IE 511: Integer Programming, Spring 2021 **25 Mar, 2021**

Lecture 18: Solution techniques: Primal and Primal-Dual

Lecturer: Karthik Chandrasekaran *Scribe: Karthik*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

18.1 Primal Technique: Max cardinality matching in bipartite graphs

We were seeing the primal technique to solve the max cardinality matching problem in bipartite graphs.

Recap

Weak duality: $|M| \leq |R|$ for every matching M and every vertex cover R in G.

Lemma 0.1. *A matching M is a maximum cardinality matching in G iff there is no M-augmenting path in G.*

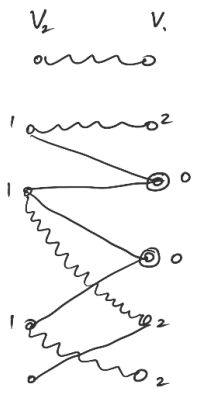
We have already seen that an M-augmenting path allows us to find a matching with larger size than the current matching M. Hence, by Lemma 0.1, in order to solve the max cardinality matching problem, it suffices to solve the following matching augmentation problem:

Matching Augmentation Problem

Given: A graph $G = (V, E)$ and a matching M in G.
Goal: Find an M-augmenting path or prove that no M-augmenting path exists.

We will design an algorithm for the matching augmentation problem in bipartite graphs. Let $G = (V_1 \cup V_2, E)$ where each edge has one end-vertex in V_1 and the other end-vertex in V_2.

We need to search for M-augmenting paths in a bipartite graph. An M-augmenting path P is of odd length and G is bipartite. Therefore, if one of the M-exposed vertices of P is in V_1 then the other M-exposed vertex of P should necessarily be in V_2. So, it suffices to start enumerating M-alternating paths from V_1.



We will search for an M-augmenting path along the following lines:

- Start labeling all vertices in V_1 that are M-exposed. These are the candidates to be the first vertex of an M-augmenting path.
- The first edge (odd edge) of an M-augmenting path should be a non-matching edge (i.e., in $E \setminus M$). So, all non-matching edges (i.e., edges in $E \setminus M$) incident to labeled vertices in V_1 are candidates. So label the end vertices of these edges in V_2.
- The second edge (even edge) of an M-augmenting path should be a matching edge (i.e., in M). So, all matching edges (i.e., edges in M) incident to labeled vertices in V_2 are candidates. So, label the end vertices of these edges in V_1.
- Follow the above rules repeatedly to label.
- If we cannot label then

  (i) either we have a vertex of V_2 that is labeled but M-exposed,

  (ii) or no more vertices can be labeled.

  If (i), then we have an M-augmenting path. If (ii), then we will see that the matching M is optimal.

The formal algorithm to solve the matching augmentation problem is summarized in Algorithm 1. We now prove the following guarantee about Algorithm 1 showing that if the algorithm terminates without finding an M-augmenting path, then M is a max cardinality matching.

Theorem 1. *Suppose that Algorithm 1 terminates with no new labels. Then,*

(i) $R = V_1^- \cup V_2^+$ *is vertex cover in* G *and*

(ii) $|M| = |R|$ *and hence,* M *is optimal.*

Proof. (i) See Figure 18.1. There is no edge $v_1v_2 \in E \setminus M$ such that $v_1 \in V_1^+$ and $v_2 \in V_2^-$ (otherwise, the algorithm would have continued labeling v_2 and not terminated). Also, there

Algorithm 1: Algorithm to solve Matching Augmentation

Input: $G = (V_1 \cup V_2, E)$ and a matching M in G

1. Give label 0 to each exposed vertex in V_1;

2. repeat

Scan labeled vertices $i \in V_1$ that are unscanned:
$\forall ij \in E \setminus M$ with j unlabeled, set label$(j) = i$;
Scan labeled vertices $j \in V_2$ that are unscanned:
if $\exists\ ji \in M$ **then**
set label$(i) = j$
else
j is M-exposed: Go to **3** with j
if *no new vertices are labeled* **then**
Go to **4**

3. (Augmentation): Use the labels to backtrack from $j \in V_2$ to find an M-augmenting path P and return $M \triangle E(P)$;

