Overview of MATLAB Code for generating lower bounds on $R^*(M)$
Written by Hooshang Ghasemi

Email: ghasemi@iastate.edu

Please feel free to ask me about the code and paper

June 24, 2016

In our paper, H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching.", available at http://www.arxiv.org/pdf/1501.06003.pdf, we introduced a new technique to establish lower bounds on the rate for the coded caching problem. Consider a coded caching systems consisting of a server with N files connected to K users each equipped with a cache of size M (normalized with respect to the file size) through a shared link. Let R be rate of the shared link as defined in the paper.

1 An summary of results in the paper

In this section we present a top level overview of the results in [1]. In what follows, we refer extensively to results in the paper. This section can be skipped if you are only interested in generating the actual bounds. For this system we are interested in establishing a lower bound like:

$$\alpha \mathtt{R} + \beta \mathtt{M} \geq \mathtt{L}$$

Corollary 2 of the paper gives us,

$$\alpha \mathtt{R} + \beta \mathtt{M} \geq \min(\alpha \min(\beta, \mathtt{K}), \alpha_l \min(\beta_l, \mathtt{K}) + \alpha_r \min(\beta_r, \mathtt{K}) + \mathtt{N} - \mathtt{N}_0) \quad (1)$$

provided that $\mathtt{N} \geq \mathtt{N}_0$ where,

$$\begin{aligned} &\alpha_l = \mathtt{ceil}(\alpha/2), \qquad \beta_l = \mathtt{floor}(\beta/2), \\ &\alpha_r = \alpha - \alpha_l, \qquad \beta_r = \beta - \beta_l, \\ &\mathtt{N}_0 = \max(\mathtt{N}_{sat}(\alpha_l, \beta_l, \mathtt{K}), \mathtt{N}_{sat}(\alpha_r, \beta_r, \mathtt{K})). \end{aligned}$$

Therefore,

$$\mathtt{L} = \min(\alpha \min(\beta, \mathtt{K}), \alpha_l \min(\beta_l, \mathtt{K}) + \alpha_r \min(\beta_r, \mathtt{K}) + \mathtt{N} - \mathtt{N}_0)$$

provided that $\mathtt{N} \geq \mathtt{N}_0$.

2 Manual for using the code

There are two ways to generate the lower bounds. There is an web-based form available at `http://www.ece.iastate.edu/~ghasemi/Lower%20bounds%20on%20coded%20caching` where one can simply enter appropriate parameters and generate the bounds. One can also download the corresponding MATLAB code that generates the lower bounds. This will return the actual problem instance that corresponds to the best lower bound. The corresponding inequalities can be obtained by interpreting the problem instance according to the steps in [1].

The Matlab code contains two functions called "`comp_bound.m`" and "`ComputeNsat.m`". The main function to compute the lower bound is "`comp_bound.m`". The function "`ComputeNsat.m`" is used to upper bound the saturation number $\mathtt{N}_{sat}(\alpha, \beta, \mathtt{K})$ and $\mathtt{N}_0$. The file "`Example_genr_bound.m`" is an example of using "`comp_bound.m`" to generate lower bounds. One can use `comp_bound.m` with different inputs as explained below.

1. Inputs: `N`, `K`, `M`, `alpha` , `beta`.

   Use the following command:

   $$\texttt{comp\_bound}(\mathtt{N}, \mathtt{K}, \mathtt{M}, \texttt{alpha}, \texttt{beta}).$$

   Then the function returns `R` so that we have the following bound.

   $$\mathtt{R} \geq (\mathtt{L} - \texttt{beta} * \mathtt{M})/\texttt{alpha};$$

   Note: In this case the code calls "`ComputeNsat.m`" to compute upper bounds on $\mathtt{N}_{sat}(\alpha, \beta, \mathtt{K})$ and $\mathtt{N}_0$. If $\mathtt{N} \geq \mathtt{N}_{sat}(\texttt{alpha}, \texttt{beta}, \mathtt{K})$ then $\mathtt{L} = \texttt{alpha} * \min(\texttt{beta}, \mathtt{K})$. If $\mathtt{N} < \mathtt{N}_0$ then the problem instance corresponding to given setting is non-atomic (please look at the paper for definition of non-atomic problem instance) and the result is $\mathtt{R} = 0$. When the problem instance is non-atomic the code will return $\mathtt{R} = 0$ as output and print out the following message.

   ```
   Problem instance associated with given alpha and beta is non-atomic.
   For definition of atomic problem instance please refer to the manuscript.
   Please try smaller alpha and beta.
   ```

   A non-atomic problem instance can be represented by two or more atomic problem instances with smaller `alpha` and `beta`. Therefore, it does not provide new lower bounds on the rate. In this situation, we suggest that you try lower values of `alpha` and `beta`.

