Approximation algorithms, Linear programming

László Papp

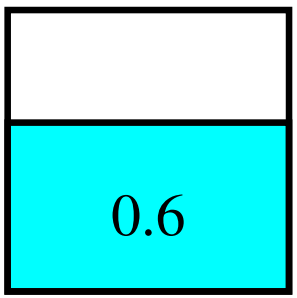
BME

2nd of May, 2023

First Fit Decreasing (FFD) Algorithm

First we sort the items according to their sizes to descending order, then we run the First Fit algorithm.

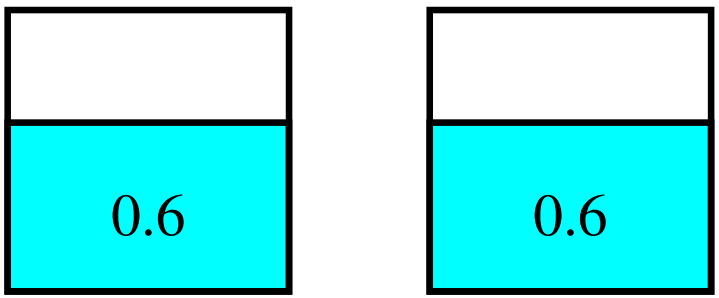
Example: If the input is 4 Bins of sizes 0.4, 0.4, 0.6, 0.6, then the decreasing order is 0.6, 0.6, 0.4, 0.4 and the First Fit algorithm finds the optimal solution:



First Fit Decreasing (FFD) Algorithm

First we sort the items according to their sizes to descending order, then we run the First Fit algorithm.

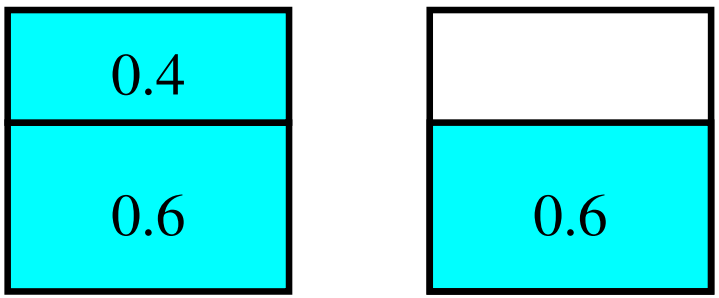
Example: If the input is 4 Bins of sizes 0.4, 0.4, 0.6, 0.6, then the decreasing order is 0.6, 0.6, 0.4, 0.4 and the First Fit algorithm finds the optimal solution:



First Fit Decreasing (FFD) Algorithm

First we sort the items according to their sizes to descending order, then we run the First Fit algorithm.

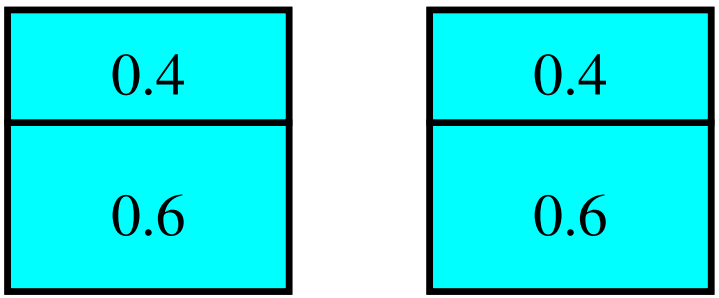
Example: If the input is 4 Bins of sizes 0.4, 0.4, 0.6, 0.6, then the decreasing order is 0.6, 0.6, 0.4, 0.4 and the First Fit algorithm finds the optimal solution:



First Fit Decreasing (FFD) Algorithm

First we sort the items according to their sizes to descending order, then we run the First Fit algorithm.

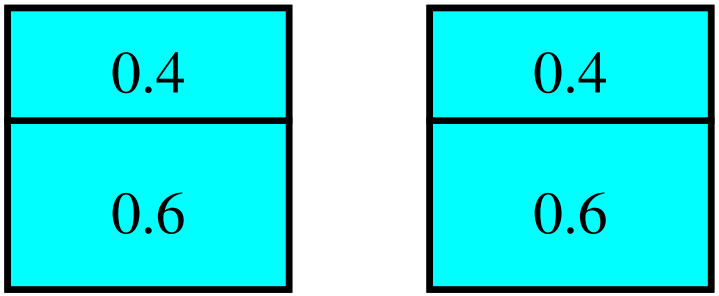
Example: If the input is 4 Bins of sizes 0.4, 0.4, 0.6, 0.6, then the decreasing order is 0.6, 0.6, 0.4, 0.4 and the First Fit algorithm finds the optimal solution:



First Fit Decreasing (FFD) Algorithm

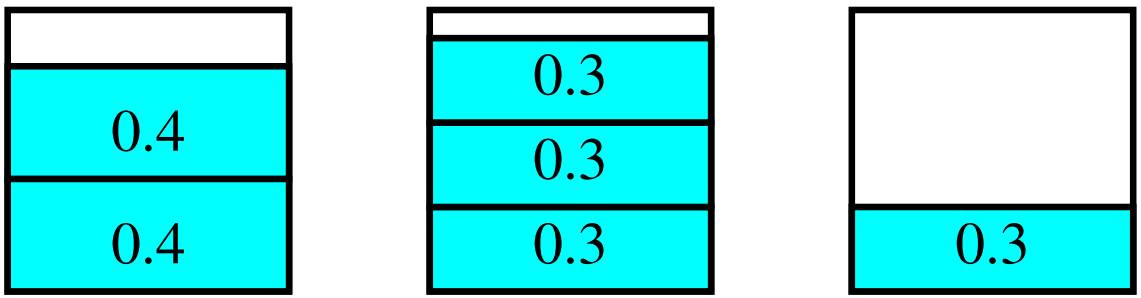
First we sort the items according to their sizes to descending order, then we run the First Fit algorithm.

Example: If the input is 4 Bins of sizes 0.4, 0.4, 0.6, 0.6, then the decreasing order is 0.6, 0.6, 0.4, 0.4 and the First Fit algorithm finds the optimal solution:



However FFD also does not find the optimum in all the cases:

Example: The input is 6 Bins of size 0.4, 0.4, 0.3, 0.3, 0.3, 0.3**. Then the FFD gives:**



But the optimal solution uses only 2 bins.

Why FF and FFD are good?

Claim

If $OPT(I)$ denotes the number of bins used in an optimal packing of an input I and $FF(I)$ denotes the number of bins used in the solution given by the FF algorithm, then:
$FF(I) \leq 2OPT(I)$

Why FF and FFD are good?

Claim

If $OPT(I)$ denotes the number of bins used in an optimal packing of an input I and $FF(I)$ denotes the number of bins used in the solution given by the FF algorithm, then:
$FF(I) \leq 2OPT(I)$

Proof: Denote the sum of the sizes with S, so $S = \sum_{i=1}^{n} a_i$. 2 bins which are at least half empty cannot appear during the run of the FF algorithm, because it would not create the later one.

In a packing which the FF produces all but one bin is more than half full. Thus:

$\frac{1}{2}(FF(I) - 1) < S \implies FF(I) < 2S + 1 \implies FF(I) \leq \lceil 2S \rceil$

The ceiling of the sum of the sizes is a lower bound on the number of required bins, therefore $\lceil S \rceil \leq OPT(I)$.

$$FF(I) \leq \lceil 2S \rceil \leq 2 \lceil S \rceil \leq 2OPT(I)$$

Approximation algorithms

We have an optimization problem, so we have an objective function f over the set of solutions **A** and we are looking for an optimal solution which is minimal (or maximal) among the possible solutions.
Denote an optimal solution by x_{opt}, and denote the optimum objective function value by OPT. So $OPT = f(x_{opt})$.
Instead we searching for an optimal solution which takes so much time, we search for a solution x which is quite good.

Definition: An algorithm for an optimization problem π is a **c-approximation algorithm** if for any input of π it finds a solution x such that $f(x) \leq c \cdot OPT$ if it is a minimization problem or $f(x) \geq \frac{1}{c} OPT$ if it is a maximization problem.

Polynomial time c-approximation algorithms

We are interested in efficient, so polynomial time c-approximation algorithms.
Note that we have proved that First Fit is a 2-approximation algorithm for Bin packing. Furthermore FF and FFD are polynomial time algorithms.

Polynomial time c-approximation algorithms

We are interested in efficient, so polynomial time c-approximation algorithms.
Note that we have proved that First Fit is a 2-approximation algorithm for Bin packing. Furthermore FF and FFD are polynomial time algorithms.
We do not give a proof for the following results:

Claim

- $FF(I) \leq \lceil \frac{17}{10} OPT(I) \rceil$ and there are infinitely many ipnuts which satisfies that $FF(I) \geq \frac{17}{10} OPT(I)$
- $FFD(I) \leq \frac{11}{9} OPT(I) + 4$ and there are infinitely many inputs which satisfies that $FFD(I) \geq \frac{11}{9} OPT(I)$. Where $FFD(I)$ denotes the number of bins used by the First Fit decreasing algorithm for input I.

Polynomial time c-approximation algorithms

We are interested in efficient, so polynomial time c-approximation algorithms.
Note that we have proved that First Fit is a 2-approximation algorithm for Bin packing. Furthermore FF and FFD are polynomial time algorithms.
We do not give a proof for the following results:

Claim

- $FF(I) \leq \lceil \frac{17}{10} OPT(I) \rceil$ and there are infinitely many ipnuts which satisfies that $FF(I) \geq \frac{17}{10} OPT(I)$
- $FFD(I) \leq \frac{11}{9} OPT(I) + 4$ and there are infinitely many inputs which satisfies that $FFD(I) \geq \frac{11}{9} OPT(I)$. Where $FFD(I)$ denotes the number of bins used by the First Fit decreasing algorithm for input I.

Question: Are there fast (polynomial time) approximation algorithms for all optimization problems which we have learnt?