4. (No new label): Let V_1^+, V_2^+ be the vertices of V_1, V_2 that are labeled. Let $V_1^- := V_1 \setminus V_1^+$ and $V_2^- := V_2 \setminus V_2^+$;

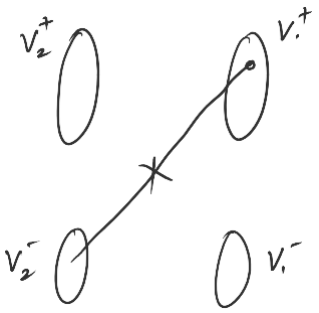


Figure 18.1: $V_1^- \cup V_2^+$ is a vertex cover.

is no edge $v_2v_1 \in M$ such that $v_2 \in V_2^-$ and $v_1 \in V_1^+$ that is in M (otherwise, the algorithm could not have labeled v_1). Therefore there are no edges within $V \setminus R$. Hence, $R = V_1^- \cup V_2^+$ is a vertex cover in G.

(ii) We know that $|M| \leq |R|$ because R is a vertex cover in G. We will show that $|M| \geq |V_1^- \cup V_2^+|$. That is, we will show that there are lot of matching edges.

Termination by 4 (i.e., due to no new label) and not by 3 (i.e., no augmenting path) implies that every vertex u of V_2^+ is incident to an edge e of M whose other end-vertex is in V_1^+.

Also, every vertex u of V_1^- is incident to an edge e of M (otherwise, such vertex will have labels and be in V_1^+). The other end-vertex of e should be in V_2^- (otherwise, u would have been labeled).

Therefore, we can associate a unique vertex for each edge of M that is in either V_2^+ or V_1^- (see Figure 18.2). Therefore, $|M| \geq |V_2^+| + |V_1^-| = |V_2^+ \cup V_1^-| = |R|$.

□

Next, we bound the run-time of Algorithm 1.

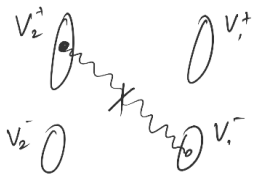


Figure 18.2: Every matching edge incident to a vertex in V_2^+ has its other end vertex in V_1^+; every matching edge incident to a vertex in V_1^- has its other end vertex in V_2^-.

Lemma 1.1. *Algorithm 1 terminates in at most $|V|$ iterations.*

Proof. Each iteration of the algorithm labels at least one new node. □

Note that each iteration of the algorithm runs in at most $|E|$ time. So, the total run-time of Algorithm 1 is $O(|V||E|)$. We can repeatedly use an augmentating path to increase the size of the matching; note that we can increase it at most $|V|/2$ times (since the size of a matching in the graph is at most $|V|/2$). Thus, we have proved the following theorem:

Theorem 2. *There exists an algorithm to find max cardinality matching in bipartite graphs that runs in $O(|V|^2|E|)$ time.*

18.2 Primal-Dual Technique: Assignment Problem

In the primal-dual technique to solve optimization problems, we will maintain a dual feasible solution and try to find a primal feasible solution that satisfies complementary slackness. We will illustrate this technique by using it to solve the assignment problem. Assignment problem is equivalent to the max weight perfect matching in bipartite graphs.

Assignment Problem

Given: A complete bipartite graph $G = (V_1 \cup V_2, E) : |V_1| = |V_2| = n, w : E \to \mathbb{R}$
Goal: $\max\{\sum_{ij \in M} w_{ij} : M \text{ is a perfect matching in } G\}$

$$\text{IP} := \max \left\{ \begin{array}{ll} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_{ij} : & \sum_{j=1}^n x_{ij} = 1 \ \forall i \in [n] \\ & \sum_{i=1}^n x_{ij} = 1 \ \forall j \in [n] \\ & x_{ij} \in \{0,1\} \ \forall i,j \in [n] \end{array} \right\}$$

We saw that it is sufficient to solve the following LP-relaxation, known as the Assignment LP:

$$\text{LP}(w) := \max \left\{ \begin{array}{ll} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_{ij} : & \sum_{j=1}^n x_{ij} = 1 \ \forall i \in [n] \\ & \sum_{i=1}^n x_{ij} = 1 \ \forall j \in [n] \\ & x_{ij} \geq 0 \ \forall i,j \in [n] \end{array} \right\}$$

