Phase Transition in the Number Partitioning Problem

Leena Salmela
T-79.7003 Research Course in Theorethical Computer Science

October 19, 2007

Abstract

Number partitioning is an NP-hard problem with applications for example in task scheduling. Two heuristic algorithms and their exact complete versions are presented. The random version of the problem has a phase transition in average complexity. This phase transition coincides with the phase transition in the probability of perfect partitions. This presentation is largely based on the review article by Mertens [4]. More details on the algorithms are given in [3] and the phase transition is analyzed more rigorously in [2].

1 Introduction

The number partitioning problem (NPP) is defined as follows. Given a list $a_1, a_2, \ldots, a_N$ of N integers, find a partition A that minimizes the discrepancy

$$E(A) = \left| \sum_{i \in A} a_i - \sum_{i \notin A} a_i \right|$$

If the sum $\sum a_i$ is even, a partition A is called perfect if $E(A) = 0$ and if the sum $\sum a_i$ is odd, a partition A is perfect if $E(A) = 1$. Thus a perfect partition is guaranteed to be an optimum partition. NPP has been proved to be NP-complete.

If the input numbers a_i are bounded by a constant A, i.e. $a_i < A$, then the discrepancy can take at most NA different values. A dynamic programming algorithm [1] exploring this search space in $O(N^2 A)$ time works as follows. The algorithm builds an $(N \times NA)$ table T. An entry (i, j) tells if a partition of size j can be constructed from the i first elements in the list. The first row $i = 1$ is initialized as follows:

$$T[1, j] = \begin{cases} \text{true} & \text{if } j = 0 \text{ or } a_1 = j \\ \text{false} & \text{otherwise} \end{cases}$$

The subsequent rows can then be filled with the following recursion:

$$T[i,j] = \begin{cases} \text{true} & \text{if } T[i-1,j] = \text{true or } T[i-1,j-a_i] = \text{true} \\ \text{false} & \text{otherwise} \end{cases}$$

The optimal partition then corresponds to a cell $T[i,j] = \text{true}$ where j is closest to $\frac{1}{2}\sum_i^N a_i$.

It would thus appear that a polynomial time algorithm exists for the NPP. However, any concise coding of the input numbers takes at least $N \log A$ bits space. Because A is not bounded by any polynomial function of $\log A$, the runtime of the algorithm is still exponential in the length of the input. This feature of NPP is called pseudo polynomiality. Also the NP-completeness of NPP requires the input numbers to be exponentially large in N.

Many analytical results of NPP have been proven for the real-valued version of the problem. In this case the input numbers are real numbers in the interval $[0,1]$. Besides the real-valued version of NPP there are several other interesting variations of the problem. In multiway NPP the numbers are partitioned into more than two subsets and in the balanced NPP there is an additional contraint that the produced sets must have the same cardinality.

A surprising feature of NPP is the poor quality of heuristic algorithms. It has been proven that the average discrepancy of optimum partitions is $O(\sqrt{N} \cdot 2^{-N})$ if the input numbers are independently and identically distributed random real numbers from the interval $[0,1]$. However the best heuristic algorithm for the real-valued version of the problem produces partitions with average discrepancies $O(N^{-\alpha \log N})$, where α is a constant.

2 Algorithms

2.1 Greedy Algorithm

The greedy algorithm attempts to keep the discrepancy as small as possible with every decision it makes. The algorithm chooses the largest unassigned number and assigns it to the set with the smallest sum. This is repeated until all numbers have been assigned to a set. For random real valued inputs the greedy algorithm yields average discrepancies $O(1/N)$ which is quite bad compared to the average discrepancies of optimum partitions. The time complexity of the greedy algorithm is dominated by the sorting of the numbers so it is $O(N \log N)$.

2.2 Differencing Algorithm

The differencing algorithm by Karmarkar and Karp attempts to reduce the length of the list of numbers in each iteration. This algorithm chooses two largest numbers of the instance and replaces them by their absolute difference. This corresponds to commiting to placing the two numbers in different

sets but leaving the final decision of which number goes where open. The algorithm continues to replace the largest numbers by their absolute difference until there is only one number left. This is the discrepancy of the partition. Note that some additional book keeping is needed to figure out the actual partition in the end of the algorithm.

The differencing algorithm produces average discrepancies $O(N^{-\alpha \log N})$ with $\alpha = 0.72$ for the real-valued problem with random numbers. This is somewhat better than the greedy algorithm but still quite far from the optimum. The running time of the differencing algorithm is dominated by sorting the initial numbers and thus the time complexity of the algorithm is $O(N \log N)$.

2.3 Complete Algorithms

Both of the above algorithms can be easily extended to algorithms for finding the exact optimal partition. In each iteration of the greedy algorithm we make a decision to put the largest remaining number to the set with the smallest sum. Obviously the only other option is to put the largest number to the set with the larger sum. The complete greedy algorithm explores all the 2^N possible partitions in this fashion. Thus the worst-case running time of the algorithm is exponential.