2. Inputs are: N, K, alpha , beta.

   Use the following command:

   $$\texttt{comp\_bound}(\texttt{N}, \texttt{K}, \texttt{alpha}, \texttt{beta}).$$

   Then the function will return L and print out the following bound.

   $$\texttt{alpha} * \texttt{R} + \texttt{beta} * \texttt{M} \geq \texttt{L}$$

   Note: In this case the code calls "ComputeNsat.m" to compute upper bounds on $\texttt{N}_{sat}(\alpha, \beta, \texttt{K})$ and $\texttt{N}_0$. If $\texttt{N} \geq \texttt{N}_{sat}(\texttt{alpha}, \texttt{beta}, \texttt{K})$ then $\texttt{L} = \texttt{alpha} * \min(\texttt{beta}, \texttt{K})$. If $\texttt{N} < \texttt{N}_0$ then the problem instance corresponding to given setting is non-atomic (please look at the paper for definition of non-atomic problem instance) and then $\texttt{L} = 0$. When the problem instance is non-atomic the code will return $\texttt{R} = 0$ as output and print out the following message.

   ```
   Problem instance associated with given alpha and beta is non-atomic.
   For definition of atomic problem instance please refere to the manuscript.
   Please try smaller alpha and beta.
   ```

   A non-atomic problem instance can be represented by two or more atomic problem instances with smaller alpha and beta. Therefore, it does not provide new lower bounds on the rate. In this situation, we suggest that you try lower values of alpha and beta.

3. Inputs are: N, K, M.

   Use the following command:

   $$\texttt{comp\_bound}(\texttt{N}, \texttt{K}, \texttt{M}).$$

   In this case the function uses an appropriate range of alpha and beta and generates lower bounds for all these cases. The final bound returned is the maximum over all of them. The specific range of the parameters is $\texttt{alpha} \in [1, 2 * \texttt{N}]$ and $\texttt{beta} \in [1, 2 * \texttt{K}]$. For the interested reader, an explanation of why these are the appropriate ranges is given in Section 3 (this will require some understanding of the concepts in [1]).

   $$\texttt{R} \geq \max_{\texttt{alpha} \in [1, 2*\texttt{N}]} \max_{\texttt{beta} \in [1, 2*\texttt{K}]} (\texttt{L} - \texttt{beta} * \texttt{M})/\texttt{alpha}$$

3 Ranges of α and β

We first show that considering $\beta \leq 2K$ is sufficient. This is because if $\beta \geq 2K$, our algorithm for splitting $\beta = \beta_l + \beta_r$ will be such that $\beta_l, \beta_r \geq K$. For this range of β eq. (1) will reduce to

$$\begin{aligned} \alpha R + \beta M &\geq \min(\alpha K,\ \alpha K + N - N_0) = \alpha K, \\ &\Rightarrow R \geq K - \beta M/\alpha. \end{aligned}$$

Therefore, choosing β to be larger than $2K$ will not result in any improvement as far as the lower bound on R is concerned.

Nex, we show that it suffices to consider $\alpha \leq 2N$. Toward this end, first note that $N_{sat}(\alpha, \beta, K) \geq \alpha$ for $\beta \leq K$. This can be seen by showing that $N_{sat}(\alpha, \beta, K) < \alpha$ for $\beta \leq K$ results in a contradiction. Firstly, we note that $N_{sat}(\alpha, 1, K) = \alpha$ (by inspection). We let $P_{sat}(\mathcal{T}, N_{sat}, K, L, \alpha, \beta)$ to be the problem instance associated with saturation number $N_{sat}(\alpha, \beta, K)$ so that $L = \alpha\beta$. According to Claim 3 in [1] w.l.o.g. we can assume that $\hat{\beta} = \beta$ in problem instance P_{sat}. Furthermore, as $\psi(v_i, v') \in \{0, 1\}$ and $L = \sum_{i=1}^{\alpha} \sum_{v' \in \mathcal{C}} \psi(v_i, v')$ in eq. (7) of [1] and $|\mathcal{C}| = \hat{\beta} = \beta$, it has to be the case that $\sum_{i=1}^{\alpha} \psi(v_i, v') = \alpha$ holds for all $v' \in \mathcal{C}$. This implies that each cache node, $v' \in \mathcal{C}$, contributes exactly α towards the lower bound L. Therefore, if we retain only one cache node of $\mathcal{C}$ we can come up with a new problem instance $P'(\mathcal{T}', N', K, L', \alpha, 1)$ such that $L' = \alpha$ and $N' \leq N_{sat}(\alpha, \beta, K)$ (as only a subset of recovered files in P_{sat} is used in P'). But $N' \leq N_{sat}(\alpha, \beta, K) < \alpha$ and $L' = \alpha$ contradict the fact that $N_{sat}(\alpha, 1, K) = \alpha$ since this implies that we have found a saturated problem instance P' with $L' = \alpha$ and $N' < N_{sat}(\alpha, 1, K)$.

So far we have shown that $N_{sat}(\alpha, \beta, K) \geq \alpha$ for $\beta \leq K$. Now we show that a problem instance with $\alpha \geq 2N$ is non-atomic thus does not provide a new lower bound. For $\alpha \geq 2N$ we have either $\alpha_l \geq N$ or $\alpha_r \geq N$ in inequality (1). W.l.o.g we assume that $\alpha_l \geq N$. Then by argument we made $N_{sat}(\alpha_l, \beta_l, K) \geq \alpha_l \geq N$ which further implies $N_0 \geq N$. Therefore, no new file will be recovered in last edge as all files are recovered in upstream nodes, i.e., the instance is non-atomic.

References

[1] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," preprint, 2016, [Online] Available: http://arxiv.org/abs/1501.06003.