

Approximation Algorithm for Extensible Bin Packing

Lasse Nielsen

10th April 2008

Defining the problem

Definition

Example

Simplifying

Creating an FPTAAS

The steps

Step1

Step2

Step3

Step4

Summing up

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.
- ▶ Number of bins is part of the problem-instance, not part of the solution.

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.
- ▶ Number of bins is part of the problem-instance, not part of the solution.
- ▶ We allow a bin to have more than unit-size, but this will be punished by increasing the cost of the bin.

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.
- ▶ Number of bins is part of the problem-instance, not part of the solution.
- ▶ We allow a bin to have more than unit-size, but this will be punished by increasing the cost of the bin.
- ▶ The cost of bin k is $\max(1, \ell(k))$.

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.
- ▶ Number of bins is part of the problem-instance, not part of the solution.
- ▶ We allow a bin to have more than unit-size, but this will be punished by increasing the cost of the bin.
- ▶ The cost of bin k is $\max(1, \ell(k))$.
- ▶ The objective is to distribute the objects in the bins, such that the sum of the costs of the bins is minimized.

Defining Extensible Bin Packing Problem

- ▶ Similar to Bin Packing.
- ▶ Number of bins is part of the problem-instance, not part of the solution.
- ▶ We allow a bin to have more than unit-size, but this will be punished by increasing the cost of the bin.
- ▶ The cost of bin k is $\max(1, \ell(k))$.
- ▶ The objective is to distribute the objects in the bins, such that the sum of the costs of the bins is minimized.
- ▶ This can be expressed as an integer programming problem.

An example

- ▶ There are two bins ($m = 2$), and four objects ($n = 4$).
 $x_1 = \frac{2}{3}$, $x_2 = x_3 = \frac{1}{2}$ and $x_4 = \frac{1}{6}$.

An example

- ▶ There are two bins ($m = 2$), and four objects ($n = 4$).
 $x_1 = \frac{2}{3}$, $x_2 = x_3 = \frac{1}{2}$ and $x_4 = \frac{1}{6}$.
- ▶ Solution one:
Bin 1: x_1, x_2 . Cost: $\frac{7}{6}$.
Bin 2: x_3, x_4 . Cost: 1.
Total cost: $2 + \frac{1}{6}$.

An example

- ▶ There are two bins ($m = 2$), and four objects ($n = 4$).
 $x_1 = \frac{2}{3}$, $x_2 = x_3 = \frac{1}{2}$ and $x_4 = \frac{1}{6}$.
- ▶ Solution one:
Bin 1: x_1, x_2 . Cost: $\frac{7}{6}$.
Bin 2: x_3, x_4 . Cost: 1.
Total cost: $2 + \frac{1}{6}$.
- ▶ Solution two:
Bin 1: x_2, x_3 . Cost: 1.
Bin 2: x_1, x_4 . Cost: 1.
Total cost: 2.

The first observation

- ▶ The cost of a solution, is the sum of all the objects, plus the sum of the unused space in the bins.

Simplifying assumptions

- ▶ We can assume that all objects are greater than $\frac{\epsilon}{1+\epsilon}$.

Simplifying assumptions

- ▶ We can assume that all objects are greater than $\frac{\epsilon}{1+\epsilon}$.
- ▶ We can assume that all objects are smaller than 1.

Simplifying assumptions

- ▶ We can assume that all objects are greater than $\frac{\epsilon}{1+\epsilon}$.
- ▶ We can assume that all objects are smaller than 1.
- ▶ We can assume that the sum of the sizes of all the objects is less than $2m$.

Simplifying assumptions

- ▶ We can assume that all objects are greater than $\frac{\epsilon}{1+\epsilon}$.
- ▶ We can assume that all objects are smaller than 1.
- ▶ We can assume that the sum of the sizes of all the objects is less than $2m$.
- ▶ There will always be an optimal solution such that all bins have cost less than 3.

The steps of the algorithm

- ▶ Step 1: Rounding
- ▶ Step 2: Transform to a new integer programming problem (Redefinition)
- ▶ Step 3: Weaken the problem to a linear programming problem
- ▶ Step 4: Solving the linear programming problem, and using this to obtain a solution for the original problem.

