Can we do better?

In the following we show how to obtain a solution where the number of bins is only

$$\mathrm{OPT}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I))) \ .$$

Note that this is usually better than a guarantee of

$$(1+\epsilon)\mathrm{OPT}(I) + 1 \ .$$

Can we do better?

In the following we show how to obtain a solution where the number of bins is only

$$\mathrm{OPT}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I))) \ .$$

Note that this is usually better than a guarantee of

$$(1+\epsilon)\mathrm{OPT}(I) + 1 \ .$$

Can we do better?

In the following we show how to obtain a solution where the number of bins is only

$$\mathrm{OPT}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I))) \ .$$

Note that this is usually better than a guarantee of

$$(1+\epsilon)\mathrm{OPT}(I) + 1 \ .$$

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.

Configuration LP

A possible packing of a bin can be described by an m-tuple $(t_1, \ldots, t_m)$, where t_i describes the number of pieces of size s_i. Clearly,

$$\sum_i t_i \cdot s_i \leq 1 \ .$$

We call a vector that fulfills the above constraint a configuration.

Configuration LP

A possible packing of a bin can be described by an m-tuple $(t_1, \ldots, t_m)$, where t_i describes the number of pieces of size s_i. Clearly,

$$\sum_i t_i \cdot s_i \leq 1 \ .$$

We call a vector that fulfills the above constraint a configuration.

Configuration LP

A possible packing of a bin can be described by an m-tuple $(t_1, \dots, t_m)$, where t_i describes the number of pieces of size s_i. Clearly,

$$\sum_i t_i \cdot s_i \leq 1 \ .$$

We call a vector that fulfills the above constraint a configuration.

Configuration LP

Let N be the number of configurations (exponential).

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

$$
\begin{array}{llrcl}
\min & & \sum_{j=1}^{N} x_j & & \\
\text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\
 & \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \\
 & \forall j \in \{1, \dots, N\} & x_j & \text{integral} &
\end{array}
$$

Configuration LP

Let N be the number of configurations (exponential).

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

$$\begin{array}{llrcl} \min & & \sum_{j=1}^{N} x_j & & \\ \text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\ & \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \\ & \forall j \in \{1, \dots, N\} & x_j & \text{integral} & \end{array}$$

Configuration LP

Let N be the number of configurations (exponential).

Let $T_1, \ldots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

$$
\begin{array}{llrcl}
\min & & \sum_{j=1}^{N} x_j & & \\
\text{s.t.} & \forall i \in \{1 \ldots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\
 & \forall j \in \{1, \ldots, N\} & x_j & \geq & 0 \\
 & \forall j \in \{1, \ldots, N\} & x_j & \text{integral} &
\end{array}
$$

Configuration LP

Let N be the number of configurations (exponential).

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

$$
\begin{array}{llrcl}
\min & & \sum_{j=1}^{N} x_j & & \\
\text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\
& \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \\
& \forall j \in \{1, \dots, N\} & x_j & \text{integral} &
\end{array}
$$

How to solve this LP?

later...

We can assume that each item has size at least $1/\text{SIZE}(I)$.

Harmonic Grouping

- Sort items according to size (monotonically decreasing).
- Process items in this order; close the current group if size of items in the group is at least 2 (or larger). Then open new group.
- I.e., G_1 is the smallest cardinality set of largest items s.t. total size sums up to at least 2. Similarly, for $G_2, \dots, G_{r-1}$.
- Only the size of items in the last group G_r may sum up to less than 2.

Harmonic Grouping

- Sort items according to size (monotonically decreasing).
- Process items in this order; close the current group if size of items in the group is at least 2 (or larger). Then open new group.
- I.e., G_1 is the smallest cardinality set of largest items s.t. total size sums up to at least 2. Similarly, for $G_2, \dots, G_{r-1}$.
- Only the size of items in the last group G_r may sum up to less than 2.

Harmonic Grouping

- Sort items according to size (monotonically decreasing).
- Process items in this order; close the current group if size of items in the group is at least 2 (or larger). Then open new group.
- I.e., G_1 is the smallest cardinality set of largest items s.t. total size sums up to at least 2. Similarly, for $G_2, \dots, G_{r-1}$.
- Only the size of items in the last group G_r may sum up to less than 2.

