Primal-Dual Algorithm

Math 482, Lecture 29

Misha Lavrov

April 17, 2020

The problem

Our goal: to solve the primal-dual pair of linear programs below.

$$(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases} \qquad (\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \\ & \mathbf{u} \text{ unrestricted} \end{cases}$$

The problem

Our goal: to solve the primal-dual pair of linear programs below.

$$
(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}
\qquad
(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \\ & \mathbf{u} \text{ unrestricted} \end{cases}
$$

We will try to improve on the simplex algorithm by taking *long jumps* across the feasible region for (**D**).

The problem

Our goal: to solve the primal-dual pair of linear programs below.

$$
(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}
\qquad
(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \\ & \mathbf{u} \text{ unrestricted} \end{cases}
$$

We will try to improve on the simplex algorithm by taking *long jumps* across the feasible region for (**D**).

Motivation: the Ford–Fulkerson method, where a single augmenting steps changes many variables at once.

The direction-finding problem

Consider the following example:

$$
(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^2}{\text{maximize}} & 3u_1 + 3u_2 \\ \text{subject to} & 2u_1 + 4u_2 \le 2 \\ & u_1 - u_2 \le 2 \\ & -4u_1 + u_2 \le 1 \end{cases}
$$

We are at the point $\mathbf{u} = (1, 0)$ and want to pick a direction $\mathbf{v}$ to go.

What is the best direction, and how do we find it?

The direction-finding problem

Consider the following example:

$$(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^2}{\text{maximize}} & 3u_1 + 3u_2 \\ \text{subject to} & 2u_1 + 4u_2 \le 2 \\ & u_1 - u_2 \le 2 \\ & -4u_1 + u_2 \le 1 \end{cases}$$

We are at the point $\mathbf{u} = (1, 0)$ and want to pick a direction $\mathbf{v}$ to go.

What is the best direction, and how do we find it?

- **Answer 1.** (From calculus.) Gradient descent: take $\mathbf{v}$ proportional to $\mathbf{b} = (3, 3)$, the cost vector.

The direction-finding problem

Consider the following example:

$$
(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^2}{\text{maximize}} & 3u_1 + 3u_2 \\ \text{subject to} & 2u_1 + 4u_2 \le 2 \\ & u_1 - u_2 \le 2 \\ & -4u_1 + u_2 \le 1 \end{cases}
$$

We are at the point $\mathbf{u} = (1, 0)$ and want to pick a direction $\mathbf{v}$ to go.

What is the best direction, and how do we find it?

- **Answer 1.** (From calculus.) Gradient descent: take $\mathbf{v}$ proportional to $\mathbf{b} = (3, 3)$, the cost vector.
- **Answer 2**. We should also make sure we don't accidentally leave the feasible region.

The linear program for **v**

We can find **v** using a linear program:

The linear program for **v**

We can find **v** using a linear program:

- Objective function: we want to improve $3u_1 + 3u_2$ as quickly as possible, so we maximize $3v_1 + 3v_2$.

The linear program for $\mathbf{v}$

We can find $\mathbf{v}$ using a linear program:

- Objective function: we want to improve $3u_1 + 3u_2$ as quickly as possible, so we maximize $3v_1 + 3v_2$.
- Boundary conditions: At $\mathbf{u} = (1, 0)$, the constraint $2u_1 + 4u_2 \leq 2$ is tight, so we make sure we don't violate it and ask that
$$2v_1 + 4v_2 \leq 0.$$

The linear program for $\mathbf{v}$

We can find $\mathbf{v}$ using a linear program:

- Objective function: we want to improve $3u_1 + 3u_2$ as quickly as possible, so we maximize $3v_1 + 3v_2$.
- Boundary conditions: At $\mathbf{u} = (1, 0)$, the constraint $2u_1 + 4u_2 \leq 2$ is tight, so we make sure we don't violate it and ask that

  $$2v_1 + 4v_2 \leq 0.$$

  The constraints $u_1 - u_2 \leq 2$ and $-4u_1 + u_2 \leq 1$ are slack, so we ignore them.

The linear program for $\mathbf{v}$

We can find $\mathbf{v}$ using a linear program:

- Objective function: we want to improve $3u_1 + 3u_2$ as quickly as possible, so we maximize $3v_1 + 3v_2$.
- Boundary conditions: At $\mathbf{u} = (1, 0)$, the constraint $2u_1 + 4u_2 \leq 2$ is tight, so we make sure we don't violate it and ask that
$$2v_1 + 4v_2 \leq 0.$$
The constraints $u_1 - u_2 \leq 2$ and $-4u_1 + u_2 \leq 1$ are slack, so we ignore them.
- Scaling constraints: we just want a direction, not a magnitude. So we limit $\mathbf{v}$ by asking that $v_1, v_2 \leq 1$.

