

# New generic algorithms for hard knapsacks

Nick Howgrave-Graham<sup>1</sup>   Antoine Joux<sup>2</sup>

35 Park St, Arlington, MA 02474  
nickhg@gmail.com

DGA and Université de Versailles Saint-Quentin-en-Yvelines  
antoine.joux@m4x.org

ESC 2010, Remich, January 12<sup>th</sup>, 2010

# Knapsack problems

- ▶ Given positive integers  $S$  and  $a_1, \dots, a_n$ , consider equation:

$$S = \sum_{i=1}^n \epsilon_i a_i,$$

- ▶ Decision knapsack problem:

Does there exist a  $\{0, 1\}$  solution?

- ▶ Computational knapsack problem: Find it!

# Hardness

- ▶ Hard in general:
  - ▶ The decision knapsack problem is NP-complete
  - ▶ The computational knapsack problem is NP-hard
- ▶ Easy for a large class:
  - ▶ Low density knapsacks
  - ▶ Definition of the density:

$$d = \frac{n}{\log_2 \max_i a_i}$$

- ▶ Solved by lattice reduction oracles when  $d < 0.94$

# Elementary algorithms for generic knapsacks

- ▶ Exhaustive search: Time  $O(2^n)$  operations
- ▶ Birthday algorithm based on:

$$\sum_{i=1}^{n/2} \epsilon_i a_i = S - \sum_{i=n/2+1}^n \epsilon_i a_i.$$

Time  $O(2^{n/2})$ , memory  $O(2^{n/2})$ .

# State of the art

The screenshot shows a web browser window displaying the 'Open Problem Garden' website. The page title is 'Exponential Algorithms for Knapsack'. The browser's address bar shows the URL: [http://ogarden.imm.ac.fy.de/?q=open\\_exponential\\_algorithms\\_for\\_knapsack](http://ogarden.imm.ac.fy.de/?q=open_exponential_algorithms_for_knapsack). The website header includes the 'Open Problem Garden' logo and navigation links like 'Home', 'Subject', 'Theoretical Comp. Sci.', and 'Algorithms'. A search bar is present in the top left. The main content area features a 'Conjecture' section with a light blue background, stating the famous 0-1 Knapsack problem and asking for the best known worst-case algorithm's runtime. Below this is a 'Bibliography' section with a reference to Richard Schroeppel and Adi Shamir's algorithm. To the right, a metadata box lists 'Importance: Medium', 'Author(s): Lipton, Dick', 'Subject: Theoretical Computer Science', 'Keywords: Algorithm construction, Exponential-time algorithm, Knapsack', and 'Posted by: dick lipton' on 'February 4th, 2009'. The page also includes 'Recent Activity' with entries for 'Subset sum or 0-1?' and '0-1 Knapsack?'. The footer contains information about the site's power (Drupal), hosting (IMPA), and content distribution (Creative Commons BY-NC-SA).

# Shamir-Schroeppel algorithm

- ▶ To use birthday algorithm, it suffices to enumerate the set

$$\mathcal{S}^{(1)} = \left\{ \sum_{i=1}^{n/2} \epsilon_i \mathbf{a}_i \right\}$$

in increasing order.

- ▶ Can be done with less memory
- ▶ Yields state of the art for generic knapsacks:  
Time  $O(2^{n/2})$ , memory  $O(2^{n/4})$ .

# Description of Shamir-Schroeppel algorithm

- ▶ Define the following sets (of size  $2^{n/4}$ ):

$$\mathcal{S}_L^{(1)} = \left\{ \sum_{i=1}^{n/4} \epsilon_i \mathbf{a}_i \right\}$$
$$\mathcal{S}_R^{(1)} = \left\{ \sum_{i=n/2+1}^{n/2} \epsilon_i \mathbf{a}_i \right\}$$

- ▶ Any  $\sigma \in \mathcal{S}^{(1)}$  can be written  $\sigma = \sigma_L + \sigma_R$
- ▶ Moreover:

$$\sigma_L + \sigma_R < \sigma_L + \sigma'_R \quad \text{iff} \quad \sigma_R < \sigma'_R$$

# Shamir-Schroeppel continued

- ▶ If  $S_R^{(1)}$  is sorted:
  - ▶ Finding successor of  $\sigma_L + \sigma_R$  with same  $\sigma_L$  is easy
- ▶ How to interlace different  $\sigma_L$  values?
  