Step 1: Rounding

- ▶ The object sizes are rounded to the values

$$s_1 = \left\lfloor \frac{(1+\varepsilon)}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad s_2 = \left\lfloor \frac{(1+\varepsilon)^2}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad \dots, s_{\rho(\varepsilon)} = 1.$$

Step 1: Rounding

- ▶ The object sizes are rounded to the values

$$s_1 = \left\lfloor \frac{(1+\varepsilon)}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad s_2 = \left\lfloor \frac{(1+\varepsilon)^2}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad \dots, \quad s_{\rho(\varepsilon)} = 1.$$

- ▶ Lemma: $\rho(\varepsilon) \leq \left\lceil \frac{1}{\varepsilon} \cdot \ln \left(\frac{1}{\varepsilon} \right) + \frac{1}{\varepsilon} \right\rceil$.

Step 1: Rounding

- ▶ The object sizes are rounded to the values

$$s_1 = \left\lfloor \frac{(1+\varepsilon)}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad s_2 = \left\lfloor \frac{(1+\varepsilon)^2}{\varepsilon} \right\rfloor \cdot \varepsilon^2, \quad \dots, s_{\rho(\varepsilon)} = 1.$$

- ▶ Lemma: $\rho(\varepsilon) \leq \left\lceil \frac{1}{\varepsilon} \cdot \ln\left(\frac{1}{\varepsilon}\right) + \frac{1}{\varepsilon} \right\rceil$.
- ▶ Lemma: If x_i is rounded to s_j then $x_i \cdot (1 + \varepsilon)^2 \geq s_j$ and $s_j \cdot (1 + \varepsilon)^2 \geq x_i$.

Step 2: Redefinition

- ▶ Similar to Lemma 9.4 from Approximation Algorithms.

Step 2: Redefinition

- ▶ Similar to Lemma 9.4 from Approximation Algorithms.
- ▶ Since there are $\rho(\varepsilon)$ types of objects, and we can put at most $3 \cdot \frac{1}{\varepsilon}$ objects in a bin, there are at most $\binom{\rho(\varepsilon) + 3 \cdot \frac{1}{\varepsilon}}{\rho(\varepsilon)}$ different ways to pack a bin.

Step 2: Redefinition

- ▶ Similar to Lemma 9.4 from Approximation Algorithms.
- ▶ Since there are $\rho(\varepsilon)$ types of objects, and we can put at most $3 \cdot \frac{1}{\varepsilon}$ objects in a bin, there are at most $\binom{\rho(\varepsilon) + 3 \cdot \frac{1}{\varepsilon}}{\rho(\varepsilon)}$ different ways to pack a bin.
- ▶ This is independent of the problem instance, but exponential in $\frac{1}{\varepsilon}$.

Step 2: Redefinition

- ▶ Similar to Lemma 9.4 from Approximation Algorithms.
- ▶ Since there are $\rho(\varepsilon)$ types of objects, and we can put at most $3 \cdot \frac{1}{\varepsilon}$ objects in a bin, there are at most $\binom{\rho(\varepsilon) + 3 \cdot \frac{1}{\varepsilon}}{\rho(\varepsilon)}$ different ways to pack a bin.
- ▶ This is independent of the problem instance, but exponential in $\frac{1}{\varepsilon}$.
- ▶ Selecting m of the found bins such that all the objects are used, and the total cost is minimized can be expressed as an integer programming problem.

The new problem

$$\text{minimize } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right) = \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max(1, \ell(b_i))$$

$$\text{subject to } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot b_i(j) \geq n_j \text{ for } j = 1 \dots \rho(\varepsilon)$$

$$\text{and } \sum_{i=1}^{\xi(\varepsilon)} z_i = m$$

$$\text{with } z_i \in \mathbb{N}_0 \text{ for } i = 1 \dots \xi(\varepsilon)$$

The new problem

$$\text{minimize } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right) = \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max(1, \ell(b_i))$$

$$\text{subject to } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot b_i(j) \geq n_j \text{ for } j = 1 \dots \rho(\varepsilon)$$

$$\text{and } \sum_{i=1}^{\xi(\varepsilon)} z_i = m$$

with $z_i \in \mathbb{N}_0$ for $i = 1 \dots \xi(\varepsilon)$

- Write this on the blackboard.

Step 3: Weakening

- ▶ We wish to obtain a linear programming problem.

