# Phase Transition in the Number Partitioning Problem

Leena Salmela

October 12th 2007

# Outline

# Outline

# Number Partitioning Problem (NPP)

- Input: a list $a_1, a_2, \ldots, a_N$ of positive integers
- Partition: a subset $A \subset \{1, \ldots, N\}$
- Discrepancy of partition $A$:

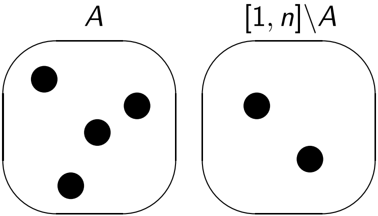
$$E(A) = \left| \sum_{i \in A} a_i - \sum_{i \notin A} a_i \right|$$

- Minimize the discrepancy

# NPP: Example

Multi-processor scheduling:

- 2 processors
- 5 tasks with runtimes $8, 7, 6, 5, 4$
- How to divide the jobs among the processors so that the tasks are completed as fast as possible?

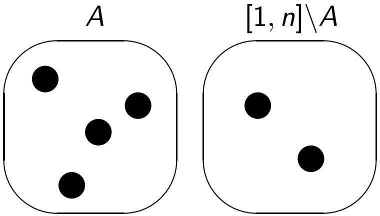



$$\sum_{i \in A} a_i = \sum_{i \notin A} a_i$$

# NPP: Example

Multi-processor scheduling:

- 2 processors
- 5 tasks with runtimes $8, 7, 6, 5, 4$
- How to divide the jobs among the processors so that the tasks are completed as fast as possible?
  - Processor 1 runs tasks with runtimes: $8, 7$
  - Processor 2 runs tasks with runtimes: $6, 5, 4$
  - Discrepancy: $|8 + 7 - 6 - 5 - 4| = 0$



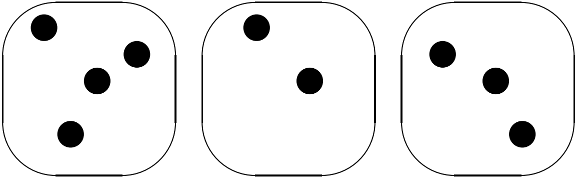

$$\sum_{i \in A} a_i = \sum_{i \notin A} a_i$$

# Perfect Partitions

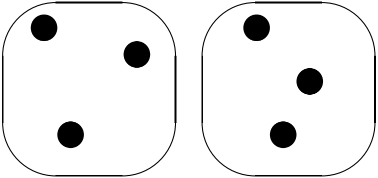
- Perfect partition:
  - Discrepancy $E(A) = 0$ if $\sum a_i$ even
  - Discrepancy $E(A) = 1$ if $\sum a_i$ odd
- The list $1, 2, 3, 4, 5, 6$ has a perfect partition $A = \{2, 3, 5\}$ because $E(A) = |2 + 3 + 5 - 1 - 4 - 6| = 1$.
- Not all lists have perfect partitions.

# Variations of NPP

- Input can be a list of real numbers from the interval $[0, 1]$
- Multiway NPP: Partitioning to more than 2 sets



- Balanced NPP: The sets must have the same cardinality



# Properties of NPP

- NP-complete
- Poor quality of heuristic algorithms
  - Average discrepancy of optimum partitions $O(\sqrt{N} \cdot 2^{-N})$ (for real-valued NPP)
  - Best heuristic algorithm yields average discrepancies $O(N^{-\alpha \log N})$ (for real-valued NPP)
- The phase transition can be analyzed rigorously analytically. (*S. Mertens: Phase Transition in the Number Partition Problem, Phys. Rev. Lett.* **81**, *4281 (1998)*)

# Outline

The Number Partitioning Problem (NPP)

Algorithms for NPP

Phase Transition in NPP

# The Greedy Algorithm

- Assign the largest unassigned number to the set with the smallest sum
- Iterate until all numbers are assigned

- Keeps the discrepancy small with every decision
- Yields average discrepancies $O(1/N)$ (for real-valued NPP) (*S. Mertens: The Easiest Hard Problem: Number Partitioning, in: A.G. Percus, G. Istrate, C. Moore, eds., Computational Complextity and Statistical Physics, Oxford University Press (2006), p. 125-139*)
- Time complexity $O(N \log N)$ (sorting)