We will develop an algorithm to find an integral optimum solution to the assignment LP. We will refer to a feasible solution to the assignment LP as an *assignment*. Note that an assignment x could be fractional. We have the following dual to the assignment LP (note that the variables are u and v below and they are free variables with no non-negativity conditions, see Figure 18.3):

$$D(w) := \min \left\{ \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j : u_i + v_j \geq w_{ij} \ \forall i, j \in [n] \right\}.$$

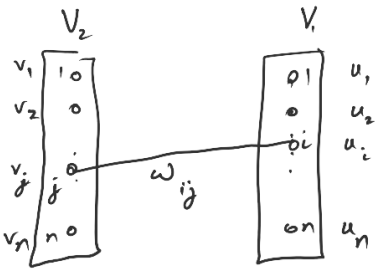


Figure 18.3: The dual program has variables corresponding to the vertices of the graph and constraints corresponding to the edges of the graph.

We can use complementary slackness to derive an optimality condition for the assignment LP.

Lemma 2.1. *Suppose that there exist $u_1, \ldots, u_n, v_1, \ldots, v_n \in \mathbb{R}$ and an assignment x such that*

(i) $u_i + v_j - w_{ij} \geq 0 \ \forall i, j \in [n]$

(ii) $x_{ij} = \begin{cases} 1 & \text{only if } u_i + v_j = w_{ij} \\ 0 & \text{otherwise.} \end{cases}$

Then x is optimal to LP(w).

Proof. Duality and complementary slackness. □

Algorithm Outline: Maintain a dual feasible solution (u, v). Try to find a perfect matching using only the *tight* edges, i.e., using only the edges in

$$\bar{E} := \{ij : u_i + v_j = w_{ij}\}.$$

If we find such a perfect matching, then we have an optimum solution. If not, then the current dual solution is not optimum so we will modify the dual values. We formalize this in Algorithm 2.

Algorithm 2: Let u, v be an initial dual feasible solution.

Repeat:

1. Primal step:

   $\bar{E} := \{ij : u_i + v_j = w_{ij}\}$.

   Find a max cardinality matching M in $\bar{G} = (V_1 \cup V_2, \bar{E})$ (using Algorithm 1).

   If $|M| = n$, STOP and output M.

   Else, let V_1^+, V_2^+ be the labeled vertices upon termination of the max cardinality matching algorithm. Let $V_1^- := V_1 \setminus V_1^+, V_2^+ = V_2 \setminus V_2^+$.

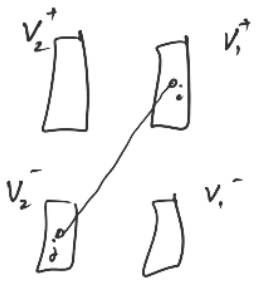


Figure 18.4: Edges of interest in the dual update step

2. Dual update:

   $\delta := \min\{u_i + v_j - w_{ij} : i \in V_1^+, j \in V_2^-\}$ (see Figure 18.4).

   $u_i \leftarrow u_i - \delta \ \forall i \in V_1^+$.

   $v_j \leftarrow v_j + \delta \ \forall j \in V_2^+$.

We now analyze the algorithm. We begin by showing that the dual update indeed modifies the dual solution.

Proposition 3. *$\delta > 0$ in dual update.*

Proof. We have already seen in the analysis of Algorithm 1 that there is no edge of $\bar{E}$ between V_1^+ and V_2^-. This means that $u_i + v_j - w_{ij} > 0 \ \forall i \in V_1^+, j \in V_2^-$. □

Correctness. We now show that the algorithm is indeed correct. For this, we need to show that the dual update preserves dual feasibility.