The differencing algorithm can be extended to a complete algorithm in a very similar fashion. Now in each iteration the algorithm decides to put the two largest numbers in different sets which corresponds to replacing them with their absolute difference. The other option is to put the two numbers in the same subset. This corresponds to replacing the numbers by their sum. Again this results in a complete algorithm that explores all the possible partitions.

The search of the complete algorithms can be pruned in several cases. For the differencing method the following pruning rules can be used:

1. It can be proven that the differencing heuristic finds the optimal solution when there are less than five numbers left. Thus in this case we can just apply the differencing heuristic.

2. If the largest number is larger or equal to the sum of the rest of the numbers, it is obvious that the best we can do is to place the largest number in one set and all the other numbers in the other set.

3. If we have found a perfect partition, we know that it is an optimal partition. Thus we do not need to continue the search.

Figure 1 shows the search space of the complete differencing method with input list $8, 7, 6, 5, 4$.

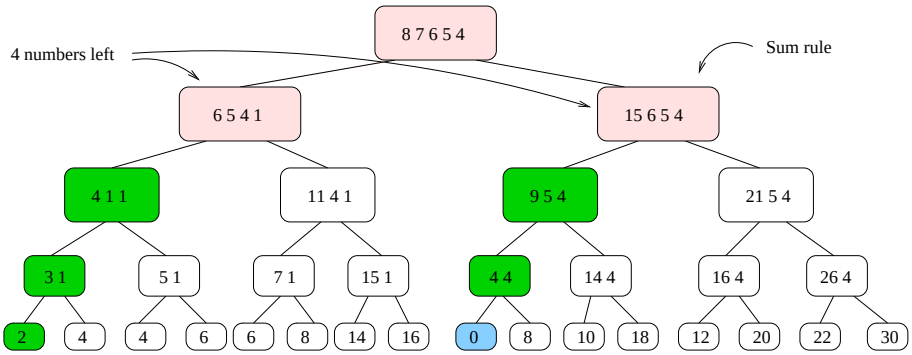

Figure 1: The search tree of the complete differencing algorithm. The pink nodes are explored by the algorithm using pruning rules, the green nodes are explored after applying a pruning rule and the blue node is the optimal solution.

3 Phase Transition

A random instance of the number partitioning problem consists of N numbers a_i which are chosen independently and uniformly at random from an integer interval $[0, A)$. If we define $A = 2^{\kappa N}$ all known exact algorithms have an exponential worst-case running time. However, the typical complexity of the problem depends on κ.

In practise, random $(\log A)$-bit integers are generated when we are looking for the phase transition in average complexity. In this case the parameter $\kappa = \log A/N$ so κ decreases as N increases. Then the algorithm is run for various N and the results are plotted on a graph. Figure 2 shows such a graph for instances with random 20-bit integers. As can be seen, the running time increases exponentially until $N = 24$ after which it decreases. In fact the complete differencing method has linear running time for large N. This can only mean that the method finds a perfect partition in the first try and because of the pruning, stops after that.

Figure 2 also shows the probability of an instance to have a perfect partition. It can be seen that the peak in the running time coincides with the sudden rise in the probability of perfect partition. The probability can be analytically analyzed rigorously [2] but we will show here an easier and shorter analysis which is somewhat sloppy [4].

A partition A can be coded with binary variables $s_i = \pm 1$ by defining that $s_i = +1$ if $a_i \in A$ and $s_i = -1$ otherwise. The discrepancy of the parition can then be defined as $E = |D(s)|$ where

$$D(s) = \sum_{i=1}^{N} a_i s_i. \tag{1}$$

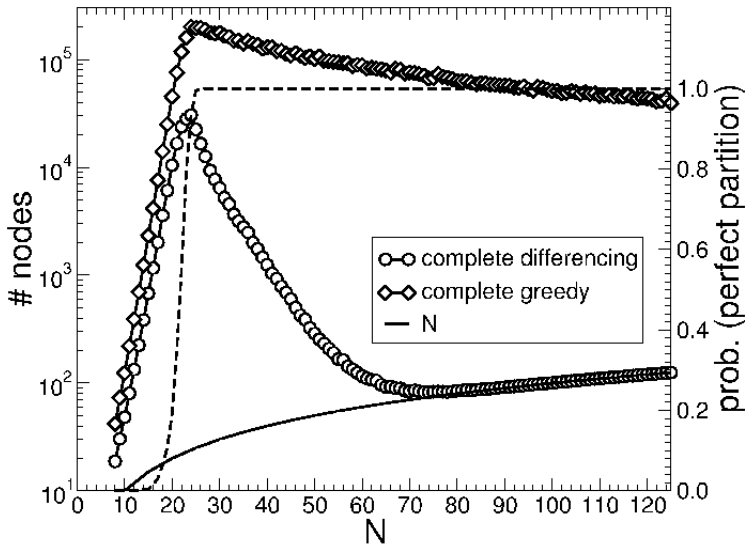

Figure 2: Phase transition in average running time for 20-bit integers. The dashed line shows the probability of a perfect partition. (From Mertens [4].)