# The Greedy Algorithm: An Example

Input: $\{8, 7, 6, 5, 4\}$

# The Greedy Algorithm: An Example

Input: $\{8, 7, 6, 5, 4\}$

Iter 1: 8 assigned to set 1
$\{8\}\{\}$

# The Greedy Algorithm: An Example

Input: $\{8, 7, 6, 5, 4\}$

Iter 1: 8 assigned to set 1
$\{8\}\{\}$

Iter 2: 7 assigned to set 2
$\{8\}\{7\}$

# The Greedy Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | 8 assigned to set 1 |
| | $\{8\}\{\}$ |
| Iter 2: | 7 assigned to set 2 |
| | $\{8\}\{7\}$ |
| Iter 3: | 6 assigned to set 2 |
| | $\{8\}\{7, 6\}$ |

# The Greedy Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | 8 assigned to set 1 |
| | $\{8\}\{\}$ |
| Iter 2: | 7 assigned to set 2 |
| | $\{8\}\{7\}$ |
| Iter 3: | 6 assigned to set 2 |
| | $\{8\}\{7, 6\}$ |
| Iter 4: | 5 assigned to set 1 |
| | $\{8, 5\}\{7, 6\}$ |

# The Greedy Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | 8 assigned to set 1<br>$\{8\}\{\}$ |
| Iter 2: | 7 assigned to set 2<br>$\{8\}\{7\}$ |
| Iter 3: | 6 assigned to set 2<br>$\{8\}\{7, 6\}$ |
| Iter 4: | 5 assigned to set 1<br>$\{8, 5\}\{7, 6\}$ |
| Iter 5: | 4 assigned to set 1<br>$\{8, 5, 4\}\{7, 6\}$ |

Discrepancy: 4

# The Differencing Algorithm by Karmarkar and Karp

- Key idea: Reduce the length of the number list

- Replace two largest numbers $a_i$ and $a_j$ by $|a_i - a_j|$
  - Commit to placing the two largest numbers in different sets
- Repeat until only one number left
- The remaining number = discrepancy

- Achieves better results than the greedy algorithm
  - Yields average discrepancies $O(N^{-\alpha \log N})$ with $\alpha = 0.72$ (for real-valued NPP)
- Time complexity $O(N \log N)$ (sorting)

# The Differencing Algorithm: An Example

Input: $\{8, 7, 6, 5, 4\}$

# The Differencing Algorithm: An Example

Input: $\{8, 7, 6, 5, 4\}$

Iter 1: Replace 8 and 7 by $|8 - 7| = 1$
$\{6, 5, 4, 1\}$

## The Differencing Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | Replace 8 and 7 by $\|8 - 7\| = 1$ |
| | $\{6, 5, 4, 1\}$ |
| Iter 2: | Replace 6 and 5 by $\|6 - 5\| = 1$ |
| | $\{4, 1, 1\}$ |

# The Differencing Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | Replace 8 and 7 by $\lvert 8 - 7\rvert = 1$ |
| | $\{6, 5, 4, 1\}$ |
| Iter 2: | Replace 6 and 5 by $\lvert 6 - 5\rvert = 1$ |
| | $\{4, 1, 1\}$ |
| Iter 3: | Replace 4 and 1 by $\lvert 4 - 1\rvert = 3$ |
| | $\{3, 1\}$ |

# The Differencing Algorithm: An Example

| | |
|---|---|
| Input: | $\{8, 7, 6, 5, 4\}$ |
| Iter 1: | Replace 8 and 7 by $\lvert 8 - 7 \rvert = 1$ |
| | $\{6, 5, 4, 1\}$ |
| Iter 2: | Replace 6 and 5 by $\lvert 6 - 5 \rvert = 1$ |
| | $\{4, 1, 1\}$ |
| Iter 3: | Replace 4 and 1 by $\lvert 4 - 1 \rvert = 3$ |
| | $\{3, 1\}$ |
| Iter 4: | Replace 3 and 1 by $\lvert 3 - 1 \rvert = 2$ |
| | $\{2\}$ |
| Discrepancy: | 2 |