The linear program for $\mathbf{v}$

We can find $\mathbf{v}$ using a linear program:

- Objective function: we want to improve $3u_1 + 3u_2$ as quickly as possible, so we maximize $3v_1 + 3v_2$.
- Boundary conditions: At $\mathbf{u} = (1, 0)$, the constraint $2u_1 + 4u_2 \leq 2$ is tight, so we make sure we don't violate it and ask that
$$2v_1 + 4v_2 \leq 0.$$
The constraints $u_1 - u_2 \leq 2$ and $-4u_1 + u_2 \leq 1$ are slack, so we ignore them.
- Scaling constraints: we just want a direction, not a magnitude. So we limit $\mathbf{v}$ by asking that $v_1, v_2 \leq 1$.

(Fine print: this only works if the coefficients in the objective function $\mathbf{u}^\mathsf{T}\mathbf{b}$ are nonnegative, but we can make sure that this holds.)

The direction-finding linear program

The original LP, (**D**), leads to the auxiliary LP, (**DRP**):

$$(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \le \mathbf{c}^\mathsf{T} \\ & \mathbf{u} \text{ unrestricted} \end{cases} \qquad (\mathbf{DRP}) \begin{cases} \underset{\mathbf{v}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{v}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{v}^\mathsf{T}A_J \le \mathbf{0}^\mathsf{T} \\ & v_1, \dots, v_m \le 1 \end{cases}$$

(Here, J indexes constraints which are tight at the initial point $\mathbf{u}$.)

The direction-finding linear program

The original LP, (**D**), leads to the auxiliary LP, (**DRP**):

$$(\mathbf{D}) \begin{cases} \underset{\mathbf{u} \in \mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T} A \le \mathbf{c}^\mathsf{T} \\ & \mathbf{u} \text{ unrestricted} \end{cases} \qquad (\mathbf{DRP}) \begin{cases} \underset{\mathbf{v} \in \mathbb{R}^m}{\text{maximize}} & \mathbf{v}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{v}^\mathsf{T} A_J \le \mathbf{0}^\mathsf{T} \\ & v_1, \dots, v_m \le 1 \end{cases}$$

(Here, J indexes constraints which are tight at the initial point $\mathbf{u}$.)

In our example, we get:

$$(\mathbf{DRP}) \begin{cases} \underset{\mathbf{v} \in \mathbb{R}^2}{\text{maximize}} & 3v_1 + 3v_2 \\ \text{subject to} & 2v_1 + 4v_2 \le 0 \\ & v_1 \le 1 \\ & v_2 \le 1 \end{cases}$$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction $\mathbf{v}$. We get $(v_1, v_2) = (1, -\frac{1}{2})$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction $\mathbf{v}$. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction $\mathbf{v}$. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

$$\begin{cases} 2(u_1 + tv_1) + 4(u_2 + tv_2) \le 2 \text{ holds automatically} \\ \\ \end{cases}$$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction $\mathbf{v}$. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

$$\begin{cases} 2(u_1 + tv_1) + 4(u_2 + tv_2) \leq 2 \text{ holds automatically} \\ (u_1 + tv_1) - (u_2 + tv_2) \leq 2 \implies t \leq \frac{2}{3} \\ \end{cases}$$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction **v**. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

$$\begin{cases} 2(u_1 + tv_1) + 4(u_2 + tv_2) \le 2 \text{ holds automatically} \\ (u_1 + tv_1) - (u_2 + tv_2) \le 2 \implies t \le \frac{2}{3} \\ -4(u_1 + tv_1) + (u_2 + tv_2) \le 1 \implies t \ge -\frac{9}{10} \end{cases}$$

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction **v**. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

$$\begin{cases} 2(u_1 + tv_1) + 4(u_2 + tv_2) \leq 2 \text{ holds automatically} \\ (u_1 + tv_1) - (u_2 + tv_2) \leq 2 \implies t \leq \frac{2}{3} \\ -4(u_1 + tv_1) + (u_2 + tv_2) \leq 1 \implies t \geq -\frac{9}{10} \end{cases}$$

Largest value of t allowed is $t = \frac{2}{3}$.

The augmenting step

We can now do the primal-dual algorithm, just very badly.

1. Start at some solution to (**D**). $(u_1, u_2) = (1, 0)$
2. Write the direction-finding problem (**DRP**). (previous slide)
3. Solve (**DRP**) to find a direction $\mathbf{v}$. We get $(v_1, v_2) = (1, -\frac{1}{2})$
4. Find the largest t such that $\mathbf{u} + t\mathbf{v}$ is still feasible for (**D**).