Polynomial time c-approximation algorithms

We are interested in efficient, so polynomial time c-approximation algorithms.
Note that we have proved that First Fit is a 2-approximation algorithm for Bin packing. Furthermore FF and FFD are polynomial time algorithms.
We do not give a proof for the following results:

Claim

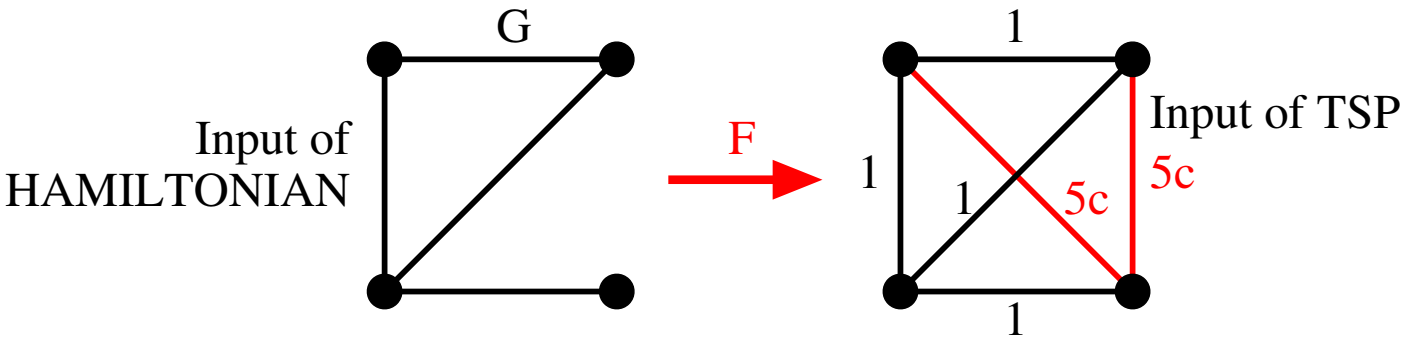
- $FF(I) \leq \lceil \frac{17}{10} OPT(I) \rceil$ and there are infinitely many ipnuts which satisfies that $FF(I) \geq \frac{17}{10} OPT(I)$
- $FFD(I) \leq \frac{11}{9} OPT(I) + 4$ and there are infinitely many inputs which satisfies that $FFD(I) \geq \frac{11}{9} OPT(I)$. Where $FFD(I)$ denotes the number of bins used by the First Fit decreasing algorithm for input I.

Question: Are there fast (polynomial time) approximation algorithms for all optimization problems which we have learnt?
If P$\neq$NP, then the answer is NO. :(

If a poly. c-approximation algorithm for the TSP exists, then P=NP.

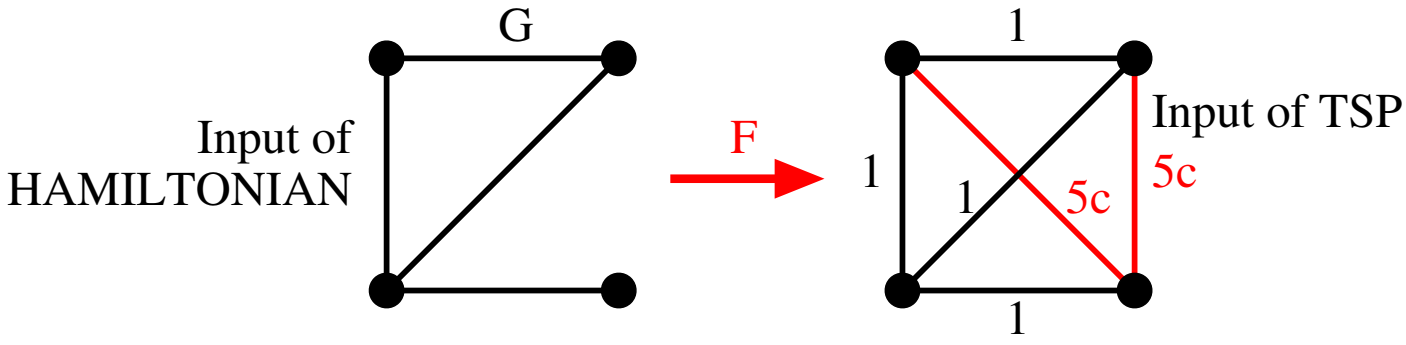
Proof: Assume that we have such an algorithm. Then we can decide HAMILTONIAN in polynomial time:

If G is an input graph, then we construct an input of the TSP where the length of an edge is 1 if it is contained in G and $c(|V(G)|+1)$ if it is not contained in G.



If a poly. c-approximation algorithm for the TSP exists, then P=NP.