- ▶ Shamir and Schroeppel idea:
  - ▶ Create set of triples  $(\sigma_L + \sigma_R^{(0)}, \sigma_L, \sigma_R^{(0)})$
  - ▶ Repeat:
    - ▶ Extract triple with minimum  $\sigma_L + \sigma_R$  from the set
    - ▶ Update into successor  $(\sigma_L + \sigma'_R, \sigma_L, \sigma'_R)$
  
  - ▶ Require specific data structure (heap or balanced tree)

# A more practical variant of Shamir-Schroeppel

- ▶ Let  $M$  be a prime near  $2^{n/4}$
- ▶ For a knapsack solution  $\sigma_L^{(1)}, \sigma_R^{(1)}, \sigma_L^{(2)}$  and  $\sigma_R^{(2)}$ , we have:

$$\sigma_L^{(1)} + \sigma_R^{(1)} \equiv S - \sigma_L^{(2)} - \sigma_R^{(2)} \pmod{M}.$$

- ▶ Let  $\sigma_M$  denotes this “middle value”
- ▶ Algorithm becomes:

- ▶ For each possible value of  $\sigma_M$ :
  - ▶ For each  $\sigma_L^{(1)}$ , find all  $\sigma_R^{(1)}$  such that:

$$\sigma_L^{(1)} + \sigma_R^{(1)} \equiv \sigma_M \pmod{M}.$$

- ▶ For each  $\sigma_L^{(2)}$ , find all  $\sigma_R^{(2)}$  such that:

$$\sigma_L^{(2)} + \sigma_R^{(2)} \equiv S - \sigma_M \pmod{M}.$$

- ▶ Match the above two lists for exact solution (not only mod  $M$ )

# Shamir-Schroeppel for unbalanced knapsacks

- ▶ Knapsack with extra information:

$$\sum_{i=1}^n \epsilon_i = \alpha n$$

- ▶ Build sets of  $\alpha n/4$  elements in each quarter
- ▶ Need “good decomposition”  $\Rightarrow$  extra polynomial factor
- ▶ Time and memory:

$$\binom{n/2}{\alpha n/2} \approx \left( \frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}} \right)^{n/2}$$
$$\binom{n/4}{\alpha n/4} \approx \left( \frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}} \right)^{n/4} .$$

# Going beyond Shamir-Schroepel

- ▶ For simplicity, assume exactly  $n/2$  elements  $a_i$  appear in  $S$
- ▶ Consider decompositions:

$$S = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4,$$

where each  $\sigma_j$  is a sum of exactly  $n/8$  values (among  $n$ ).

- ▶ A given solution of the knapsack can be split into:

$$\binom{n/2}{n/8 \ n/8 \ n/8 \ n/8} = \frac{(n/2)!}{(n/8)!^4} \approx 2^n$$

decompositions  $\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4$ .

## Going beyond Shamir-Schroepel (2)

- ▶ Fix modulus  $M$ , random values  $R_1$ ,  $R_2$  and  $R_3$
- ▶ Search only decompositions with:

$$\begin{array}{ll} \sigma_1 \equiv R_1 \pmod{M} & \sigma_2 \equiv R_2 \pmod{M} \\ \sigma_3 \equiv R_3 \pmod{M} & \sigma_4 \equiv S - R_1 - R_2 - R_3 \pmod{M} \end{array}$$

- ▶ Since first 3 conditions imply the fourth:
  - ▶ Expect one decomposition on average when  $M \approx 2^{n/3}$ .

# First algorithm

- ▶ Given  $M$ ,  $R_1$ ,  $R_2$  and  $R_3$
- ▶ Solve four unbalanced knapsacks with  $\alpha = 1/8$  on  $n$  elements modulo  $M$
- ▶ Expect:

$$\binom{n}{\alpha n} \cdot M^{-1} \approx \left( \frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}} \right)^n \cdot M^{-1} \approx 2^{0.210 n}$$

solutions for each.

- ▶ Costs time  $\approx 2^{0.272 n}$  (and memory  $\approx 2^{0.136 n}$ )
- ▶ Use Shamir-Schroepel again to paste the four sets of solutions together:
  - ▶ Costs time  $\approx 2^{0.420 n}$  and memory  $\approx 2^{0.210 n}$

# Improving the first algorithm

- ▶ Instead of  $M \approx 2^{n/3}$ , choose  $M \approx 2^{\beta n}$  with  $\beta \leq 1/3$
- ▶ Expect  $2^{(1-3\beta)n}$  decompositions
- ▶ Solving the four unbalanced knapsacks costs the same
- ▶ But now expect:

$$\binom{n}{\alpha n} \cdot M^{-1} \approx \left( \frac{1}{\alpha^\alpha \cdot (1-\alpha)^{1-\alpha}} \right)^n \cdot M^{-1} \approx 2^{(0.544-\beta)n}$$

solutions for each.