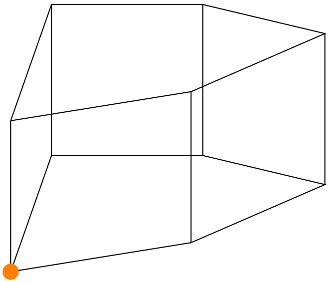
$$\begin{cases} 2(u_1 + tv_1) + 4(u_2 + tv_2) \leq 2 \text{ holds automatically} \\ (u_1 + tv_1) - (u_2 + tv_2) \leq 2 \implies t \leq \frac{2}{3} \\ -4(u_1 + tv_1) + (u_2 + tv_2) \leq 1 \implies t \geq -\frac{9}{10} \end{cases}$$

   Largest value of t allowed is $t = \frac{2}{3}$.

5. Replace $\mathbf{u}$ by $\mathbf{u} + t\mathbf{v}$ and go back to step 2. $\mathbf{u} + \frac{2}{3}\mathbf{v} = (\frac{5}{3}, -\frac{1}{3})$.

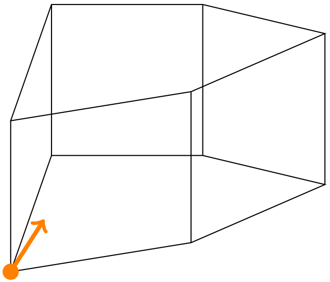
A made-up example

What this ideally looks like:



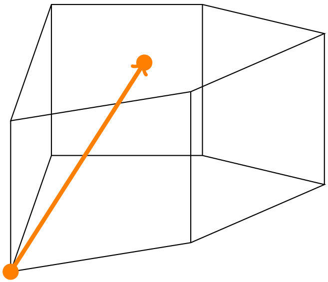
A made-up example

What this ideally looks like:



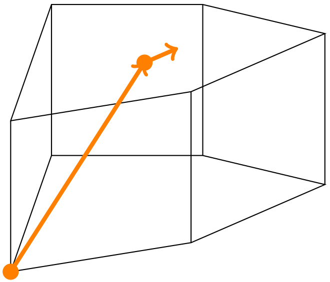
A made-up example

What this ideally looks like:



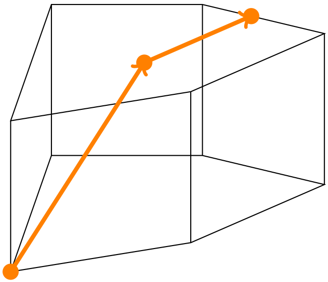
A made-up example

What this ideally looks like:



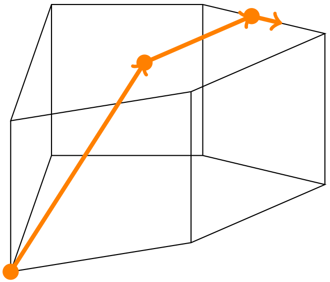
A made-up example

What this ideally looks like:



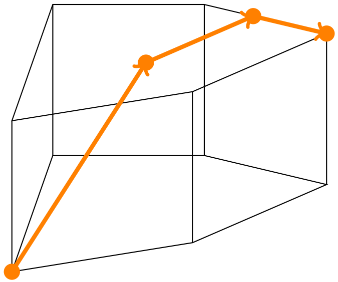
A made-up example

What this ideally looks like:



A made-up example

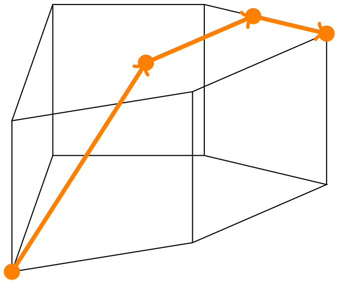
What this ideally looks like:



We stop when we reach a point where $t = 0$ and we cannot make any further improvement.

A made-up example

What this ideally looks like:

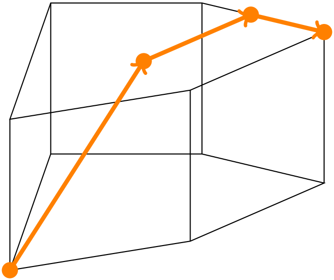


We stop when we reach a point where $t = 0$ and we cannot make any further improvement.

In theory, the advantage is that we do many fewer iterations.

A made-up example

What this ideally looks like:



We stop when we reach a point where $t = 0$ and we cannot make any further improvement.

In theory, the advantage is that we do many fewer iterations.