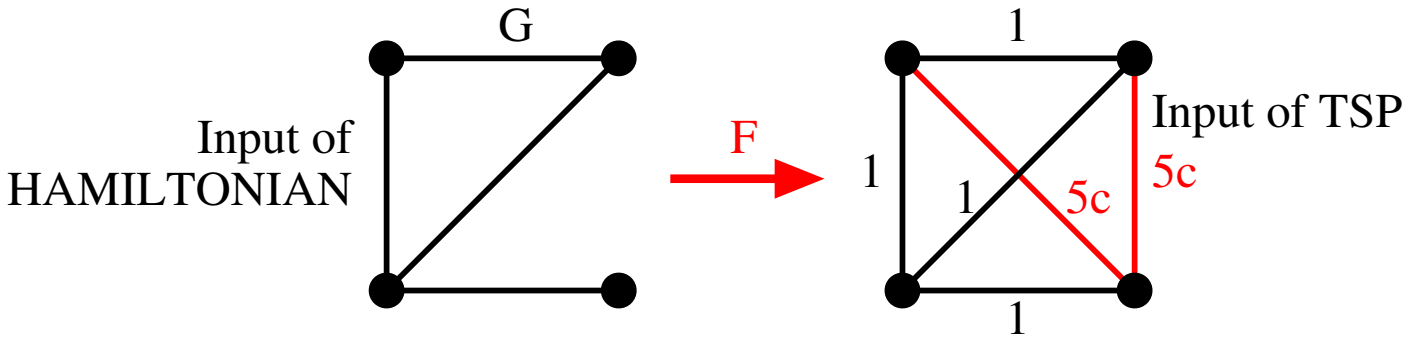
Proof: Assume that we have such an algorithm. Then we can decide HAMILTONIAN in polynomial time:
If G is an input graph, then we construct an input of the TSP where the length of an edge is 1 if it is contained in G and $c(|V(G)|+1)$ if it is not contained in G.



If G contains a Ham(iltonian) cycle, then in $F(G)$ there is a Ham cycle of length $|V(G)|$ and the c-approximation algorithm finds one whose length is at most $c|V(G)|$. If G does not contain a Ham cycle, then in $F(G)$ each Ham cycle is longer than $c|V(G)|$.

If a poly. c-approximation algorithm for the TSP exists, then P=NP.

Proof: Assume that we have such an algorithm. Then we can decide HAMILTONIAN in polynomial time:
If G is an input graph, then we construct an input of the TSP where the length of an edge is 1 if it is contained in G and $c(|V(G)|+1)$ if it is not contained in G.



If G contains a Ham(iltonian) cycle, then in $F(G)$ there is a Ham cycle of length $|V(G)|$ and the c-approximation algorithm finds one whose length is at most $c|V(G)|$. If G does not contain a Ham cycle, then in $F(G)$ each Ham cycle is longer than $c|V(G)|$. So G contains a Ham cycle if and only if the algorithm outputs a Hamiltonian cycle of length at most $c|V(G)|$. So this approximation algorithm decides HAMILTONIAN, which is NP-complete, in polynomial time. So P=NP. □

Task

We are making soft drinks from orange juice and sparkling water. We offer two kind of products:

- Sparkling Juice: contains 2 deciliters of juice and 4 deciliters of water for 1€.
- Super Orange: contains 3 deciliters of juice and 1 deciliter of water for 1€.

We can sell any amount of these (for example 1 liter of the first kind) but the mixing ratios remain the same.

Question: How much money can we earn if we have 18 deciliters of juice and 16 deciliters of sparkling water in stock?

Mathematical model

Lets denote the money which we obtain by selling the first and the second drink by x and y, respectively.
Then we can write the following set of linear inequalities:

$$\begin{aligned} 2x + 3y &\leq 18 \\ 4x + 1y &\leq 16 \\ x &\geq 0 \\ y &\geq 0 \end{aligned}$$

The first inequality corresponds to the juice usage, the second one corresponds to the water usage. It is obvious that our earnings is nonnegative. We want to maximize the total amount of money which is $x + y$.

Solving the problem - Feasible region

The inequalities can be written in the following equivalent form:

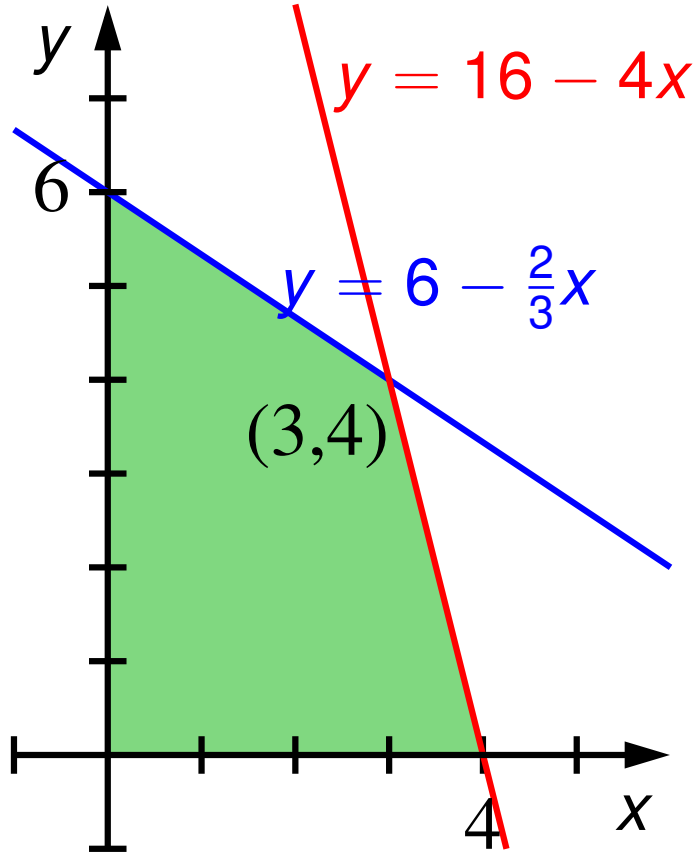
$$y \leq 6 - \frac{2}{3}x$$

$$y \leq 16 - 4x$$

$$x \geq 0$$

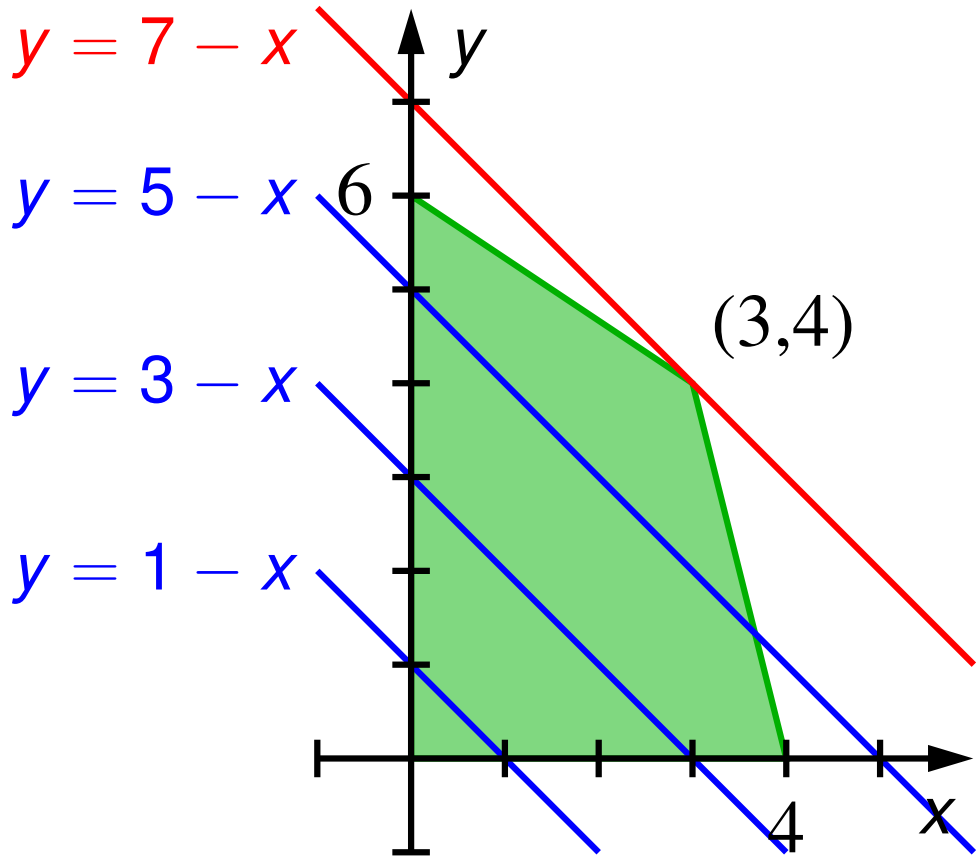
$$y \geq 0$$

The set of possible solutions (feasible region) is the green simple polygon.



Solving the problem - Objective function as a geometric object

The objective function $s = x + y$ is a line and can be written as $y = s - x$.
Therefore we are searching for a line $y = s - x$ where s is maximal and it intersects the feasible region.



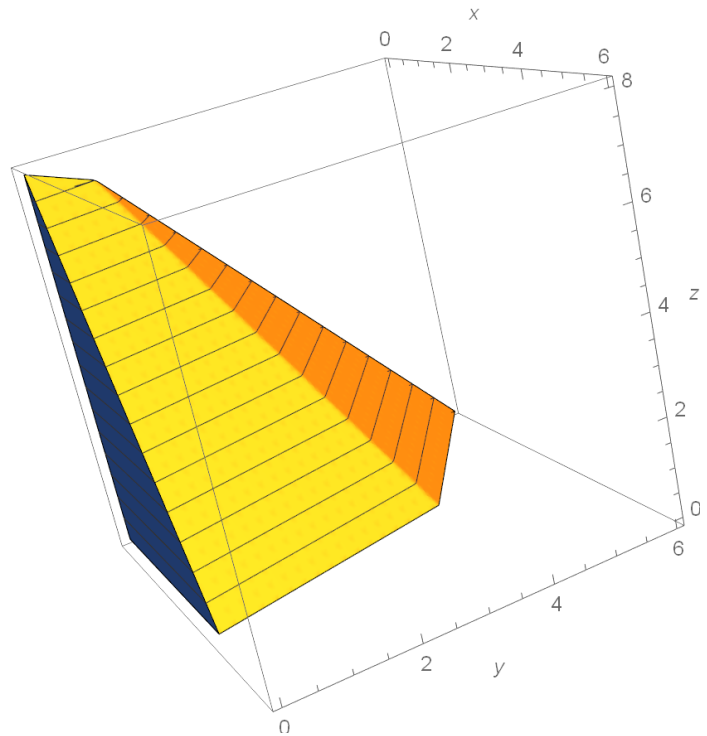
So the maximum of the objective function is 7 and it can be attained at the point $x = 3$, $y = 4$.