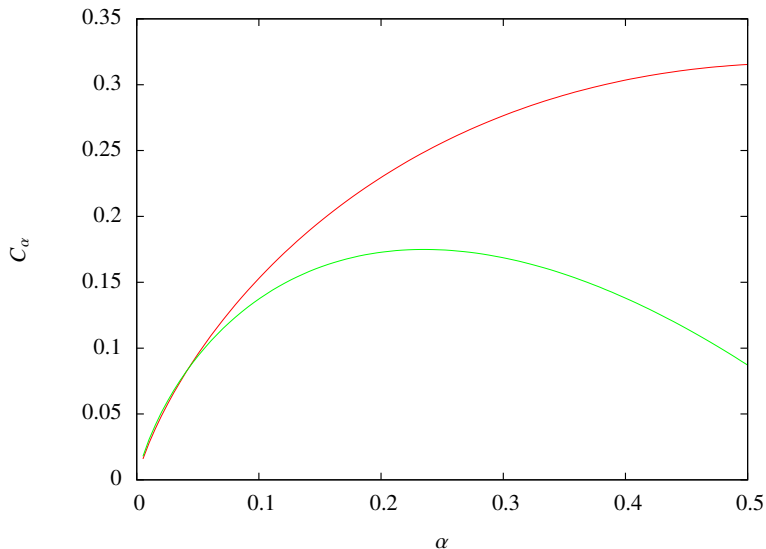
- ▶ Use Shamir-Schroeppel again to paste the four sets of solutions together:
  - ▶ But early abort: **Stop on first decomposition**
  - ▶ Costs time  $\approx 2^{(2(0.544-\beta)+3\beta-1)n} = 2^{(0.088+\beta)n}$   
and memory  $\approx 2^{(0.544-\beta)n}$
- ▶ With  $\beta \approx 0.228$ , we have time and memory  $\approx 2^{0.315n}$

## Also works for unbalanced knapsacks

- ▶ Balancing time and memory, complexity  $\approx 2^{C_\alpha n}$  with:

$$C_\alpha = \max \left( \begin{array}{l} \frac{3}{2} \log_2 \left( \frac{4}{\alpha^{\alpha/4} \cdot (4-\alpha)^{(4-\alpha)/4}} \right) - \alpha, \\ 2 \log_2 \left( \frac{4}{\alpha^{\alpha/4} \cdot (4-\alpha)^{(4-\alpha)/4}} \right) - 2\alpha \end{array} \right).$$

# Complexity for unbalanced knapsacks



# How good is this?

- ▶ Better than Shamir-Schroeppel as time goes.
- ▶ Worse than Shamir-Schroeppel as memory goes.
- ▶ Is it practical ?

# How good is this?

- ▶ Better than Shamir-Schroeppel as time goes.
- ▶ Worse than Shamir-Schroeppel as memory goes.
- ▶ Is it practical ?
- ▶ Implementation for  $n = 96$ 
  - ▶ Shamir-Schroeppel 1:
    - ▶ Time 1 500 days, Memory 1.8 Gbytes
  - ▶ Shamir-Schroeppel 2:
    - ▶ Time 4 400 days, Memory 300 Mbytes
  - ▶ Our algorithm:
    - ▶ Time 5 days, Memory 2.7 Gbytes

## Going further

- ▶ Instead of cutting in four, cut in two
- ▶ Choose  $M \approx 2^{\gamma n}$  with  $\gamma \leq 1/2$
- ▶ Choose  $R \pmod{M}$  and write  $S = \sigma_1 + \sigma_2$ 
  - ▶ With  $\sigma_1 \equiv R$  and  $\sigma_2 \equiv S - R \pmod{M}$
- ▶ Solve two unbalanced knapsacks with  $\alpha = 1/4$  on  $n$  elements modulo  $M$ 
  - ▶ Use first improved algorithm instead of Shamir-Schroeppel

# Going further

- ▶ Instead of cutting in four, cut in two
- ▶ Choose  $M \approx 2^{\gamma n}$  with  $\gamma \leq 1/2$
- ▶ Choose  $R \pmod{M}$  and write  $S = \sigma_1 + \sigma_2$ 
  - ▶ With  $\sigma_1 \equiv R$  and  $\sigma_2 \equiv S - R \pmod{M}$
- ▶ Solve two unbalanced knapsacks with  $\alpha = 1/4$  on  $n$  elements modulo  $M$ 
  - ▶ Use first improved algorithm instead of Shamir-Schroeppel
- ▶ With  $\gamma \approx 0.555$ , sub-knapsacks cost  $\approx 2^{0.256 n}$
- ▶ Complete algorithm with:
  - ▶ Time  $\approx 2^{0.311 n}$ , Memory  $\approx 2^{0.256 n}$