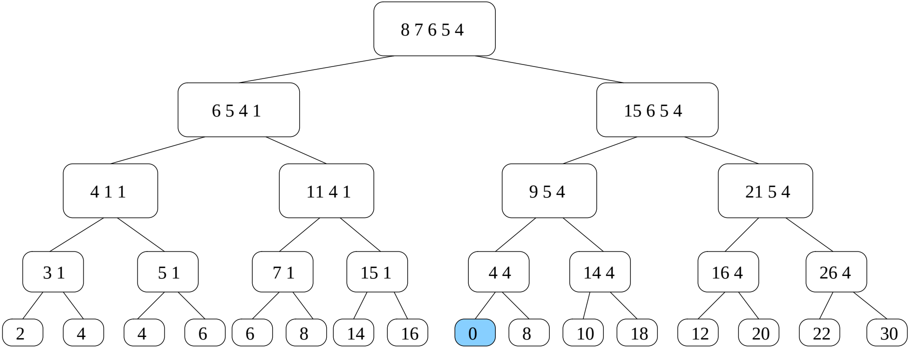
# The Differencing Algorithm: An Example

| | | |
|---|---|---|
| Input: | $\{8,7,6,5,4\}$ | |
| Iter 1: | Replace 8 and 7 by $\lvert 8-7\rvert = 1$ | $\{7,5,4\}\{8,6\}$ |
| | $\{6,5,4,1\}$ | |
| Iter 2: | Replace 6 and 5 by $\lvert 6-5\rvert = 1$ | $\{5,4\}\{6,1\}$ |
| | $\{4,1,1\}$ | |
| Iter 3: | Replace 4 and 1 by $\lvert 4-1\rvert = 3$ | $\{4\}\{1,1\}$ |
| | $\{3,1\}$ | |
| Iter 4: | Replace 3 and 1 by $\lvert 3-1\rvert = 2$ | $\{3\}\{1\}$ |
| | $\{2\}$ | |
| Discrepancy: | 2 | $\{2\}\{\}$ |

# Complete Algorithms

- Complete greedy algorithm
  - First try to put the largest number to the set with smaller sum.
  - Then try to put it to the other set.
  - I.e. search a tree of all $2^N$ possible partitions
- Complete differencing algorithm
  - First try to put the two largest numbers $a_i$ and $a_j$ to different sets
    $\Longrightarrow$ Replace them by $|a_i - a_j|$
  - Then try to put them into the same set
    $\Longrightarrow$ Replace them by $a_i + a_j$

# Complete Differencing Method: Example

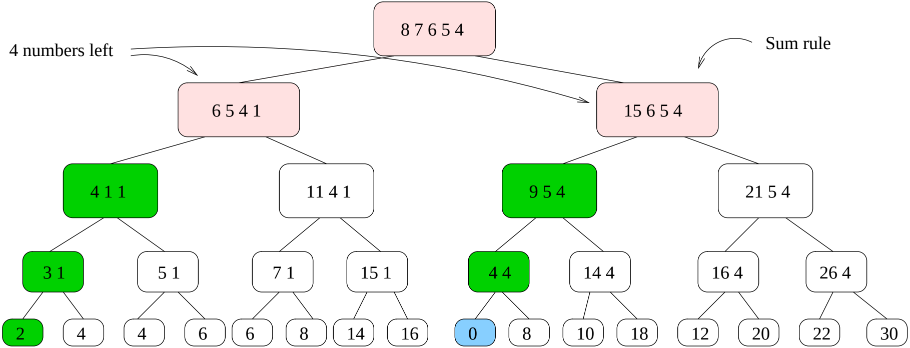


# Complete Differencing Method with Pruning

Pruning rules for the complete differencing method:

- If less than 5 numbers left, take the left branch
- If largest number $\geq$ sum of the rest of the numbers, place the largest number in one set and the others in the other set
- If a perfect partition has been found, stop.

# Complete Differencing Method with Pruning: Example



# Outline

The Number Partitioning Problem (NPP)

Algorithms for NPP

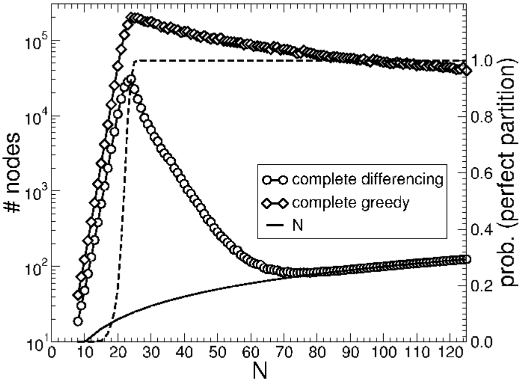
Phase Transition in NPP

# Parameterization of NPP Instance Ensemble

- NPP instance: $N$ random numbers $a_i$
- $a_i < A$
- $A = 2^{\kappa N} \implies$ worst case complexity of any (known) algorithm exponential in $N$ for all $\kappa > 0$
- Typical complexity of NPP depends on $\kappa$

