are even, for example, the set size can reach $m/2$ and still not produce any odd sums. We can also find a set of size $m/2+1$ that has missing central sums. This result is described as a theorem in [25]:

Theorem 2.2: For every $m \geq 4$, there is a subset S of $\{1, 2, \ldots, m\}$ such that $|S| = \lfloor m/2 \rfloor + 1$ and no subset of S has sum $m + 2$.

Proof: Let $S = \{1, \lceil m/2 \rceil + 1, \lceil m/2 \rceil + 2, \ldots, m\}$. We observe that there is a gap in the sums of two-element subsets of S. The largest sum of a 2-element subset that contains 1 is $m + 1$, and the next larger subset has sum $\lceil m/2 \rceil + 1 + \lceil m/2 \rceil + 2$. So whether m is even or odd, no subset has sum $m + 2$. ■

It is interesting to note that if any additional number is added to the set in the proof of Theorem 2.2, every central sum is produced by some subset. This does not prove Conjecture 2.1, but it does establish a lower bound for the decision threshold. The proof of Conjecture 2.1 remains elusive. The research literature contains a density-based threshold theorem for subset sums, but it is not a strong as the conjecture above. The previously mentioned algorithms from [6] and [14] use a theorem from [1]. The theorem establishes that as the density of a set S grows, a cluster of subset sums will form in the sum set around and including $\frac{1}{2}\Sigma S$. The width of the sum cluster, however, is $2m \log m$, which is not wide enough to include the entire central region of conjecture 2.1.

III. The Empirical Study

The empirical study to provide support for Conjecture 2.1 has been conducted in two phases. The first phase uses a sequential Java program called *SumFinder*, adapted from the visualization program for Subset Sum instances described in [24]. The code that is applied to individual instances is an implementation of the dynamic programming logic from Figure 1 above. Java's BigInteger type, which provides built-in *shift* and *or* methods, is used as the data structure for *summap*. A *BitGenerator* class is used to generate problem instances. Each instance is a subset of $\{1, \ldots, m\}$. These sets are represented as bit maps of length m, and the *BitGenerator* has methods that iterate all bit strings of length m containing n ones in lexicographic order (as illustrated in [19], p. 17).

Various properties of the sum sets can be exploited to reduce the amount of work required for the exhaustive enumeration of problem instances and sum sets. The following definitions introduce some terminology to facilitate the discussion.

Definition 3.1: For a set of natural numbers S, let $sums(S)$ represent the set of all sums of subsets of S.

Definition 3.2: The set *sums(S)* has a *complete central region* if $|S| > 2$ and $sums(S)$ contains all central sums of subsets of S.

First we note that the sum sets for all problem instances are symmetric. This can save time in the enumeration of *sums(S)* for a single problem instance.

Theorem 3.1: The Subset Sum problem is symmetric around $\Sigma S/2$. Specifically, S has a subset with sum t if and only if S has a subset with sum $\Sigma S - t$.

Proof: If $S_1 \subseteq S$ has sum t, then its complement $S_2 = S - S_1$ has sum $\Sigma S - \Sigma S_1 = \Sigma S - t$. ■

It is also useful to observe that as numbers are processed by the DP algorithm from high to low, if the central region of the sum set becomes complete, it remains complete until all remaining numbers are processed. This property is formalized and proven in the lemma below.

Lemma 3.2: Consider $S = \{a_1, a_2, \ldots, a_n\}$, a subset of $\{1, 2, \ldots, m\}$ where $a_n = m$, $n \geq \lfloor m/2 \rfloor + 2$, and the elements of S are enumerated in increasing order, so that $a_i < a_{i+1}$ for $1 \leq i \leq n-1$. Let S_i represent the subset of S containing the largest i elements, $S_i = \{a_{n-i+1}, \ldots, a_n\}$. If $sums(S_k)$ has a complete central region where $k \geq 4$, then for all $k \leq j \leq n$, $sums(S_j)$ has a complete central region.

Proof: We can develop the proof in the context of the DP algorithm of Figure 2, which computes $sums(S_i)$ in the i^{th} iteration of its for loop. In the i^{th} iteration, a new sum set is computed by adding a_{n-i+1} to all sums in $sums(S_{i-1})$. Then $sums(S_i)$ is set to be the union of the new sum set and $sums(S_{i-1})$. If $sums(S_{i-1})$ has a complete central region, then the new sum set will contain a complete sequence of sums from $m + a_{n-i+1}$ to $\Sigma S_{i-1} - m + a_{n-i+1}$. If S_{i-1} has more than three elements, then it contains the three largest elements a_{n-2}, a_{n-1}, and m, and since $n \geq \lfloor m/2 \rfloor + 2$, it must be the case that $a_{n-2} + a_{n-1} \geq m$. it follows that $m + a_{n-i+1} \leq \Sigma S_{i-1} - m$, so that the new sum sequence will overlap the end of the previous sum set's central region, thus extending the central region for $sums(S_i)$ from m to $\Sigma S_{i-1} - m + a_{n-i+1} = \Sigma S_i - m$ without introducing any gaps in the sequence. ■

Lemma 3.2 can be used to formulate a more specific version of Conjecture 2.1, whose testing in the empirical study will be sufficient to confirm the more general version.