## Practical parameters for $n = 96$

- ▶ Outer  $M = (2^{23} + 9) \cdot 1009 \cdot 50\,021 = 423\,383\,474\,055\,613$
- ▶ For sub-knapsacks inner  $M = 50\,021$ 
  - ▶ Also replace innermost Shamir-Schroeppel by simple birthday paradox
- ▶ For assembling two solutions of sub-knapsacks:
  - ▶ Use middle values modulo 1009
- ▶ For a good decomposition of 48/96 in two 24/48
  - ▶ Runtime: 6 to 12 hours
  - ▶ Need to do it 6.2 times

## Practical parameters for $n = 96$

- ▶ Outer  $M = (2^{23} + 9) \cdot 1009 \cdot 50\,021 = 423\,383\,474\,055\,613$
- ▶ For sub-knapsacks inner  $M = 50\,021$ 
  - ▶ Also replace innermost Shamir-Schroeppel by simple birthday paradox
- ▶ For assembling two solutions of sub-knapsacks:
  - ▶ Use middle values modulo 1009
- ▶ For a good decomposition of 48/96 in two 24/48
  - ▶ Runtime: 6 to 12 hours
  - ▶ Need to do it 6.2 times
  
- ▶ Complete algorithm with:
  - ▶ Time 3 days, Memory 500 Mbytes

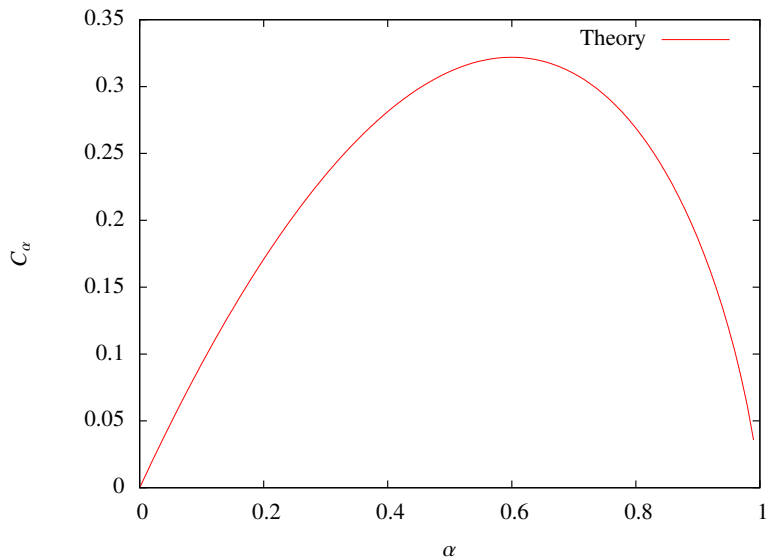
# Could we do better?

- ▶ In theory:
  - ▶ Split the knapsack in two
  - ▶ For  $\alpha n$  elements in  $n$ :
    - ▶ Get  $\binom{\alpha n}{\alpha n/2}$  decompositions
    - ▶ At best, would yield complexity:

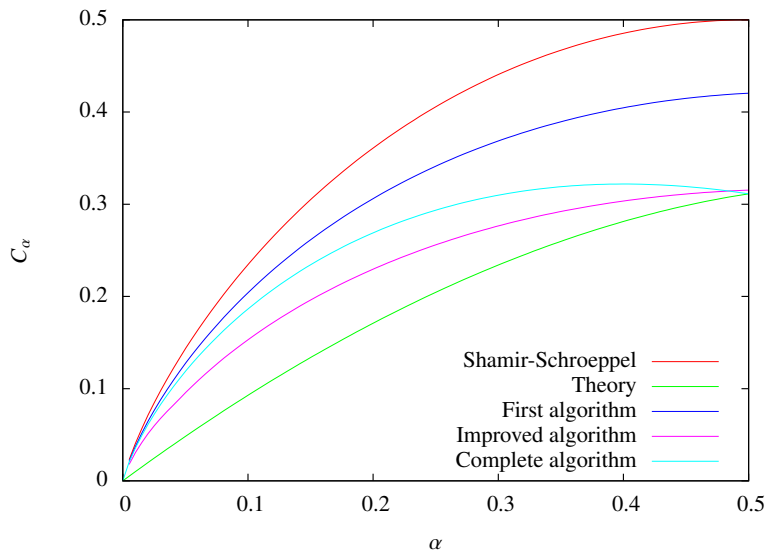
$$\frac{\binom{n}{\alpha n/2}}{\binom{\alpha n}{\alpha n/2}}$$

- ▶ For  $\alpha = 1/2$ , this is  $\approx 2^{0.311 n}$ .

# Could we do better?



# Could we do better?



# Generalizations

- ▶ Noisy knapsacks
- ▶ Modular knapsacks
- ▶ Vectorial knapsacks

# Conclusion