Note: This solution method is called as the graphical method.

What if we have a 3rd type of drink?

Assume that we also sell a third drink for 1€ and it contains 2 deciliters of juice and 2 deciliters of sparkling water.
Then we need a third variable z, and we obtain the following set of inequalities:

$$\begin{aligned} 2x + 3y + 2z &\leq 18 \\ 4x + y + 2z &\leq 16 \\ x &\geq 0 \\ y &\geq 0 \\ z &\geq 0 \end{aligned}$$



The feasible region is a polyhedron.

The objective function $s = x + y + z$ is a plane. Therefore we are looking for a plane, which intersects this polyhedron and where s is maximal.

What if there are even more type of drinks?

If we have n drinks then we end up with n variables. The feasible region is an n dimensional polytope and the objective function corresponds to an $n-1$ dimensional hyperplane (a subspace whose dimension is n-1). In this case we cannot utilize geometry.

So if we have more than two variables, then we cannot use the graphical method to solve the problem.

Question: What to do?

Answer: Use linear algebra!

Notations and conventions used in the slides

- We use upper case for matrices and lower case for vectors.
- In the beginning I underline each vector, but later I am going to omit this notation.
- Each vector is a column vector, therefore if I want to write a row vector $\underline{v}$, then I write $\underline{v}^T$.
- If $\underline{v}$ is a vector, then $v(i)$ or v_i denotes its ith element, but I do not use both simultaneously. If I write $v(i)$ it means that I use v_i for something else! Sorry.
- $\underline{v}^T \underline{u}$ is the dot product of $\underline{v}$ and $\underline{u}$.

Linear program

We can write the inequalities of the soft drink problem in the following equivalent form:

$$
\begin{aligned}
2x + 3y &\leq 18 \\
4x + 1y &\leq 16 \\
x &\geq 0 \\
y &\geq 0 \\
\max\ x + y &
\end{aligned}
$$

$$
\begin{bmatrix} 2 & 3 \\ 4 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 18 \\ 16 \\ 0 \\ 0 \end{bmatrix}
$$

$$
\max \begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}
$$

Notation: Let $\underline{a}$ and $\underline{b}$ be vectors having n rows. $\underline{a} \leq \underline{b}$ means that $a(i) \leq b(i)$ for each $1 \leq i \leq n$.

So we have written the problem in the following form:
Maximize $\underline{c}^T \underline{x}$ subject to $A\underline{x} \leq \underline{b}$.

Linear programming

The basic problem of linear programming is the following task: An $m \times n$ martix A, a vector $\underline{b} \in \mathbb{R}^m$ and a vector $\underline{c} \in \mathbb{R}^n$ are given. Find a vector $\underline{x} \in \mathbb{R}^n$ which satisfies $A\underline{x} \leq \underline{b}$ and $\underline{c}^T \underline{x}$ is the biggest possible. $\underline{c}^T \underline{x}$ is called as the **objective function**.

This is usually written for short in the following form: Maximize $\underline{c}^T \underline{x}$, respect to $A\underline{x} \leq \underline{b}$.

Some other forms of linear programming problems, which we call as linear programs for short:

- Minimize $\underline{c}^T \underline{x}$, respect to $A\underline{x} \leq \underline{b}$.
- Maximize/Minimize $\underline{c}^T \underline{x}$, respect to $A\underline{x} \leq \underline{b}$ and $\underline{x} \geq 0$.
- Maximize/Minimize $\underline{c}^T \underline{x}$, respect to $A\underline{x} = \underline{b}$ and $\underline{x} \geq 0$.

We can solve all of these if we can solve the task that Maximize $\underline{c}^T \underline{x}$, respect to $A\underline{x} \leq \underline{b}$.

Is there a solution?

The first natural question when we obtain such a problem is the following: Do we have a feasible solution? So is there an $\underline{x}$ which satisfies $A\underline{x} \leq \underline{b}$?

There are algorithms which can determine this. For example Fourier-Moltzkin elimination, which is an exponential time algorithm. Due to the shortage of time we are not going to talk about that. Instead of that we mention two theorems which connects the solvability of two different systems. These theorems will be useful later.