Conjecture 3.3 (The Sufficient Threshold Conjecture): Let S be a subset of $\{1, 2, \ldots, m\}$. If S contains m and $|S| = \lfloor m/2 \rfloor + 2$, then for every $m <= t <= \Sigma S - m$, some subset of S has sum t.

Theorem 3.4: If Conjecture 3.3 is true, then the following propositions also hold:

1) Let S be a subset of $\{1, 2, \ldots, m\}$ that contains m. If $|S| \geq \lfloor m/2 \rfloor + 2$, then for every t such that $m \leq t \leq \Sigma S - m$, some subset of S has sum t.
2) Let S be a subset of $\{1, 2, \ldots, m\}$ that does not contain m. If $|S| \geq \lfloor m/2 \rfloor + 2$, then for every $m \leq t \leq \Sigma S - m$, some subset of S has sum t.

Proof: Proposition (1) is equivalent to Conjecture 3.3 when $|S| = n = \lfloor m/2 \rfloor + 2$. Suppose we have $S_n = \{a_1, a_2, \ldots, a_n\}$ where $a_n = m$ and $n > \lfloor m/2 \rfloor + 2$, and the elements of S_n are enumerated in increasing order, so that $a_i < a_{i+1}$ for $1 \leq i \leq n-1$. Consider the subset S_{n-h} formed by removing the smallest h elements from S_n, where $h > 0$ is chosen to make $n - h = \lfloor m/2 \rfloor + 2$. According to Conjecture 3.3, S_{n-h} generates all sums t in the central region $m <= t <= \Sigma S_{n-h} - m$. We can assume that $n - h \geq 4$, since otherwise the constraint $n - h = \lfloor m/2 \rfloor + 2$ cannot be satisfied with $h > 0$ and $n \leq m$. This allows us to invoke Lemma 3.2 on S. Since S_{n-h} generates a complete central region and $n - h \geq 4$, the extension of $sums(S_{n-h})$ to $sums(S_n)$ maintains a complete central region.

For proposition (2), we have S a subset of $\{1, 2, \ldots, m\}$ that does not contain m and $|S| \geq \lfloor m/2 \rfloor + 2$. We can also characterize S as a subset of $\{1, 2, \ldots, m'\}$, where m' is the largest number in S. Since $m' < m$, we have $|S| > \lfloor m'/2 \rfloor + 2$ and we can apply Proposition 1 to get a complete sum range $m' \leq t \leq \Sigma S - m'$, which includes the complete subrange $m \leq t \leq \Sigma S - m$. ■

Corollary 3.5: If Conjecture 3.3 is true, so is Conjecture 2.1.

So the empirical study can be restricted to verification of Conjecture 3.3, reducing the number of sets that need to be tested. Only sets containing m need to be tested, and sets with density higher than the threshold do not need to be tested. Lemma 3.2 can also be used to justify early exit from the for loop in the DP algorithm for individual problem instances. The DP logic can be modified to return *true* as soon as a complete central region is detected in the sum set, even if all numbers in the input set have not yet been processed. This strategy was implemented in an alternative version of the *SumFinder* program mentioned above. The new version, however, actually took more time than the old. The average number of loop iterations per instance was reduced from n to about $.7n$, but the test for gaps in the central region was executed for each loop iteration, instead of once after the last iteration. The overhead of a linear scan of the sum list per iteration outweighs the savings in a reduced number of iterations. If the test for gaps could be conducted in connection with the shift operation, with only a constant additional cost, the new version would be faster than the old.

The first phase of the empirical study tested all problem instances with maximum value m and size $\lfloor m/2 \rfloor + 1$ for values of m up to and including 40. At $m = 40$, the running time was about two weeks. This sounds like a long time for a relatively small value of m, but it needs to be borne in mind that there are $\binom{40}{22} = 113,380,261,800$ distinct sets with maximum value 40 and size 22.

IV. The SubsetSum@Home Volunteer Computing Project

To extend the empirical evidence to significantly higher values of m, it was necessary to apply parallel and distributed processing. All problem instances relevant to the conjecture are very dense and so far the instance sizes are relatively small. The DP algorithm is efficient for such instances, so there was no need to parallelize the DP algorithm. Translation from Java to optimized C code provided a constant speed-up (by a factor of about ten) for individual instances, but beyond this, no attempt was made to develop a parallel version of DP.

