Warsaw PhD Open Course: From Joins to Aggregates and Optimization Problems - Exam solutions

Djordje Zivanovic

January 27, 2019

1 Let $Q(\mathbf{A_1} \cup \mathbf{A_2} \cup \cdots \cup \mathbf{A_n})$ denote a join query on relations $R_1(\mathbf{A_1}) = R_1(A_1, A_2), \cdots R_n(\mathbf{A_n}) = R_n(A_n, A_1)$. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ denote a corresponding hypegraph to the join query, then $\mathcal{V} = \{A_1, .., A_n\}$ and $\mathcal{E} = \{\{A_1, A_2\}, ..., \{A_n, A_1\}\}$.

1.1 Let x_{R_i} denote the weight of an hyperedge (relation) R_i in the join query Q. In addition let $cover_i$ denote the set of relations corresponding to edges that are part of the cover for the variable i. The fractional edge cover number $\rho^*(Q)$ is the cost of an optimal solution to the linear program:

$$\text{minimize} \sum_{i \in [n]} x_{R_i} \tag{1}$$

$$\text{subject to} \sum_{rel_j \in cover_i} x_{rel_j} \geq 1, i \in [n] \tag{2}$$

$$x_{R_i} \geq 0, i \in [n]$$

By a definition of a cover, only edges incident to variable are part of its cover. Thus, in our case $cover_i = \{R_{i-1}, R_i\}$, where R_0 is R_n in the case of A_1. If we rewrite the inequalities 2 we will get:

$$\begin{aligned} x_{R_1} + x_{R_n} &\geq 1 \\ x_{R_1} + x_{R_2} &\geq 1 \\ &\vdots \\ x_{R_{n-1}} + x_{R_n} &\geq 1 \end{aligned}$$

By summing inequalities we get:

$$2 \left(\sum_{i \in [n]} x_{R_i} \right) \geq n$$

This means the minimized function $\rho^*(Q)$ cannot be less than $\frac{n}{2}$. This is achievable if we set $x_{R_i} = \frac{1}{2}, i \in [n]$. We can see that this solution satisfies all constraints in the linear program. Finally, $\rho^*(Q) = \frac{n}{2}$.
A width of a specific hypertree decomposition is defined as a maximum among all edge covers for each of the nodes of the hypertree. A hypertree width (htw) for the specific query Q is defined as a minimum width among all possible widths of hypertree decompositions of a query Q.
A fractional width of a specific hypertree decomposition is defined as a maximum among all fractional edge covers for each of the nodes of the hypertree. A fractional hypertree width (fhtw) for the specific query Q is defined as a minimum width among all possible widths of hypertree decompositions of a query Q. The only difference is that in the case of fractional hypertree width we use fractional edge covers instead of edge covers.
In the case of $n = 2$ $fhtw(Q) = htw(Q) = 1$. For $n >= 3$ the hypertree decomposition on the figure 1 is the optimal one. From picture we can figure out that $htw(Q) = 2$ in all cases, while $fhtw(Q) = 2$ for $n > 3$, but for $n = 3$ it is $\frac{3}{2}$.
The decomposition on the figure 1 is optimal because we have to "propagate" one variable through nodes of the hypertree in order to fulfill the condition that all hyperedges of hypergraph are contained in the nodes of hypertree. Adding more than three variables inside a node of hypertree would increase (fractional) width of a hypertree decomoposition because we would need to cover at least two variables that are not in the same relation. (Fractional) edge cover number for each of nodes of hypegraph inside nodes of hyperetree is 2 because $A_1 A_i A_{i-1}$ can be covered with $x_{R_i} = 1, x_{R_1} = 1$. The only exception to the previous case is $n = 3$ where fractional edge cover of $\frac{3}{2}$ is possible.

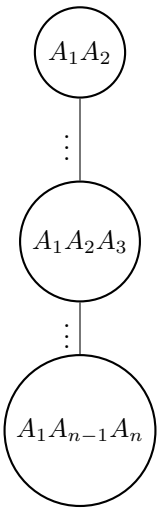

Figure 1: Hypertree decomposition that corresponds to fhtw

1.2 A size of listing representation is $\mathcal{O}(N^{\rho^*(Q)}) = \mathcal{O}(N^{\frac{n}{2}})$ and can be computed in time $\mathcal{O}(N^{\rho^*(Q)}) = \mathcal{O}(N^{\frac{n}{2}})$. A loss function is:

$$\begin{aligned} 2|D|\mathcal{L}(\mathbf{X},\mathbf{y},\theta) &= (\mathbf{y}-\mathbf{X}\theta)^2 + \lambda||\theta||_2^2 \\ &= (\mathbf{y}-\mathbf{X}\theta)^{\mathrm{T}}(\mathbf{y}-\mathbf{X}\theta) + \lambda||\theta||_2^2 \\ &= \mathbf{y}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{X}\theta - \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda||\theta||_2^2 \\ &= \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\mathbf{y}^{\mathrm{T}}\mathbf{X}\theta + \theta^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda\theta^{\mathrm{T}}\theta \end{aligned} \tag{3}$$

In the equation 3 we used the property that transpose of a scalar is the same scalar. All equations before that are simple vector multiplications and additions. If we derive the equation 3 we get the gradient of the loss function: :

$$\begin{aligned} \nabla_\theta\mathcal{L}(\mathbf{X},\mathbf{y},\theta) &= 0 - \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta + \lambda\frac{1}{|D|}\theta \\ &= \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{X}\theta - \frac{1}{|D|}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \lambda\frac{1}{|D|}\theta \end{aligned}$$

A gradient descent formula is:

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta\mathcal{L}(\mathbf{X},\mathbf{y},\theta)$$

After joining relations we need in each step to calculate gradient. We can notice that $\mathbf{X}^{\mathrm{T}}\mathbf{y}$ can be cached and the complexity to do so is $\mathcal{O}(n\cdot N^{\frac{n}{2}})$. Similarly with $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, whose calculation takes $\mathcal{O}(n^2\cdot N^{\frac{n}{2}})$. In each iteration we need to multiply $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ with θ which takes $\mathcal{O}(n^2)$ time. When we add all time complexities we get:

$$\mathcal{O}(N^{\frac{n}{2}} + n^2\cdot N^{\frac{n}{2}} + n\cdot N^{\frac{n}{2}} + n^2\cdot t)$$

Memory complexity after the join is $\mathcal{O}(N^{\frac{n}{2}})$. $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ cached takes $\mathcal{O}(n^2)$ and $\mathbf{X}^{\mathrm{T}}\mathbf{y}$ takes $\mathcal{O}(n)$. When we add these complexities we get memory complexity at each iteration is: $\mathcal{O}(n^2)$.