If we consider a random walker in one dimension who takes steps to the right ($s_i = +1$) or left ($s_i = -1$) with random step sizes (a_i) then we can interpret D as the distance to the origin at the end of the walk. Then the average number of walks ending at D is

$$\Omega(D) = \sum_{\{s_i\}} \left\langle \delta \left(D - \sum_{i=1}^{N} a_i s_i \right) \right\rangle \tag{2}$$

where the averaging is over the random numbers $\{a_i\}$.

If we consider a fixed walk $\{s_i\}$ then for large N the distance to the origin, $\sum_{j=1}^{N} a_j s_j$, is Gaussian with mean

$$\langle D \rangle = \langle a \rangle M \tag{3}$$

where $M = \sum_j s_j$ and variance

$$\left\langle D^2 \right\rangle - \langle D \rangle^2 = M^2 \left\langle a^2 \right\rangle - (M \langle a \rangle)^2 = M^2 (\left\langle a^2 \right\rangle - \langle a \rangle^2) \tag{4}$$

Next we need to calculate averages of M and M^2 when averaging over the random walk $\{s_i\}$. For large N we get

$$[M] = 0 \tag{5}$$

and

$$\left[M^2\right] = \left\langle \sum_{i=1}^{N} s_i \sum_{j=1}^{N} s_j \right\rangle = \sum_{i=1}^{N} \left\langle s_i^2 \right\rangle + \sum_{i \neq j} \langle s_i s_j \rangle = N. \tag{6}$$

Now we can average $\langle D \rangle$ and $\langle D^2 \rangle$ over the random walk $\{s_i\}$:

$$[\langle D \rangle] = [M] \langle a \rangle = 0 \tag{7}$$

and

$$\left[\left\langle D^2 \right\rangle\right] - [\langle D \rangle]^2 = \left[M^2\right] \left\langle a^2 \right\rangle = N \left\langle a^2 \right\rangle \tag{8}$$

Substituting the mean (7) and variance (8) to the probability function of the Gaussian distribution, the probability of ending the walk at distance D reads

$$p(D) = \frac{1}{\sqrt{2\pi N \langle a^2 \rangle}} \exp\left(-\frac{D^2}{2N \langle a^2 \rangle}\right). \tag{9}$$

Now we have to note that the walk can only end at even numbers if the sum $\sum a_i$ is even and at odd numbers otherwise. Thus the average number of walks ending at D is

$$\Omega(D) = 2^N 2p(D) = \frac{2^{N+1}}{\sqrt{2\pi N \langle a^2 \rangle}} \exp\left(-\frac{D^2}{2N \langle a^2 \rangle}\right). \tag{10}$$

The second moment of a variable a with uniform discrete distribution is

$$\left\langle a^2 \right\rangle = \frac{1}{3} 2^{2\kappa N} (1 - O(2^{-\kappa N})). \tag{11}$$

Substituting this to Equation 10 and solving for $\log_2 \Omega(0)$ we get

$$\log_2 \Omega(0) = N(\kappa_c - \kappa) \tag{12}$$

with

$$\kappa_c = 1 - \frac{\log_2 N}{2N} - \frac{1}{2N} \log_2 \left(\frac{\pi}{6}\right). \tag{13}$$

From Equation 12 we see that when $\kappa < \kappa_c$ there are exponentially many perfect partitions and when $\kappa > \kappa_c$ the probability of a perfect partition is exponentially small. Thus the problem space of NPP is divided into two regions: The easy phase with $\kappa < \kappa_c$ where perfect partitions are abundant and heuristic algorithms often find a perfect partition in polynomial time and the hard phase with $\kappa > \kappa_c$ where perfect partitions are very rare and heuristic algorithms are no better than blind search.

References

[1] M.R. Garey and D.S. Johnson: *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freeman, San Fransisco (1979).

[2] Mertens, S.: Phase transition in the number partitioning problem. Phys. Rev. Lett. **81**, 4281 (1998).

[3] Mertens, S.: A complete anytime algorithm for balanced number partitioning. Preprint `arxiv.org/abs/cs/9903011` (1999)

[4] Mertens, S.: The easiest hard problem: Number partitioning. in: Percus, A.G., Istrate, G., Moore, C. (eds.), Computational complexity and statistical physics, Oxford University Press (2006), pp. 125–139