Farkas' Lemma

Farkas' Lemma (1st version)

Let A be an $m \times n$ matrix and let $\underline{b} \in \mathbb{R}^m$. Exactly one of the following two systems has a solution:

1. $A\underline{x} \leq \underline{b}$
2. $\underline{y}^T A = \underline{0},\ \underline{y} \geq \underline{0},\ \underline{y}^T \underline{b} < 0$

Note: $(\underline{y}^T A)^T = A^T \underline{y}$

Farkas' Lemma

Farkas' Lemma (1st version)

Let A be an $m \times n$ matrix and let $\underline{b} \in \mathbb{R}^m$. Exactly one of the following two systems has a solution:

1. $A\underline{x} \leq \underline{b}$
2. $\underline{y}^T A = \underline{0},\ \underline{y} \geq \underline{0},\ \underline{y}^T \underline{b} < 0$

Note: $(\underline{y}^T A)^T = A^T \underline{y}$

Sketch of the proof: These systems cannot have solutions at the same time:

Assume the contrary, so let $\underline{x}$ and $\underline{y}$ be solutions of the first and the second system, respectively. Then

$0 = \underline{0}^T \underline{x} = \underline{y}^T A \underline{x} \leq \underline{y}^T \underline{b} < 0$ which is a contradiction.

To prove that one of the two systems is solvable it is enough to show that if the first one does not have a solution, then the second one does. We skip the proof of this.

Equivalent form of Farkas' Lemma

Farkas' Lemma (2nd version)

Let A be an $m \times n$ matrix and let $\underline{b} \in \mathbb{R}^m$. Exactly one of the following two systems has a solution:

1. $A\underline{x} = \underline{b},\ \underline{x} \geq \underline{0}$
2. $\underline{y}^T A \geq \underline{0},\ \underline{y}^T \underline{b} < 0$

Sketch of the proof: These systems cannot have solutions at the same time:
Assume the contrary, so let x and y be solutions of the first and the second system, respectively. Then $0 > y^T b = y^T Ax \geq 0$, because the dot product of two non-negative vectors (does not have negative elements) is non-negative. This is a contradiction.
To prove that one of the two systems is solvable it is enough to show that if the second one does not have a solution, then the first one does.

Equivalent form of Farkas' Lemma

Farkas' Lemma (2nd version)

Let A be an $m \times n$ matrix and let $\underline{b} \in \mathbb{R}^m$. Exactly one of the following two systems has a solution:

1. $A\underline{x} = \underline{b},\ \underline{x} \geq \underline{0}$
2. $\underline{y}^T A \geq \underline{0},\ \underline{y}^T \underline{b} < 0$

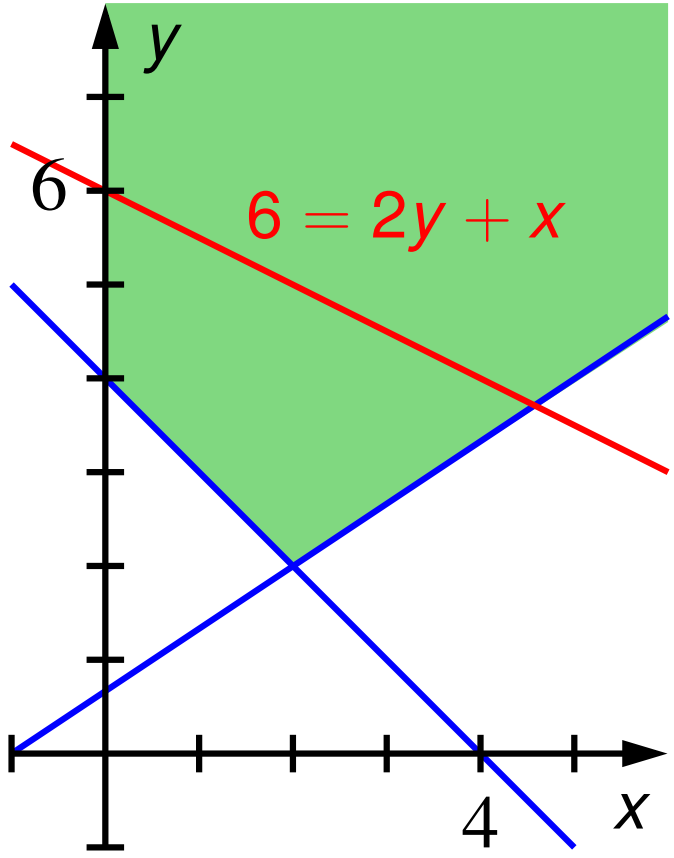
Sketch of the proof: These systems cannot have solutions at the same time:
Assume the contrary, so let x and y be solutions of the first and the second system, respectively. Then $0 > y^T b = y^T Ax \geq 0$, because the dot product of two non-negative vectors (does not have negative elements) is non-negative. This is a contradiction.
To prove that one of the two systems is solvable it is enough to show that if the second one does not have a solution, then the first one does. This can be done by using the 1st version of Farkas' Lemma, but we skip it.

Does the objective function have an upper bound on the solution set?

If we are looking for a solution $\underline{x}$ which maximizes the objective function, then it is possible that no such $\underline{x}$ exists, because $\underline{c}^T \underline{x}$ can be arbitrary large on the set of feasible solutions.