Harmonic Grouping

- Sort items according to size (monotonically decreasing).
- Process items in this order; close the current group if size of items in the group is at least 2 (or larger). Then open new group.
- I.e., G_1 is the smallest cardinality set of largest items s.t. total size sums up to at least 2. Similarly, for $G_2, \ldots, G_{r-1}$.
- Only the size of items in the last group G_r may sum up to less than 2.

Harmonic Grouping

From the grouping we obtain instance I' as follows:

- Round all items in a group to the size of the largest group member.
- Delete all items from group G_1 and G_r.
- For groups $G_2, \ldots, G_{r-1}$ delete $n_i - n_{i-1}$ items.
- Observe that $n_i \geq n_{i-1}$.

Harmonic Grouping

From the grouping we obtain instance I' as follows:

- Round all items in a group to the size of the largest group member.
- Delete all items from group G_1 and G_r.
- For groups $G_2, \ldots, G_{r-1}$ delete $n_i - n_{i-1}$ items.
- Observe that $n_i \geq n_{i-1}$.

Harmonic Grouping

From the grouping we obtain instance I' as follows:

- Round all items in a group to the size of the largest group member.
- Delete all items from group G_1 and G_r.
- For groups $G_2, \ldots, G_{r-1}$ delete $n_i - n_{i-1}$ items.
- Observe that $n_i \geq n_{i-1}$.

Harmonic Grouping

From the grouping we obtain instance I' as follows:

- Round all items in a group to the size of the largest group member.
- Delete all items from group G_1 and G_r.
- For groups $G_2, \ldots, G_{r-1}$ delete $n_i - n_{i-1}$ items.
- Observe that $n_i \geq n_{i-1}$.

Lemma 10

The number of different sizes in I' is at most $\mathrm{SIZE}(I)/2$.

- Each group that survives (recall that G_1 and G_k are deleted) has total size at least 2.
- Hence, the number of surviving groups is at most $\mathrm{SIZE}(I)/2$.
- All items in a group have the same size in I'.

Lemma 10

The number of different sizes in I' is at most $\mathrm{SIZE}(I)/2$.

- Each group that survives (recall that G_1 and G_r are deleted) has total size at least 2.
- Hence, the number of surviving groups is at most $\mathrm{SIZE}(I)/2$.
- All items in a group have the same size in I'.

Lemma 10

The number of different sizes in I' is at most $\mathrm{SIZE}(I)/2$.

- Each group that survives (recall that G_1 and G_r are deleted) has total size at least 2.
- Hence, the number of surviving groups is at most $\mathrm{SIZE}(I)/2$.
- All items in a group have the same size in I'.

Lemma 10

The number of different sizes in I' is at most $\mathrm{SIZE}(I)/2$.

- Each group that survives (recall that G_1 and G_r are deleted) has total size at least 2.
- Hence, the number of surviving groups is at most $\mathrm{SIZE}(I)/2$.
- All items in a group have the same size in I'.

Lemma 11

The total size of deleted items is at most $\mathcal{O}(\log(\text{SIZE}(I)))$.

Lemma 11

The total size of deleted items is at most $\mathcal{O}(\log(\mathrm{SIZE}(I)))$.

- The total size of items in G_1 and G_r is at most 6 as a group has total size at most 3.
- Consider a group G_i that has strictly more items than G_{i-1}.
- It discards $n_i - n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

since the smallest piece has size at most $3/n_i$.

- Summing over all i that have $n_i > n_{i-1}$ gives a bound of at most

$$\sum_{j=1}^{n_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\mathrm{SIZE}(I))) \ .$$

(note that $n_r \le \mathrm{SIZE}(I)$ since we assume that the size of each item is at least $1/\mathrm{SIZE}(I)$).

Lemma 11

The total size of deleted items is at most $\mathcal{O}(\log(\mathrm{SIZE}(I)))$.

- The total size of items in G_1 and G_r is at most 6 as a group has total size at most 3.
- Consider a group G_i that has strictly more items than G_{i-1}.
- It discards $n_i - n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

  since the smallest piece has size at most $3/n_i$.
- Summing over all i that have $n_i > n_{i-1}$ gives a bound of at most

$$\sum_{j=1}^{n_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\mathrm{SIZE}(I))) \ .$$

  (note that $n_r \le \mathrm{SIZE}(I)$ since we assume that the size of each item is at least $1/\mathrm{SIZE}(I)$).

