Graphs and Algorithms

Weighted Bipartite Matching

Recall the algorithm to determine a perfect matching of maximum weight in a bipartite graph from the lecture. It consists of two nested repeat loops. In the outer loop, a matching $M$ in the equality graph $G_\ell$ is build up step by step, in the inner loop the labelling function $\ell$ is modified until an augmenting path appears that allows to extend the matching $M$. Clearly the outer loop takes $O(n)$ iterations, and in the lecture you have seen that the running time of the inner loop can be bounded by $O(n^3)$. We show that this estimate is very crude and that the running time of the inner loop can be bounded by $O(n^2)$, giving an overall running time of $O(n^3)$.

We call one iteration of the outer loop a phase, and consider the inner repeat loop during one phase. First observe that the sets $S$ and $T$ do not have to be calculated from scratch every time. We only have to continue our breadth first search along edges that have appeared between the sets $S$ and $B \setminus T$. Hence the cumulated time needed to calculate $S$ and $T$ during one phase is $O(n^2)$ (we run one BFS over many iterations).

We introduce an array slack[] that is defined for all vertices $b \in B \setminus T$ as

$$\text{slack}[b] := \min_{a \in S} \{(\ell(a) + \ell(b)) - w(a,b)\}. \tag{1}$$

Moreover, we keep a list edges[] that contains all edges from $b$ to vertices from $S$ for which equality holds in (1). In this way $\lambda$ can be calculated as $\lambda := \min_{b \in B \setminus T} \text{slack}[b]$ in time $O(n)$. Also $G_\ell$ can be updated in time $O(n)$, as we only have to add all edges from edges[] for all $b \in B \setminus T$ with $\lambda = \text{slack}[b]$ to it (actually we also have to remove all edges between $A \setminus S$ and $T$ from $G_\ell$, but as they do not interfere our current BFS, we do this once at the end of a phase).

The initialization of slack[] and edges[] in the beginning of a phase takes $O(n^2)$ time. We have to update them as soon as a new vertex is added to $S$. Updating takes $O(n)$ time per vertex and as there are at most $n$ vertices in $S$, we need $O(n^2)$ steps in total.

Hence we have shown that each phase takes only $O(n^2)$ time.

Hamilton Cycles in the Hypercube

Obviously the stronger statement holds for $n = 2$. Any perfect matching in $K(H_2) = K_4$ can be extended to a Hamilton cycle using only edges from $H_2 = C_4$.

Now suppose the statement is true for $n \geq 2$ and consider a perfect matching $M$ in $K(H_{n+1})$. Clearly we can choose a coordinate direction such that when cutting apart $K(H_{n+1})$ orthogonally to this direction into two copies of $K(H_n)$ (call them $G_1$ and $G_2$), then there is a nonempty subset of edges $M' \subseteq M$ going across the cut. As the number of vertices of $G_1$ and $G_2$ is even and $M$ is a perfect matching, $|M'| = 2k$ for some integer $k$. This implies that an even number of vertices from $G_1$ and an even number of vertices from $G_2$ are not saturated by edges from $M_1 := M \cap E(G_1)$ and $M_2 := M \cap E(G_2)$, respectively. We can add any set $A_1 \subseteq E(G_1)$ of $k$ edges to $M_1$ to make it a perfect matching in $G_1$ (these edges are not necessarily hypercube edges) and by the induction hypothesis we obtain a set $B_1$ of hypercube edges such that
$A_1 \cup M_1 \cup B_1$ is a Hamilton cycle in $G_1$. Now define a set of edges $A_2$ as $A_2 = \{ e \in E(G_2) \mid e \text{ together with two edges from } M' \text{ and a path in } M_1 \cup B_1 \text{ forms a cycle in } K(H_{n+1}) \}$. Note that $|A_2| = k$ and $A_2 \cup M_2$ is a perfect matching in $G_2$. By the induction hypothesis there is a set of hypercube edges $B_2$ such that $A_2 \cup M_2 \cup B_2$ is a Hamilton cycle in $G_2$. We conclude by observing that $B_1 \cup B_2$ is a set of hypercube edges that extends $M$ to a Hamilton cycle in $K(H_{n+1})$ (the two Hamilton cycles in $G_1$ and $G_2$ are glued together via the edges from $M'$).