Example:

$$x + y \geq 4$$
$$-2x + 3y \geq 1$$
$$x \geq 0$$
$$y \geq 0$$
$$\max\ x + 2y$$



Solutions where $x + 2y = s$:
$y = \frac{s-x}{2}$

These statements are equivalent

1. $c^T x$ is bounded above on the set of solutions of $Ax \leq b$.
2. The system $Az \leq \underline{0}$, $c^T z > 0$ does not have a solution.
3. The system $y^T A = c^T$, $y \geq 0$ has a solution.

Proof: 1.$\rightarrow$ 2.: Indirectly assume that there is a solution z_0 of $Az \leq \underline{0}$, $cz > 0$. Consider a solution x_0 of $Ax \leq b$.

These statements are equivalent

1. $c^T x$ is bounded above on the set of solutions of $Ax \leq b$.
2. The system $Az \leq \underline{0}$, $c^T z > 0$ does not have a solution.
3. The system $y^T A = c^T$, $y \geq 0$ has a solution.

Proof: 1.$\rightarrow$ 2.: Indirectly assume that there is a solution z_0 of $Az \leq \underline{0}$, $cz > 0$. Consider a solution x_0 of $Ax \leq b$.
Then $x_0 + \lambda z_0$ is also a solution of $Ax \leq b$ if $\lambda > 0$:
$A(x_0 + \lambda z_0) = Ax_0 + \lambda Az_0 \leq b + \lambda \underline{0} = b$.

These statements are equivalent

1. $c^T x$ is bounded above on the set of solutions of $Ax \leq b$.
2. The system $Az \leq \underline{0}$, $c^T z > 0$ does not have a solution.
3. The system $y^T A = c^T$, $y \geq 0$ has a solution.

Proof: 1.$\rightarrow$ 2.: Indirectly assume that there is a solution z_0 of $Az \leq \underline{0}$, $cz > 0$. Consider a solution x_0 of $Ax \leq b$.
Then $x_0 + \lambda z_0$ is also a solution of $Ax \leq b$ if $\lambda > 0$:
$A(x_0 + \lambda z_0) = Ax_0 + \lambda Az_0 \leq b + \lambda \underline{0} = b$.
$c^T(x_0 + \lambda z_0) = c^T x_0 + \lambda c^T z_0$ so it is a linear function in λ which can be arbitrary large if we choose λ big enough.

These statements are equivalent

1. $c^T x$ is bounded above on the set of solutions of $Ax \leq b$.
2. The system $Az \leq \underline{0}$, $c^T z > 0$ does not have a solution.
3. The system $y^T A = c^T$, $y \geq 0$ has a solution.

Proof: 1.$\rightarrow$ 2.: Indirectly assume that there is a solution z_0 of $Az \leq \underline{0}$, $cz > 0$. Consider a solution x_0 of $Ax \leq b$.
Then $x_0 + \lambda z_0$ is also a solution of $Ax \leq b$ if $\lambda > 0$:
$A(x_0 + \lambda z_0) = Ax_0 + \lambda Az_0 \leq b + \lambda \underline{0} = b$.
$c^T(x_0 + \lambda z_0) = c^T x_0 + \lambda c^T z_0$ so it is a linear function in λ which can be arbitrary large if we choose λ big enough.
2.$\rightarrow$ 3.: If $Az \leq \underline{0}, c^T z > 0$ does not have a solution, then the system $Az \geq \underline{0}, c^T z < 0$ also does not have. Then by the 2nd version of Farkas' Lemma $y^T A = c^T, y \geq \underline{0}$ has a solution.

These statements are equivalent

1. $c^T x$ is bounded above on the set of solutions of $Ax \leq b$.
2. The system $Az \leq \underline{0}$, $c^T z > 0$ does not have a solution.
3. The system $y^T A = c^T$, $y \geq 0$ has a solution.

Proof: 1.→ 2.: Indirectly assume that there is a solution z_0 of $Az \leq \underline{0}$, $cz > 0$. Consider a solution x_0 of $Ax \leq b$.
Then $x_0 + \lambda z_0$ is also a solution of $Ax \leq b$ if $\lambda > 0$:
$A(x_0 + \lambda z_0) = Ax_0 + \lambda Az_0 \leq b + \lambda \underline{0} = b$.
$c^T(x_0 + \lambda z_0) = c^T x_0 + \lambda c^T z_0$ so it is a linear function in λ which can be arbitrary large if we choose λ big enough.
2.→ 3.: If $Az \leq \underline{0}, c^T z > 0$ does not have a solution, then the system $Az \geq \underline{0}, c^T z < 0$ also does not have. Then by the 2nd version of Farkas' Lemma $y^T A = c^T, y \geq \underline{0}$ has a solution.
3.→ 1.: Let x_0 be any solution of $Ax \leq b$ and let y_0 be a solution of $y^T A = c^T, y \geq 0$. Then:
$c^T x_0 = y_0^T A x_0 \leq y_0^T b = s$, where s is a number and it is an upper bound on the objective function $c^T x$, because the value of s does not depend on the choice of x_0. □