# Minimum Cost Perfect Matching

March 24, 2017

**Abstract**

This is pseudocode for the Hungarian Algorithm for Minimum Cost Perfect Matching in Bipartite Graphs, adapted from the paperback edition of *A Gentle Introduction to Optimization*, B. Guerin *et al.* This finds subgraphs of the original bipartite graph (which is often a complete bipartite graph), and uses the Perfect Matching algorithm as a subroutine to figure out if there is a perfect matching, in which case we are done.

$N_H(S)$ is the set of neighbors of the set $S$ on the graph $H$. One thing I do not like about this particular pseudocode is that the interface with the perfect matching subroutine is imprecise, and this algorithm depends on side effects from that subroutine. The subroutine might return (a) a new (larger) matching $M$, (b) a new (larger) tree $T$, or (c) a deficient set for that particular graph $H$.

**Algorithm 3.5** Minimum cost perfect matching in bipartite graphs (fast version)

**Input :** Bipartite graph $G = (V, E)$ with bipartition $U, W$ where $|U| = |W| \geq 1$

**Output:** A minimum cost perfect matching $M$ or a deficient set $S$

$M := \varnothing$

$T := (\{r\}, \varnothing)$ where $r \in U$ is any $M$-exposed vertex

$\vec{y_v} := \frac{1}{2} \min \{c_e : e \in E\}$, for all $v \in V$

**while** *(1)* **do**

{

  Construct graph $H$ (a subset of the original graph $G$)

  with vertices $V$ and edges $\{uv \in E : c_{uv} = \vec{y_u} + \vec{y_v}\}$

  Invoke the subroutine (Algo. 3.4) with $H$,$M$, and $T$

  **if** *outcome (a) of subroutine occurs* **then**

  {

    **if** *$M$ is a perfect matching of $H$* **then**

    {

      **stop** ($M$ is minimum cost perfect matching of $G$)

    }

  }

  **else if** *outcome (c) of subroutine occurs* **then**

  {

    Let $S := B(T)$

    **if** *all edges of $G$ with an endpoint in $S$ have an endpoint in $N_H(S)$* **then** { **stop** ($G$ has no perfect matching) ; }

    Otherwise, adjust the costs to create a new zero, and loop

    $\epsilon = \min\{c_{uv} - \vec{y_u} - \vec{y_v} : u \in S, v \notin V(T)\}$

$$\vec{y_v} := \begin{cases} \vec{y_v} + \epsilon \text{ for } v \in S \\ \vec{y_v} - \epsilon \text{ for } v \in N_H(S) \\ \vec{y_v} \text{ otherwise} \end{cases} \quad (1)$$

  }

  **else if** *outcome (b) of subroutine occurs* **then**

  {

    Just loop again with augmented path

  }

}