The greatest potential for speed-up is to divide the space of problem instances into work packets that can be computed in parallel by participating processors. Instances can be tested independently, and the order of testing does not matter, so the overhead is limited to dividing the problem space, checking the results, and assembling the results. Redundancy is used to reduce the probability of malicious or erroneous results. Each instance is sent to different computers with different owners, amd the results are checked for consistency. If two results for the same work packet agree, they are accepted. Otherwise, the work packet is sent out again to two more different participants.

A. Volunteer Computing

Volunteer computing, where people volunteer their idle compute cycles to different computing projects, has emerged as a viable and significant source of computing power. It is being successfully used to perform research in scientific applications ranging from astronomy [8], [10], biology [11], [27], [23], [4], chemistry [15], and physics [28], [20], to climate modeling [7] as well as many other fields of inquiry. Berkeley's Open Infrastructure for Network Computing (BOINC) [2], [3] is the most widely deployed volunteer computing framework, in part due to its open source code and easy extension. As of February 2014, over 400,000 volunteered computers are participating in BOINC and contributing over 8.1 petaFLOPS (10^{15} floating point operations) per second of computing power [5], more powerful than many supercomputers [12]. Because of its easy extensibility and large user base, BOINC was chosen as the framework to implement the SubsetSum@Home project.

B. SubsetSum@Home

As of February 2014, SubsetSum@Home[1] has had 4,504 compute hosts participate in the project, volunteered by 1,229 users. The SubsetSum@Home source code is open source, hosted on GitHub[2], and the progress of all the $\binom{m}{n}$ subsets tested, along with every failed set are publicly available[3]. The original SubsetSum@Home application has evaluated all problem instances with $n = \lfloor m/2 \rfloor + 1$ and $n = \lfloor m/2 \rfloor + 2$ for $m \leq 49$, reaching the limits of what the client application can calculate without using multiprecision variables to perform the SubsetSum calculation.

A new SubsetSum@Home application is currently under development that uses the Boost C++ Libraries' [9] multiprecision integer class. The switch to multiprecision integers initially resulted in a 50-100x degradation in performance on standard processors. Because of this a GPU application is being developed to handle larger sized problem sets in a reasonable amount of time. Due to the inherently parallel nature of the problem, significant performance improvement is expected with a GPU implementation.

V. Conclusion

The theorems of the preceding sections, together with verification of the code used for the empirical testing, provide the basis for a proof that Conjecture 2.1 is valid for $n < 50$. While the SubsetSum@Home project cannot prove the conjecture for all n, it can potentially provide a counter example that disproves the conjecture. It can also provide data that may eventually lead to a proof of the general conjecture. The results for problem instances just below the threshold, for example, are informative. For even m and $n = \lfloor m/2 \rfloor + 1$, the number of instances that exhibit missing central sums is exactly two, while the number of failed instances for odd m shows at least polynomial growth. The authors hope that further study of these and other phenomena will eventually lead to a rigorous proof of what remains, for now, an open problem.

[1]http://volunteer.cs.und.edu/subset_sum
[2]https://github.com/travisdesell/Subset-Sum/
[3]http://volunteer.cs.und.edu/subset_sum/progress/index.php

References

[1] N. Alon and G. Freiman, On Sums of Subsets of a Set of Integers, it Combinatorica **8:4** (1988), pp. 297-306.

[2] D. P. Anderson, Volunteer computing: the ultimate cloud, *Crossroads* **16:3** (2010), pp. 7-10.

[3] D. P. Anderson, E. Korpela, and R. Walton, High-Performance Task Distribution for Volunteer Computing, *First International Conference on e-Science and Grid Techniques* (*e-Science* 2005), pp. 196-203.

[4] Adam L. Beberg, Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq and Vijay S. Pande, Folding@home: Lessons from eight years of volunteer distributed computing, *International Parallel and Distributed Processing Symposium* (IEEE Computer Society, 2009), pp. 1-8.

[5] BOINC Stats, *http://boincstats.com/* (2014).

[6] M. Chaimovich, G. Freiman, and Z. Galil, Solving Dense Subset-Sum Problems by Using Analytical Number Theory, *Journal of Complexity* **5** (Academic Press, 1989), pp. 271-282.

[7] C. Christensen, T. Aina, and D. Stainforth, The challenge of volunteer computing with lengthy climate model simulations, *First International Conference on e-Science and Grid Computing* (2005).

