Primal-Dual Algorithm Examples

We just saw the general primal-dual algorithm schema.

We will now see how to apply it to the

Shortest Path Problem

and the

Max Flow Problem

The Shortest Path Problem

Given $G = (V, E)$, let A be its $(|V| - 1) \times |E|$ node-arc incidence matrix (with the row for t erased since it is redundant). Primal for *shortest path problem* is

$$\min c'f$$

$$\begin{array}{ll} Af = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \begin{array}{l} \longleftarrow \text{Row } s \\ \\ \\ \\ \end{array} \end{array}$$

$$f \geq 0$$

Dual is

$$\begin{array}{rcll} & \max \pi_s & & \\ \pi_i - \pi_j & \leq & c_{ij} & \forall (i,j) \in E \\ \pi_i & \gtrless & 0 & \forall i \\ \pi_t & = & 0 & \end{array}$$

Note that $\pi_t = 0$ must be added in for consistency; in this case π_t is a **constant** and not a variable.

Dual

$$
\begin{array}{rcll}
\multicolumn{4}{l}{\max \pi_s} \\
\pi_i - \pi_j & \leq & c_{ij} & (i,j) \in E \\
\pi_i & \gtrless & 0 & \\
\pi_t & = & 0 &
\end{array}
$$

$$J = \{(i,j) \in E : \pi_i - \pi_j = c_{ij}\}$$

$$D \Rightarrow RP$$

Restricted primal (RP)

$$\min \xi = \sum_{i=1}^{m-1} x_i^a$$

$$Af + x^a = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$
\begin{array}{rcll}
f_j & \geq & 0 & \text{for all } j \\
f_j & = & 0 & j \notin J \\
x_i^a & \geq & 0 &
\end{array}
$$

Restricted primal (RP)

$$\begin{aligned}
\min \xi &= \textstyle\sum_{i=1}^{m-1} x_i^a \\
Af + x^a &= \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
f_j &\geq 0 \quad \text{for all } j \\
f_j &= 0 \quad j \notin J \\
x_i^a &\geq 0
\end{aligned}$$

$$RP \Rightarrow DRP$$

DRP

$$\begin{aligned}
\max w &= \pi_s \\
\pi_i - \pi_j &\leq 0 \qquad \forall (i,j) \in J \\
\pi_i &\leq 1 \\
\pi_i &\gtrless 0 \\
\pi_t &= 0
\end{aligned}$$

Have seen $P \Rightarrow D \Rightarrow RP \Rightarrow DRP$

Note differences between D and DRP

Dual

$$\begin{array}{rcll} \multicolumn{4}{l}{\max \pi_s} \\ \pi_i - \pi_j & \leq & c_{ij} & (i,j) \in E \\ \pi_i & \gtrless & 0 & \\ \pi_t & = & 0 & \end{array}$$

$$J = \{(i,j) \in E : \pi_i - \pi_j = c_{ij}\}$$

DRP

$$\max w = \pi_s$$

$$\pi_i - \pi_j \leq 0 \qquad (i,j) \in J$$

$$\pi_i \leq 1$$

$$\pi_i \gtrless 0$$

$$\pi_t = 0$$

DRP

$$
\begin{aligned}
\max w = \pi_s & \\
\pi_i - \pi_j \leq 0 & \qquad (i,j) \in J \\
\pi_i \leq 1 & \\
\pi_i \gtrless 0 & \\
\pi_t = 0 &
\end{aligned}
$$

The generic Primal-Dual algorithm solves RP using simplex and then uses the optimal solution of RP to derive optimal solution $\bar{\pi}$ of DRP. In this case it turns out to be easy to derive optimal solution to DRP directly.

First note that if there is a path from s to t using only edges in J then optimal cost of RP is $\xi = 0$ (why?) and we have reached optimality in the original primal and dual.

We may therefore assume that there is no path from s to t using only edges in J.