Right now, the disadvantage is that each iteration requires solving its own LP. This is way too slow!

The restricted primal

We fix this by coming up with a *fourth* linear program.

The restricted primal

We fix this by coming up with a *fourth* linear program.

$$(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}$$

The restricted primal

We fix this by coming up with a *fourth* linear program.

$$(\mathbf{P})\ \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}$$

$$(\mathbf{D})\ \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \end{cases}$$

The restricted primal

We fix this by coming up with a *fourth* linear program.

$$(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}$$

$$(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \end{cases}$$

$$(\mathbf{DRP}) \begin{cases} \underset{\mathbf{v}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{v}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{v}^\mathsf{T}A_J \leq \mathbf{0}^\mathsf{T} \\ & v_1, \dots, v_m \leq 1 \end{cases}$$

The restricted primal

We fix this by coming up with a *fourth* linear program.

$$(\mathbf{P}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{cases}$$

$$(\mathbf{RP}) \begin{cases} \underset{\mathbf{x}\in\mathbb{R}^n,\mathbf{y}\in\mathbb{R}^m}{\text{minimize}} & y_1 + \cdots + y_m \\ \text{subject to} & A_J\mathbf{x}_J + I\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{cases}$$

$$(\mathbf{D}) \begin{cases} \underset{\mathbf{u}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{u}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{u}^\mathsf{T}A \leq \mathbf{c}^\mathsf{T} \end{cases}$$

$$(\mathbf{DRP}) \begin{cases} \underset{\mathbf{v}\in\mathbb{R}^m}{\text{maximize}} & \mathbf{v}^\mathsf{T}\mathbf{b} \\ \text{subject to} & \mathbf{v}^\mathsf{T}A_J \leq \mathbf{0}^\mathsf{T} \\ & v_1, \ldots, v_m \leq 1 \end{cases}$$

Constructing the restricted primal

The restricted primal (**RP**) is the dual of (**DRP**), which is what we called the direction-finding linear program.

Constructing the restricted primal

The restricted primal (**RP**) is the dual of (**DRP**), which is what we called the direction-finding linear program.

It can also be obtained directly from (**P**):

- First, delete all primal variables x_j for which the j^{th} dual constraint is slack.

Constructing the restricted primal

The restricted primal (**RP**) is the dual of (**DRP**), which is what we called the direction-finding linear program.

It can also be obtained directly from (**P**):

- First, delete all primal variables x_j for which the j^{th} dual constraint is slack.
- Then, add a new variable y_i to the i^{th} constraint, for every i.

Constructing the restricted primal

The restricted primal (**RP**) is the dual of (**DRP**), which is what we called the direction-finding linear program.

It can also be obtained directly from (**P**):

- First, delete all primal variables x_j for which the j^{th} dual constraint is slack.
- Then, add a new variable y_i to the i^{th} constraint, for every i.
- Replace the objective function, instead minimizing $y_1 + \cdots + y_m$.

Constructing the restricted primal

The restricted primal (**RP**) is the dual of (**DRP**), which is what we called the direction-finding linear program.

It can also be obtained directly from (**P**):

- First, delete all primal variables x_j for which the j^{th} dual constraint is slack.
- Then, add a new variable y_i to the i^{th} constraint, for every i.
- Replace the objective function, instead minimizing $y_1 + \cdots + y_m$.

Independent motivation: (**RP**) has an objective value of 0 if and only if $A_J \mathbf{x}_J = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ has a solution, which is the complementary slackness condition to see if $\mathbf{u}$ is optimal.

Goal of the restricted primal

We will improve the primal-dual algorithm by solving (**RP**) instead of (**DRP**).

(We'll still be able to find the direction $\mathbf{v}$ from the optimal tableau for (**RP**).)

Goal of the restricted primal

We will improve the primal-dual algorithm by solving (**RP**) instead of (**DRP**).

(We'll still be able to find the direction $\mathbf{v}$ from the optimal tableau for (**RP**).)

Here's why this will help:

- Solving (**DRP**) requires starting from scratch every time: whatever the optimal direction $\mathbf{v}$ was at the previous iteration, it's definitely not valid any more.

Goal of the restricted primal

We will improve the primal-dual algorithm by solving (**RP**) instead of (**DRP**).

(We'll still be able to find the direction **v** from the optimal tableau for (**RP**).)

Here's why this will help:

- Solving (**DRP**) requires starting from scratch every time: whatever the optimal direction **v** was at the previous iteration, it's definitely not valid any more.
- However, (**RP**) keeps its constraints the same, possibly adding or removing variables, and it turns out that the optimal solution to (**RP**) will be a valid starting point for the next iteration of (**RP**).