[8] Nathan Cole, Heidi Newberg, Malik Magdon-Ismail, Travis Desell, Kristopher Dawsey, Warren Hayashi, Jonathan Purnell, Boleslaw Szymanski, Carlos A. Varela, Benjamin Willett and James Wisniewski, Maximum Likelihood Fitting of Tidal Streams with Application to the Sagittarius Dwarf Tidal Tails, *Astrophysical Journal* **683** (2008), pp. 750-766.

[9] Bewman Dawes, David Abrahams, and Rene Rivera, The Boost C++ Libraries, *http://www.boost.org* (February 2014).

[10] Travis Desell, Benjamin A. Willet, Matthew Arsenault, Heidi Newberg, Malik Magdon-Ismail, Boleslaw Szymanski and Carlos Varela, Evolving N-Body Simulations to Determine the Origin and Structure of the Milky Way Galaxy's Halo using Volunteer Computing, *The IPDPS'11 Fifth Workshop on Desktop Grids and Volunteer Computing Systems* (*PCGrid* 2011).

[11] Travis Desell, Lee Newberg, Malik Magdon-Ismail, Boleslaw K. Szymanski and William Thompson, Finding Protein Binding Sites Using Volunteer Computing Grids, *2nd International Congress on Computer Applications and Computational Science* (*CACS* 2011).

[12] Jack Dongarra, *Top 500 Supercomputer Sites*, http://www.top500.org/ (2013).

[13] A. Flaxman and B. Przydatek, Solving Medium-Density Subset Sum Problems in Expected Polynomial Time, *Lecture Notes in Computer Science* **3404**, (2005) pp. 305-314.

[14] Z. Galil and O. Margalit, An Almost linear-time Algorithm for the Dense Subset-Sum Problem, *SIAM Journal on Computing* **20:6** (1991), pp. 1157-1189.

[15] I. J. General, E. K. Asciutto, and J. D. Madura, Structure of Aqueous Sodium Perchlorate Solutions, *The Journal of Physical Chemistry B* **112:48** (2008), pp. 15417-15425.

[16] I. Gent and T. Walsh, Phase Transitioning and Annealed Theories: Number Partitioning as a Case Study, Instituto per la Ricerca Scientifica e Tecnologica (IRST), Technical Report #9601-06 (1996).

[17] E. Horowitz and S. Sahni, Computing Partitions with Applications to the Knapsack Problem, *Journal of the Association for Computing Machinery* **21:2**(1974), pp. 277-292.

[18] R. Karp, Reducibility Among Combinatorial Problems, in *Complexity and Computer Computations*, eds. R. E. Miller and J. W. Thatcher (Plenum Press, New York, 1972), pp. 85–103.

[19] D. Knuth, *The Art of Computer Programming*, volume **4**, fascicle **3**, (Addison-Wesley, 2005).

[20] J. A. Lopez-Perez, J. Salt, and E. Ros, Distributed computing and farm management with application to the search for heavy gauge bosons using the ATLAS experiment at the LHC (CERN), Ph. D. Dissertation, University of Valencia (2008).

[21] V. Lyubashevsky, On Random High Density Subset Sums, *Electronic Colloquium on Computational Complexity*, Report No. **7** (2005).

[22] S. Mertens, The Easiest Hard Problem: Number Partitioning, Inst. f. Theor. Physik, University of Magdeburg, Magdeburg, Germany (2003).

[23] C. Mony, M. Garbey, M. Smaoui and M.-L. Benot, Large scale parameter study of an individual-based model of clonal plant with volunteer computing, *Ecological Modelling* **222:4** (2011), pp. 935-946.

[24] T. E. O'Neil, On Clustering in the Subset Sum Problem, *Proceedings of the 44th Midwest Instruction and Computing Symposium* (Duluth, MN, 2011).

[25] T. E. O'Neil, Complement, Complexity, and Symmetric Representation, accepted for publication, *International Journal of Foundations of Computer Science* (World Scientific, January 2015).

[26] T. E. O'Neil and S. Kerlin, A Simple $2^{O(\sqrt{x})}$ Algorithm for Partition and Subset Sum, *Proceedings of the 2010 International Conference on Foundations of Computer Science* (CSREA Press, 2010), pp. 55–58.

[27] S. Pellicer, N. Ahmed, Yi Pan and Yao Zheng, Gene sequence alignment on a public computing platform, *International Conference on Parallel Processing* (*ICPP* 2005), pp. 95 - 102.

[28] C. B. Ries, ComsolGrid - A framework for performing large-scale parameter studies using COMSOL Multiphysics and the Berkeley Open Infrastructure for Network Computing (BOINC), *Applied Sciences* (2010).

[29] R. Stearns and H. Hunt, Power Indices and Easier Hard Problems, *Mathematical Systems Theory* **23** (1990), pp. 209–225.