Lemma 11

The total size of deleted items is at most $\mathcal{O}(\log(\text{SIZE}(I)))$.

- The total size of items in G_1 and G_r is at most 6 as a group has total size at most 3.
- Consider a group G_i that has strictly more items than G_{i-1}.
- It discards $n_i - n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

since the smallest piece has size at most $3/n_i$.

- Summing over all i that have $n_i > n_{i-1}$ gives a bound of at most

$$\sum_{j=1}^{n_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\text{SIZE}(I))) \enspace .$$

(note that $n_r \le \text{SIZE}(I)$ since we assume that the size of each item is at least $1/\text{SIZE}(I)$).

Lemma 11

The total size of deleted items is at most $\mathcal{O}(\log(\text{SIZE}(I)))$.

- The total size of items in G_1 and G_r is at most 6 as a group has total size at most 3.
- Consider a group G_i that has strictly more items than G_{i-1}.
- It discards $n_i - n_{i-1}$ pieces of total size at most

$$3\frac{n_i - n_{i-1}}{n_i} \le \sum_{j=n_{i-1}+1}^{n_i} \frac{3}{j}$$

  since the smallest piece has size at most $3/n_i$.
- Summing over all i that have $n_i > n_{i-1}$ gives a bound of at most

$$\sum_{j=1}^{n_{r-1}} \frac{3}{j} \le \mathcal{O}(\log(\text{SIZE}(I))) \enspace .$$

  (note that $n_r \le \text{SIZE}(I)$ since we assume that the size of each item is at least $1/\text{SIZE}(I)$).

Algorithm 1 BinPack

```
if SIZE(I) < 10 then
    pack remaining items greedily
Apply harmonic grouping to create instance I'; pack discarded items in at most O(log(SIZE(I))) bins.
Let x be optimal solution to configuration LP
Pack ⌊x_j⌋ bins in configuration T_j for all j; call the packed instance I_1.
Let I_2 be remaining pieces from I'
Pack I_2 via BinPack(I_2)
```

Analysis

$$\mathrm{OPT_{LP}}(I_1) + \mathrm{OPT_{LP}}(I_2) \le \mathrm{OPT_{LP}}(I') \le \mathrm{OPT_{LP}}(I)$$

Proof:

- Each piece surviving in I' can be mapped to a piece in I of no lesser size. Hence, $\mathrm{OPT_{LP}}(I') \le \mathrm{OPT_{LP}}(I)$
- $\lfloor \lambda_j \rfloor$ is feasible solution for I_1 (even integral).
- $\lambda_j - \lfloor \lambda_j \rfloor$ is feasible solution for I_2.

Analysis

$$\mathrm{OPT}_{\mathrm{LP}}(I_1) + \mathrm{OPT}_{\mathrm{LP}}(I_2) \le \mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$$

Proof:

- Each piece surviving in I' can be mapped to a piece in I of no lesser size. Hence, $\mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$
- $\lfloor x_j \rfloor$ is feasible solution for I_1 (even integral).
- $x_j - \lfloor x_j \rfloor$ is feasible solution for I_2.

Analysis

$$\mathrm{OPT}_{\mathrm{LP}}(I_1) + \mathrm{OPT}_{\mathrm{LP}}(I_2) \le \mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$$

Proof:

- Each piece surviving in I' can be mapped to a piece in I of no lesser size. Hence, $\mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$
- $\lfloor x_j \rfloor$ is feasible solution for I_1 (even integral).
- $x_j - \lfloor x_j \rfloor$ is feasible solution for I_2.

Analysis

$$\mathrm{OPT}_{\mathrm{LP}}(I_1) + \mathrm{OPT}_{\mathrm{LP}}(I_2) \le \mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$$

Proof:

- Each piece surviving in I' can be mapped to a piece in I of no lesser size. Hence, $\mathrm{OPT}_{\mathrm{LP}}(I') \le \mathrm{OPT}_{\mathrm{LP}}(I)$
- $\lfloor x_j \rfloor$ is feasible solution for I_1 (even integral).
- $x_j - \lfloor x_j \rfloor$ is feasible solution for I_2.

Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in I_1.
3. Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mathrm{OPT_{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\mathrm{SIZE}(I))) \cdot L$$

many bins where L is the number of recursion levels.

Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in I_1.
3. Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mathrm{OPT_{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\mathrm{SIZE}(I))) \cdot L$$

many bins where L is the number of recursion levels.

Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in I_1.
3. Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mathrm{OPT}_{\mathrm{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\mathrm{SIZE}(I))) \cdot L$$

many bins where L is the number of recursion levels.

Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in I_1.
3. Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mathrm{OPT}_{\mathrm{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\mathrm{SIZE}(I))) \cdot L$$

many bins where L is the number of recursion levels.

Analysis

Each level of the recursion partitions pieces into three types

1. Pieces discarded at this level.
2. Pieces scheduled because they are in I_1.
3. Pieces in I_2 are handed down to the next level.

Pieces of type 2 summed over all recursion levels are packed into at most $\mathrm{OPT}_{\mathrm{LP}}$ many bins.

Pieces of type 1 are packed into at most

$$\mathcal{O}(\log(\mathrm{SIZE}(I))) \cdot L$$

many bins where L is the number of recursion levels.

Analysis

We can show that $\text{SIZE}(I_2) \le \text{SIZE}(I)/2$. Hence, the number of recursion levels is only $\mathcal{O}(\log(\text{SIZE}(I_{\text{original}})))$ in total.

- The number of non-zero entries in the solution to the configuration LP for I' is at most the number of constraints, which is the number of different sizes ($\le \text{SIZE}(I)/2$).
- The total size of items in I_2 can be at most $\sum_{j=1}^{N} x_j - \lfloor x_j \rfloor$ which is at most the number of non-zero entries in the solution to the configuration LP.

Analysis

We can show that $\text{SIZE}(I_2) \leq \text{SIZE}(I)/2$. Hence, the number of recursion levels is only $\mathcal{O}(\log(\text{SIZE}(I_{\text{original}})))$ in total.

- The number of non-zero entries in the solution to the configuration LP for I' is at most the number of constraints, which is the number of different sizes ($\leq \text{SIZE}(I)/2$).
- The total size of items in I_2 can be at most $\sum_{j=1}^{N} x_j - \lfloor x_j \rfloor$ which is at most the number of non-zero entries in the solution to the configuration LP.

Analysis

We can show that $\text{SIZE}(I_2) \leq \text{SIZE}(I)/2$. Hence, the number of recursion levels is only $\mathcal{O}(\log(\text{SIZE}(I_{\text{original}})))$ in total.

- The number of non-zero entries in the solution to the configuration LP for I' is at most the number of constraints, which is the number of different sizes ($\leq \text{SIZE}(I)/2$).
- The total size of items in I_2 can be at most $\sum_{j=1}^{N} x_j - \lfloor x_j \rfloor$ which is at most the number of non-zero entries in the solution to the configuration LP.

How to solve the LP?

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).

In total we have b_i pieces of size s_i.

Primal

$$\begin{array}{llrcl} \min & & \sum_{j=1}^{N} x_j & & \\ \text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\ & \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \end{array}$$

Dual

$$\begin{array}{llrcl} \max & & \sum_{i=1}^{m} y_i b_i & & \\ \text{s.t.} & \forall j \in \{1, \dots, N\} & \sum_{i=1}^{m} T_{ji} y_i & \leq & 1 \\ & \forall i \in \{1, \dots, m\} & y_i & \geq & 0 \end{array}$$

How to solve the LP?

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).
In total we have b_i pieces of size s_i.

Primal

$$\begin{array}{llrcl} \min & & \sum_{j=1}^{N} x_j & & \\ \text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\ & \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \end{array}$$

Dual

$$\begin{array}{llrcl} \max & & \sum_{i=1}^{m} y_i b_i & & \\ \text{s.t.} & \forall j \in \{1, \dots, N\} & \sum_{i=1}^{m} T_{ji} y_i & \leq & 1 \\ & \forall i \in \{1, \dots, m\} & y_i & \geq & 0 \end{array}$$

How to solve the LP?