DRP

$$
\begin{aligned}
\max w &= \pi_s \\
\pi_i - \pi_j &\le 0 \quad (i,j) \in J \\
\pi_i &\le 1 \\
\pi_i &\gtrless 0 \\
\pi_t &= 0
\end{aligned}
$$

We assume that there is no path from s to t using only edges in J. Note that $\pi_s \le 1$. So, if we can find a feasible solution of DRP with $\pi_s = 1$ we have optimality in DRP. Here is such a solution:

$$
\bar{\pi}_i = \begin{cases} 1 & \text{If } i \text{ is reachable from } s \text{ using arcs in } J \\ 0 & \text{If } t \text{ is reachable from } i \text{ using arcs in } J \\ 1 & \text{Otherwise} \end{cases}
$$

We then set

$$
\begin{aligned}
\theta_1 &= \min_{\substack{\text{arcs}(i,j)\notin J \\ s.t.\ \bar{\pi}_i - \bar{\pi}_j > 0}} \{c_{ij} - (\pi_i - \pi_j)\} \\
\pi &= \pi + \theta_1 \bar{\pi}
\end{aligned}
$$

update J and continue with our new DRP.

Note that we have replaced solving the original
Primal, Shortest Path problem,
with the repeated application of the simpler
DRP, Finding reachable nodes problem.

Our algorithm only terminates if there is a path from s to t in J which we saw implied optimality (shortest path). So, to prove optimality, suffices to prove that algorithm terminates.

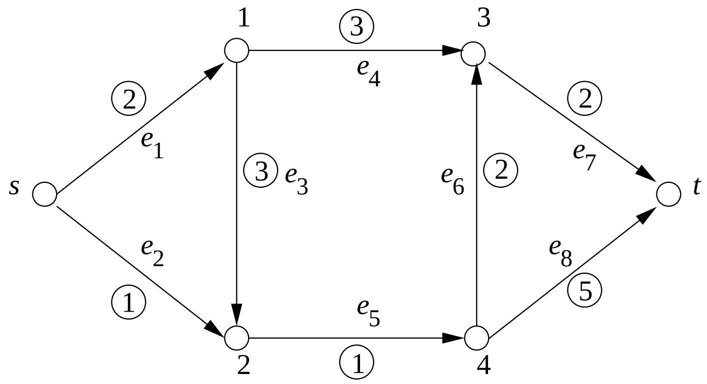
Technically, the proof for finiteness of generic Primal-Dual algorithm **doesn't work for this case** since it assumed that we used simplex to solve RP to give optimal solution to DRP. Since we solved DRP directly without solving RP, that proof doesn't apply here.

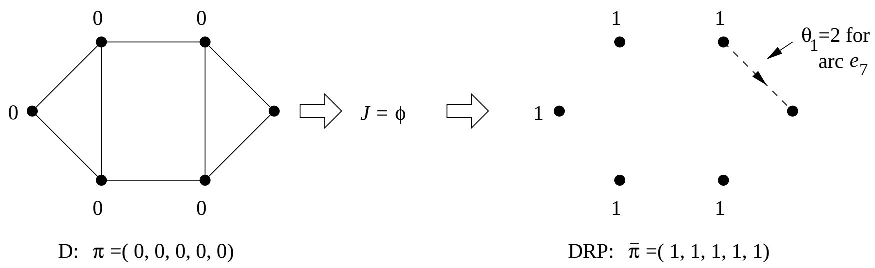
Finiteness, and therefore optimality, will follow, though, from two simple observations about DRP.

Lemma: Once edge (i, j) becomes admissible (enters J) it never leaves J at any later stage.