# Phase Transition

- Instances: $N$ random 20-bit integers
- $A = 2^{20} \implies \kappa = 20/N$
- Phase transition at $\kappa_c \approx 20/24 \approx 0.83$




(From: *S. Mertens: The Easiest Hard Problem: Number Partitioning, in: A.G. Percus, G. Istrate, C. Moore, eds., Computational Complextity and Statistical Physics, Oxford University Press (2006), p. 125-139*)

# Pseudo Polynomiality of NPP

- $a_i < A$
- Discrepancy has at most $NA$ different values
- The search space can be explored in time $O(N^2A)$
- A polynomial algorithm?

# Pseudo Polynomiality of NPP

- $a_i < A$
- Discrepancy has at most $NA$ different values
- The search space can be explored in time $O(N^2 A)$
- A polynomial algorithm?
  - Input size $O(N \log A)$
    $\Longrightarrow$ Running time is exponential in the size of the input!
- NP-hardness: input numbers allowed to be exponentially large in N
- If $N$ is increased for constant $A$, NPP is eventually not NP-hard anymore

# Analytical Results

(*S. Mertens: Phase Transition in the Number Partition Problem, Phys. Rev. Lett.* **81**, *4281 (1998)*)

- Analyze the average number of perfect partitions $\Omega(0)$
- It can be shown that

$$\log_2 \Omega(0) = N(\kappa_c - \kappa)$$

with

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

- $\kappa < \kappa_c \implies$ Exponential number of perfect partitions $\implies$ Easy phase
- $\kappa > \kappa_c \implies$ No perfect partitions $\implies$ Hard phase

# Characteristics of the Easy Phase

- Exponential number of perfect partitions
- $\kappa$ decreases $\implies$ the number of perfect partitions increases
- For small $\kappa$ perfect partitions found by heuristic algorithms

# Characteristics of the Hard Phase

- Exponentially small probability of perfect partitions $\implies$ optimum almost always unique
- Heuristic algorithms no better than blind search

# Summary

- Number partitioning problem
- Two heuristic algorithms
  - Greedy
  - Differencing
- Phase transition in average complexity $\iff$
  Phase transition in probability of perfect partitions
  - Easy phase where heuristic algorithms are effective
  - Hard phase where heuristic algorithms are no better than blind search

# Analytical Results – Proof

(*S. Mertens: The Easiest Hard Problem: Number Partitioning, in: A.G. Percus, G. Istrate, C. Moore, eds., Computational Complextity and Statistical Physics, Oxford University Press (2006), p. 125-139*)

- Code a partition $A$ with binary variables $s_i = \pm 1$:
  - $s_i = +1 \implies i \in A$ and $s_i = -1 \implies i \notin A$
- The cost function: $E = |D(s)|$ where

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

- $D$ can be interpreted as distance to the origin of a random walker in one dimension who takes steps to the right ($s_i = +1$) or left ($s_i = -1$) with random step sizes ($a_i$).
- Average number of walks ending at $D$:

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

# Analytical Results – Proof (continued)

- For a fixed walk $\{s_i\}$ and large $N$ the distance to the origin is Gaussian with mean
$$\langle D \rangle = M \langle a \rangle$$
and variance
$$\left\langle D^2 \right\rangle - \langle D \rangle^2 = M^2(\left\langle a^2 \right\rangle - \langle a \rangle^2)$$
where $M = \sum_j s_j$.
- Averaging over the random walk $\{s_i\}$:
$$[\langle D \rangle] = [M] \langle a \rangle = 0$$
$$\left[\left\langle D^2 \right\rangle\right] - [\langle D \rangle]^2 = \left[M^2\right] \left\langle a^2 \right\rangle = N \left\langle a^2 \right\rangle$$
- The probability of ending the walk at distance $D$
$$p(D) = \frac{1}{\sqrt{2\pi N \left\langle a^2 \right\rangle}} \exp\left(-\frac{D^2}{2N \left\langle a^2 \right\rangle}\right)$$

# Analytical Results – Proof (continued)

- Note: The walk ends at even (odd) numbers for $\sum a_i$ even (odd)
- Average number of walks ending at $D$

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

- Substituting

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

we get

$$\log_2 \Omega(0) = N(\kappa_c - \kappa)$$

with

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