Let $T_1, \dots, T_N$ be the sequence of all possible configurations (a configuration T_j has T_{ji} pieces of size s_i).
In total we have b_i pieces of size s_i.

Primal

$$\begin{array}{llrcl} \min & & \sum_{j=1}^N x_j & & \\ \text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^N T_{ji} x_j & \geq & b_i \\ & \forall j \in \{1, \dots, N\} & x_j & \geq & 0 \end{array}$$

Dual

$$\begin{array}{llrcl} \max & & \sum_{i=1}^m y_i b_i & & \\ \text{s.t.} & \forall j \in \{1, \dots, N\} & \sum_{i=1}^m T_{ji} y_i & \leq & 1 \\ & \forall i \in \{1, \dots, m\} & y_i & \geq & 0 \end{array}$$

Separation Oracle

Suppose that I am given variable assignment y for the dual.

How do I find a violated constraint?

I have to find a configuration $T_j = (T_{j1}, \dots, T_{jm})$ that

- is feasible, i.e.,

$$\sum_{i=1}^{m} T_{ji} \cdot s_i \leq 1 ,$$

- and has a large profit

$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

Separation Oracle

Suppose that I am given variable assignment y for the dual.

How do I find a violated constraint?

I have to find a configuration $T_j = (T_{j1}, \dots, T_{jm})$ that

- is feasible, i.e.,

$$\sum_{i=1}^{m} T_{ji} \cdot s_i \leq 1 \ ,$$

- and has a large profit

$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

Separation Oracle

Suppose that I am given variable assignment y for the dual.

How do I find a violated constraint?

I have to find a configuration $T_j = (T_{j1}, \ldots, T_{jm})$ that

- is feasible, i.e.,
$$\sum_{i=1}^{m} T_{ji} \cdot s_i \leq 1 \ ,$$
- and has a large profit
$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

Separation Oracle

Suppose that I am given variable assignment y for the dual.

How do I find a violated constraint?

I have to find a configuration $T_j = (T_{j1}, \ldots, T_{jm})$ that

- is feasible, i.e.,

$$\sum_{i=1}^{m} T_{ji} \cdot s_i \leq 1 \ ,$$

- and has a large profit

$$\sum_{i=1}^{m} T_{ji} y_i > 1$$

But this is the Knapsack problem.

Separation Oracle

We have FPTAS for Knapsack. This means if a constraint is violated with $1+\epsilon' = 1+\frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1-\epsilon)$ of the optimal profit.

The solution we get is feasible for:

Dual′

$$\begin{array}{llrcl} \max & & \sum_{i=1}^{m} y_i b_i & & \\ \text{s.t.} & \forall j \in \{1,\dots,N\} & \sum_{i=1}^{m} T_{ji} y_i & \leq & 1+\epsilon' \\ & \forall i \in \{1,\dots,m\} & y_i & \geq & 0 \end{array}$$

Primal′

$$\begin{array}{llrcl} \min & & (1+\epsilon')\sum_{j=1}^{N} x_j & & \\ \text{s.t.} & \forall i \in \{1\dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \geq & b_i \\ & \forall j \in \{1,\dots,N\} & x_j & \geq & 0 \end{array}$$

Separation Oracle

We have FPTAS for Knapsack. This means if a constraint is violated with $1 + \epsilon' = 1 + \frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1 - \epsilon)$ of the optimal profit.

The solution we get is feasible for:

Dual′

$$\begin{array}{llrcl} \max & & \sum_{i=1}^{m} y_i b_i & & \\ \text{s.t.} & \forall j \in \{1,\ldots,N\} & \sum_{i=1}^{m} T_{ji} y_i & \le & 1+\epsilon' \\ & \forall i \in \{1,\ldots,m\} & y_i & \ge & 0 \end{array}$$

Primal′

$$\begin{array}{llrcl} \min & & (1+\epsilon') \sum_{j=1}^{N} x_j & & \\ \text{s.t.} & \forall i \in \{1 \ldots m\} & \sum_{j=1}^{N} T_{ji} x_j & \ge & b_i \\ & \forall j \in \{1,\ldots,N\} & x_j & \ge & 0 \end{array}$$

Separation Oracle

We have FPTAS for Knapsack. This means if a constraint is violated with $1 + \epsilon' = 1 + \frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1 - \epsilon)$ of the optimal profit.