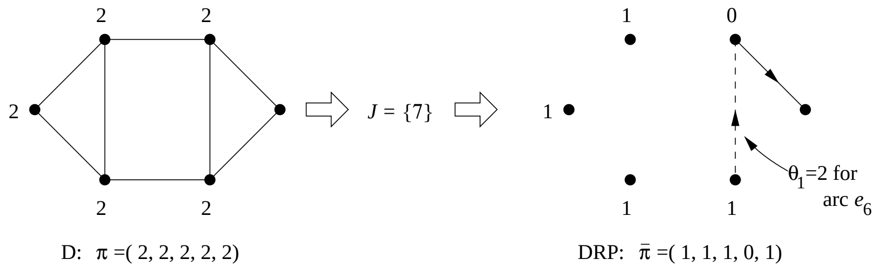
Lemma: At every iteration of DRP, at least one new (i, j), the one that defines θ_1, becomes admissible.

As an example we will see how to find the shortest $s-t$ path in

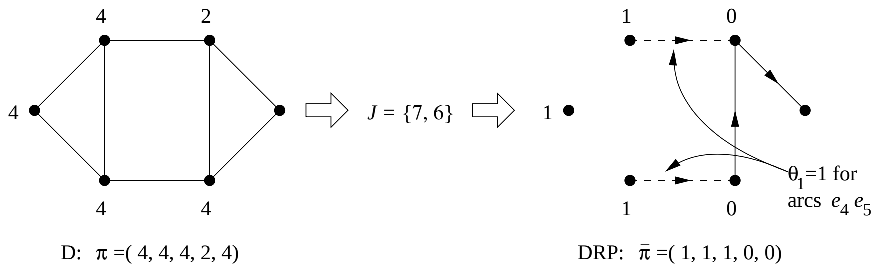




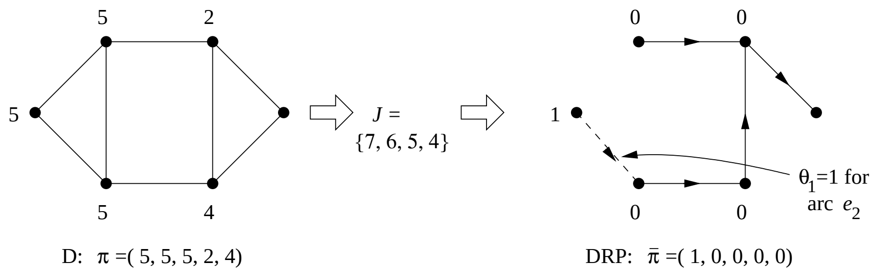
Iteration 1



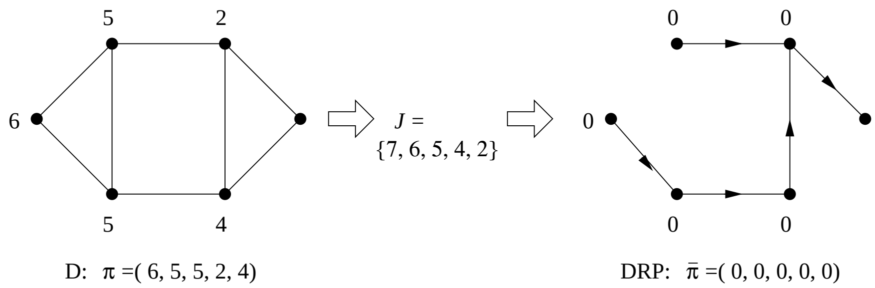
Iteration 2



Iteration 3



Iteration 4



Iteration 5

We will now see that the Primal-dual algorithm is, essentially, a disguised version of Dijkstra's shortest path algorithm. For simplicity, assume that all edge costs are ≥ 0 so we can start with feasible dual $\pi = 0$.
An iteration will be one step of solving DRP and updating D.

Let $W = \{j : t$ is reachable from j using edges in $J\}$.
When we first start, $J = \emptyset$ so $W = \{t\}$.

Lemma: Once a node enters W it never leaves W.
Each iteration adds at least one new node to W.

Lemma: The algorithm terminates in $< V$ iterations.

Lemma: Once j enters W, π_j never changes.
At any time, all $i \notin W$ share same value π_i.

$W = \{j : t$ is reachable from j using edges in $J\}$.