Lemma 3.1. *After a dual update, the new solution (u^{new}, v^{new}) is still dual feasible.*

Proof. Let $i, j \in [n]$. We have four possibilities.

i	j	
V_1^+	V_2^+	$\implies u_i^{\text{new}} + v_j^{\text{new}} - w_{ij} = u_i^{\text{old}} - \delta + v_j^{\text{old}} + \delta - w_{ij} \geq 0.$
V_1^-	V_2^-	$\implies u_i^{\text{new}} + v_j^{\text{new}} - w_{ij} = u_i^{\text{old}} + v_j^{\text{old}} - w_{ij} \geq 0.$
V_1^-	V_2^+	$\implies u_i^{\text{new}} + v_j^{\text{new}} - w_{ij} = u_i^{\text{old}} + v_j^{\text{old}} + \delta - w_{ij} \geq 0.$
V_1^+	V_2^-	$\implies u_i^{\text{new}} + v_j^{\text{new}} - w_{ij} = u_i^{\text{old}} - \delta + v_j^{\text{old}} + \delta - w_{ij} \geq 0$ (by choice of δ).

□

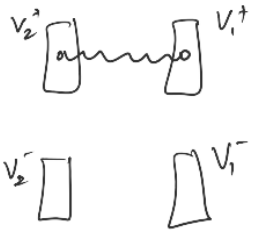
By Lemmas 2.1 and 3.1, we have the following result:

Corollary 3.1. *If algorithm terminates, then it returns an optimal solution.*

Termination/Runtime. We now show that the algorithm will terminate and in particular, it will terminate in polynomial number of iterations. Suppose that the algorithm has not terminated at the end of the primal step. Under this assumption, we will prove a series of claims that will show that the algorithm is making progress.

Claim 3.1. $|V_1^+| > |V_2^+|$.

Proof.



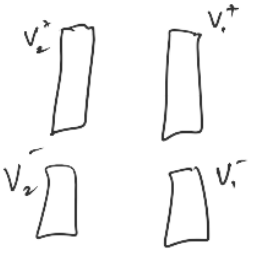
Each vertex of V_2^+ is adjacent to a matching edge whose other end vertex is in V_1^+. This tells us that $|V_1^+| \geq |V_2^+|$. But V_1 has at least one exposed node. Therefore, $|V_1^+| > |V_2^+|$. □

From Claim 3.1 and Corollary 3.1, the dual objective value after the dual update step *decreases* by $\delta\left(|V_1^+| - |V_2^+|\right) \geq \delta > 0$. This fact suggests that the algorithm makes improvements towards the dual optimum.

Caution: Although this fact implies that the algorithm is making progress, it does not imply finite termination: it is possible that δ is becoming smaller and smaller and the objective value of the dual solution constructed by the algorithm keeps getting closer and closer to the dual optimum objective value but never attains the dual optimum objective value. So, we need more arguments to show termination. The next claim is the crucial observation that allows us to show finite termination.

Claim 3.2. *The labels V_1^+ and V_2^+ are valid after the dual update, i.e., they still correspond to M-alternating paths.*

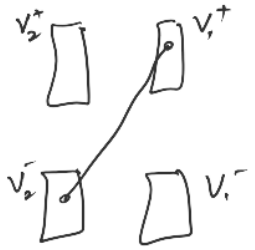
Proof.



For $i \in V_1^+$ and $j \in V_2^+$ we have $u_i^{\text{new}} + v_j^{\text{new}} - w_{ij} = u_i^{\text{old}} + v_j^{\text{old}} - w_{ij} \geq 0$. Graph G^{new} restricted to the vertices in $V_2^+ \cup V_1^+$ is a supergraph of $\bar{G}^{\text{old}}$ restricted to the vertices in $V_2^+ \cup V_1^+$. □

Claim 3.3. *If $|M|$ has not increased after the next primal step, then $|V_2^+|$ increases.*

Proof.



By Claim 3.2 and by the choice of δ, an edge ij between a vertex $i \in V_1^+$ and a vertex $j \in V_2^-$ is added to $\bar{E}$. This enables j to become a labeled vertex in the next primal step. □

Claim 3.3, $|V_2^+| \leq n$, and Claim 3.2, together imply that the cardinality of the maximum matching found in the primal step must increase after at most n dual updates. Also, the cardinality of the matching can increase at most n times. Thus, we have the following corollary:

Corollary 3.2. *Algorithm 2 executes at most n^2 dual updates. Each dual update examines at most n^2 edges. Hence, Algorithm 2 can be implemented to run in $O(n^4)$ time.*

Hence, we have shown the following theorem:

Theorem 4. *There exists an algorithm that solves the assignment problem in $O(n^4)$-time.*