The solution we get is feasible for:

Dual′

$$
\begin{array}{llrcl}
\max & & \sum_{i=1}^{m} y_i b_i & & \\
\text{s.t.} & \forall j \in \{1, \dots, N\} & \sum_{i=1}^{m} T_{ji} y_i & \le & 1 + \epsilon' \\
 & \forall i \in \{1, \dots, m\} & y_i & \ge & 0
\end{array}
$$

Primal′

$$
\begin{array}{llrcl}
\min & & (1 + \epsilon') \sum_{j=1}^{N} x_j & & \\
\text{s.t.} & \forall i \in \{1 \dots m\} & \sum_{j=1}^{N} T_{ji} x_j & \ge & b_i \\
 & \forall j \in \{1, \dots, N\} & x_j & \ge & 0
\end{array}
$$

Separation Oracle

We have FPTAS for Knapsack. This means if a constraint is violated with $1+\epsilon' = 1+\frac{\epsilon}{1-\epsilon}$ we find it, since we can obtain at least $(1-\epsilon)$ of the optimal profit.

The solution we get is feasible for:

Dual′

$$\begin{array}{lrrcl} \max & & \sum_{i=1}^m y_i b_i & & \\ \text{s.t.} & \forall j \in \{1,\dots,N\} & \sum_{i=1}^m T_{ji} y_i & \le & 1+\epsilon' \\ & \forall i \in \{1,\dots,m\} & y_i & \ge & 0 \end{array}$$

Primal′

$$\begin{array}{lrrcl} \min & & (1+\epsilon')\sum_{j=1}^N x_j & & \\ \text{s.t.} & \forall i \in \{1\dots m\} & \sum_{j=1}^N T_{ji} x_j & \ge & b_i \\ & \forall j \in \{1,\dots,N\} & x_j & \ge & 0 \end{array}$$

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\mathrm{OPT} \le z \le (1+\epsilon')\mathrm{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL′.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL″.
- Let DUAL″ be DUAL without unused constraints.
- The dual to DUAL″ is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL″ is at most $(1+\epsilon')\mathrm{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\mathrm{OPT} \leq z \leq (1+\epsilon')\mathrm{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1+\epsilon')\mathrm{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\text{OPT} \le z \le (1 + \epsilon')\text{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1 + \epsilon')\text{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\mathrm{OPT} \le z \le (1+\epsilon')\mathrm{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1+\epsilon')\mathrm{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\mathrm{OPT} \le z \le (1+\epsilon')\mathrm{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1+\epsilon')\mathrm{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\text{OPT} \leq z \leq (1 + \epsilon')\text{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1 + \epsilon')\text{OPT}$.
- We can compute the corresponding solution in polytime.

Separation Oracle

If the value of the computed dual solution (which may be infeasible) is z then

$$\mathrm{OPT} \leq z \leq (1+\epsilon')\mathrm{OPT}$$

How do we get good primal solution (not just the value)?

- The constraints used when computing z certify that the solution is feasible for DUAL'.
- Suppose that we drop all unused constraints in DUAL. We will compute the same solution feasible for DUAL'.
- Let DUAL'' be DUAL without unused constraints.
- The dual to DUAL'' is PRIMAL where we ignore variables for which the corresponding dual constraint has not been used.
- The optimum value for PRIMAL'' is at most $(1+\epsilon')\mathrm{OPT}$.
- We can compute the corresponding solution in polytime.

This gives that overall we need at most

$$(1+\epsilon')\mathrm{OPT}_{\mathrm{LP}}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I)))$$

bins.

We can choose $\epsilon' = \frac{1}{\mathrm{OPT}}$ as $\mathrm{OPT} \leq \#\text{items}$ and since we have a fully polynomial time approximation scheme (FPTAS) for knapsack.

This gives that overall we need at most

$$(1+\epsilon')\mathrm{OPT}_{\mathrm{LP}}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I)))$$

bins.

We can choose $\epsilon' = \frac{1}{\mathrm{OPT}}$ as $\mathrm{OPT} \leq$ #items and since we have a fully polynomial time approximation scheme (FPTAS) for knapsack.