Lemma:
When (i', j') becomes admissible it satisfies

$$c_{i',j'} + \pi_{j'} = \min_{i \notin W,\, j \in W} \{c_{i,j} + \pi_j\}.$$

Proof: Let $\alpha = \pi_i$ for all $i \notin W$ at start of current iteration.Then

$$\begin{aligned}\theta_1 &= \min_{\substack{\text{arcs}(i,j)\notin J \\ s.t.\ \bar{\pi}_i - \bar{\pi}_j > 0}} \{c_{ij} - (\pi_i - \pi_j)\} \\ &= -\alpha + \min_{i \notin W,\, j \in W} \{c_{ij} + \pi_j\}\end{aligned}$$

Lemma:
When (i, j) becomes admissible, i enters W with $\pi_i = c_{i,j} + \pi_j$

Proof: $\pi^* = \pi + \theta_1 \bar{\pi}$
Then $\pi_i^* = \pi_i + (c_{ij} - (\pi_i - \pi_j))1 = c_{ij} + \pi_j$

$W = \{j : t$ is reachable from j using edges in $J\}$.
We start with $J = \emptyset$ and $W = \{t\}$.

Each iteration of the algorithm finds
Edge (i, j) that achieves $\min_{i \notin W,\, j \in W} \{c_{i,j} + \pi_j\}$
and then
i) adds the new edge(s) to J
ii) adds i to W with cost $\pi_i = c_{i,j} + \pi_j$.

But this is exactly the same as running
Dijkstra's shortest path algorithm
backwards from t.

Note what happened.

We started with the primal-dual algorithm

$$(P \Rightarrow)\ D \Rightarrow RP \Rightarrow DRP.$$

We then got rid of the explicit linear programming machinery by interpreting DRP as a *combinatorial* problem that could be solved using a combinatorial algorithm, rather than using a linear programming subroutine.

Notice also that even though the generic Simplex Algorithm and Primal-Dual algorithm are **not** polynomial, the algorithm we ended up with **is** polynomial!

The above are very typical attributes when using the primal dual algorithm to design combinatorial algorithms.

Max Flow

We will first write Max-Flow in Dual form. We have already seen that Max-Flow can be written using the node-arc incidence matrix as

$$\begin{aligned} &\max v \\ Af + dv &= 0 \\ f &\le b \\ f &\ge 0 \end{aligned}$$

Note that $Af + dv \le 0$ implies a flow deficit at some node. But this implies a flow surplus at some other node. So $Af + dv \le 0$ actually implies $Af + dv = 0$. Thus, problem above can be rewritten as

$$\begin{aligned} &\max v \\ Af + dv &\le 0 \\ f &\le b \\ f &\ge 0 \end{aligned}$$

which is in form of the Dual of a standard form LP.

Dual (D)

$$\begin{aligned} \max v & \\ Af + dv &\le 0 \\ f &\le b \\ f &\ge 0 \end{aligned}$$

DRP

$$\begin{array}{rcll} \max v & & & \\ Af + dv & \le & 0 & \text{for all rows} \\ f & \le & 0 & \text{for rows where } f = b \text{ in } D \\ -f & \le & 0 & \text{for rows where } f = 0 \text{ in } D \\ f & \le & 1 & \\ v & \le & 1 & \end{array}$$

Working through the technical steps we go from D to DRP. DRP can be interpreted as saying:
Find a path $\bar{\pi}$ from s to t that uses

- Saturated Arcs: only in backward direction
- Unused Arcs: Only in forward direction
- Other Arcs: Any direction.

Path π will be indicated by $\bar{\pi}_{i,j} = 1$ if (i, j) in the path, and 0, otherwise.

π is exactly an augmenting path in a residual network of original flow π.

One we find such a path we add as much as much flow as possible along the residual path by setting

$$\pi = \pi + \theta_1 \bar{\pi}$$

where we can work out that θ_1 is the bottleneck value along path $\bar{\pi}$.

This is exactly the Ford-Fulkerson augmenting path algorithm for Max-Flow.