Step 3: Weakening

- ▶ We wish to obtain a linear programming problem.
- ▶ We can change

$$\sum_{i=1}^{\xi(\varepsilon)} z_i = m \quad \text{to} \quad \sum_{i=1}^{\xi(\varepsilon)} z_i \geq m$$

because merging bins creates a solution with no greater cost.

Step 3: Weakening

- ▶ We wish to obtain a linear programming problem.
- ▶ We can change

$$\sum_{i=1}^{\xi(\varepsilon)} z_i = m \quad \text{to} \quad \sum_{i=1}^{\xi(\varepsilon)} z_i \geq m$$

because merging bins creates a solution with no greater cost.

- ▶ By allowing z_i to assume non-integer values, we can only improve OPT.

We will still need to convert the solution to an integer-solution.

The new problem

$$\text{minimize } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right) = \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot \max(1, \ell(b_i))$$

$$\text{subject to } \sum_{i=1}^{\xi(\varepsilon)} z_i \cdot b_i(j) \geq n_j \text{ for } j = 1 \dots \rho(\varepsilon)$$

$$\text{and } \sum_{i=1}^{\xi(\varepsilon)} z_i \geq m$$

$$\text{with } z_i \geq 0 \text{ for } i = 1 \dots \xi(\varepsilon)$$

Step 4: Using the solution

- ▶ This problem can be solved in polynomial time (of the problem size).

Step 4: Using the solution

- ▶ This problem can be solved in polynomial time (of the problem size).
- ▶ The number of inequalities is $\rho(\varepsilon) + 1$.

Step 4: Using the solution

- ▶ This problem can be solved in polynomial time (of the problem size).
- ▶ The number of inequalities is $\rho(\varepsilon) + 1$.
- ▶ The number of non-integer values in the solution is at most $\rho(\varepsilon) + 1$.

Step 4: Using the solution

- ▶ This problem can be solved in polynomial time (of the problem size).
- ▶ The number of inequalities is $\rho(\varepsilon) + 1$.
- ▶ The number of non-integer values in the solution is at most $\rho(\varepsilon) + 1$.
- ▶ Problem: Because of the size of the constraints we the runningtime is not polynomial in $\frac{1}{\varepsilon}$.

Step 4b: Solving in polynomial time

- ▶ Consider the dual problem:

$$\text{maximize } t \cdot m + \sum_{j=1}^{\rho(\varepsilon)} u_j \cdot n_j$$

$$\text{subject to } t + \sum_{i=1}^{\rho(\varepsilon)} u_j \cdot b_i(j) \leq \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right)$$

for $i = 1, 2, \dots, \xi(\varepsilon)$

with $t, u_j \geq 0$ for $i = 1 \dots \rho(\varepsilon)$

Step 4b: Solving in polynomial time

- If we can solve the *strong separation problem*

Given $t, u_1, \dots, u_{\rho(\varepsilon)}$

Determine if $\exists i. t + \sum_{j=1}^{\rho(\varepsilon)} u_j \cdot b_i(j) > \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right)$

If so, find $b_i(1) \dots b_i(\rho(\varepsilon))$

in polynomial time, then we can solve the entire problem in polynomial time.

Step 4b: Solving in polynomial time

- ▶ If we can solve the *strong separation problem*

Given $t, u_1, \dots, u_{\rho(\varepsilon)}$

Determine if $\exists i. t + \sum_{j=1}^{\rho(\varepsilon)} u_j \cdot b_i(j) > \max \left(1, \sum_{j=1}^{\rho(\varepsilon)} b_i(j) \cdot s_j \right)$

If so, find $b_i(1) \dots b_i(\rho(\varepsilon))$

in polynomial time, then we can solve the entire problem in polynomial time.

- ▶ Because of the structure of the bins, this can be done in polynomial time using dynamic programming.

Summing up

- ▶ We have seen how we given an instance I of an Extensible Bin Packing Problem, and an ε can find a solution with cost at most $(1 + \varepsilon)^2 \cdot (\text{OPT}(I) + 3 \cdot (\lceil \frac{1}{\varepsilon} \cdot \ln(\frac{1}{\varepsilon}) + \frac{1}{\varepsilon} \rceil + 1))$.
- ▶ In Step 2, it was mentioned how to obtain a PTAS.
- ▶ The article states that there is no FPTAS if $NP \neq P$.
- ▶ The article states that the greedy algorithm obtains a factor $\frac{13}{12}$ approximation.