Designing an efficient dual solver for discrete energy minimization Appendix

Contents:

A. PROOFS

Theorem 1. Let $\mathcal{G}'$ be a tree and $\sum_{uv\in\mathcal{E}'} \min_{s,t} \theta^{\phi}_{uv}(s,t) = 0$. The function $g(y) = \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u)$ is a minorant for the energy $E_{\mathcal{G}'}(y)$ *if and only if* ϕ is dual optimal on $\mathcal{G}'$.

Proof. **The "if" part** Let ϕ be a dual optimal reparameterization for $E_{\mathcal{G}'}$ on the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$. We need to show that $g(y)$ is a minorant. *i.e.*

- $g(y) \leq E_{\mathcal{G}'}(y)$ for all y. (`lower-bound property`)
- $g(y^*) = E_{\mathcal{G}'}(y^*)$ is the cost of a minimizing labeling y^*. (`same-minima property`)

We have by the definition of reparametrization

$$\sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \theta^{\phi}_{uv}(y_u, y_v) = \sum_{u\in\mathcal{V}'} \theta_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \theta_{uv}(y_u, y_v) = E_{\mathcal{G}'}(y). \tag{15}$$

We assume w.l.o.g. $\theta^{\phi}_{uv}(y_u, y_v) \geq 0$. Substituting this in (15) we have

$$g(y) = \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) \leq \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \theta^{\phi}_{uv}(y_u, y_v) = E_{\mathcal{G}'}(y), \forall y \in \mathcal{Y}^n \implies g(y) \leq E_{\mathcal{G}'}(y), \forall y \in \mathcal{Y}^n, \tag{16}$$

where the left hand side matches the definition of $g(y)$ in the theorem. With this we have proved the lower bound property.

Now we prove the same minima property. Comparing the dual function (3) with $g(y)$ and $g(y)$ with (1) we have the following inequalities:

$$D(\phi) = \sum_{u\in\mathcal{V}'} \min_{y_u} \theta^{\phi}_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \min_{y_u, y_v} \theta^{\phi}_{uv}(y_u, y_v) \leq \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \min_{y_u, y_v} \theta^{\phi}_{uv}(y_u, y_v) = g(y), \forall y \in \mathcal{Y}^n; \tag{17}$$

$$g(y) = \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) = \sum_{u\in\mathcal{V}'} \theta^{\phi}_{u}(y_u) + \sum_{uv\in\mathcal{E}'} \theta^{\phi}_{uv}(y_u, y_v) \leq E_{\mathcal{G}'}(y), \forall y \in \mathcal{Y}^n. \tag{18}$$

Since $\mathcal{G}'$ is a tree-subgraph, strong duality holds and we have for all pairs of an optimal labeling y^* and an optimal dual ϕ that $D(\phi) = E_{\mathcal{G}'}(y^*)$ and there holds complementarity slackness conditions. It follows that $\min_{y_u} \theta^{\phi}_{u}(y_u)$ is attained at y^*_u and $\min_{y_u, y_v} \theta^{\phi}_{u,v}(y_u, y_v)$ is attained at (y^*_u, y^*_v) (there is an optimal solution composed of minimal nodes and edges). It follows that the next inequalities are satisfied:

$$\theta^{\phi}_{u}(y_u) \geq \theta^{\phi}_{u}(y^*_u), \forall y_u \in \mathcal{Y}; \tag{19}$$

$$\theta^{\phi}_{u}(y_u, y_v) \geq \theta^{\phi}_{u,v}(y^*_u, y^*_v), \ \forall y_u, \ y_v \in \mathcal{Y}. \tag{20}$$

Using (19) in $g(y)$ we obtain

$$g(y) \geq \sum_{u} \theta^{\phi}_{u}(y^*_u) = E^*_{\mathcal{G}'}. \tag{21}$$

Thus as $E_{\mathcal{G}'}(y^*) \leq g(y^*) \leq E_{\mathcal{G}'}(y^*)$, $g(y^*) = E_{\mathcal{G}'}(y^*)$, proving the `equal-minima` property.

The "only if" part We have to show that if $g(y)$ is a minorant of $E_{\mathcal{G}'}$, then ϕ is an optimal reparameterization, *i.e.* $D(\phi) = E_{\mathcal{G}'}(y^*) = g(y^*)$, where y^* is the optimal labelling for $E_{\mathcal{G}'}$.

Due to the minorant `equal-minima property`, we have

$$g(y^*) = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) + \sum_{uv\in\mathcal{E}'} \theta_{uv}^\phi(y_u^*, y_v^*) = E_{\mathcal{G}'}(y \mid \theta^\phi) \implies \sum_{uv\in\mathcal{E}'} \theta_{uv}^\phi(y_u^*, y_v^*) = 0; \tag{22}$$

As we assume $\theta_{uv}^\phi(s,t) \geq 0$, for all $s,t \in \mathcal{Y}^2$ and $uv \in \mathcal{E}'$, this would imply all terms $\theta_{uv}^\phi(y_u^*, y_v^*)$ are identically zero, *i.e.*

$$\theta_{uv}^\phi(y_u^*, y_v^*) = 0, \quad \forall uv \in \mathcal{E}'. \tag{23}$$

Our initial objective was to show $D(\phi) = g(y^*) = E(y^* \mid \theta^\phi)$. As we assume $\sum_{uv\in\mathcal{E}'} \min_{s,t} \theta_{uv}^\phi(s,t) = 0$, we just have to show

$$D(\phi) = \sum_{u\in\mathcal{V}'} \min_s \theta_u^\phi(s) = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) = g(y^*). \tag{24}$$

Following a proof by contradiction argument, we claim

$$D(\phi) = \sum_{u\in\mathcal{V}'} \min_s \theta_u^\phi(s) = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) = g(y^*) = E_{\mathcal{G}'}(y^* \mid \theta^\phi). \tag{25}$$

Assume the above statement is false and let D^* be the optimal dual.

Further, w.l.o.g. let's assume the $\min_s \theta_u^\phi(s) = y_u^*$ for all $u \in \mathcal{V}' \setminus k$ and $\min_s \theta_k^\phi(s) = y_k^+$. As strong duality holds, we have

$$D(\phi) = \sum_{u\in\mathcal{V}'} \min_s \theta_u^\phi(s) = \sum_{u\in\mathcal{V}'\setminus k} \theta_u^\phi(y_u^*) + \theta_k^\phi(y_k^+); \tag{26}$$

$$D^* = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) = E_{\mathcal{G}'}(y^* \mid \theta^\phi). \tag{27}$$

Thus by assumption $D(\phi) \geq D^*$,

$$D(\phi) \geq D^* \implies \sum_{u\in\mathcal{V}'\setminus k} \theta_u^\phi(y_u^*) + \theta_k^\phi(y_k^+) \geq \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u^*) \implies \theta_k^\phi(y_k^+) \geq \theta_k^\phi(y_k^*). \tag{28}$$

But $\theta_k^\phi(y_k^+) = \min_{y_k} \theta_k^\phi(y_k) \leq \theta_k^\phi(y_k^*)$, this is therefore a contradiction and $D(\phi) = D^* = g(y^\phi) = E(y^\phi \mid \theta^\phi)$. □

Theorem 2. Let $\mathcal{G}'$ be a tree and reparametrization ϕ be dual optimal on $\mathcal{G}'$. The function $g(y) = \sum_{u\in\mathcal{V}'} \theta_u^\phi(y_u)$ is a maximal minorant if and only if $\forall uv \in \mathcal{E}'$ and $\forall s,t \in \mathcal{Y}$:

$$\min_{s'\in\mathcal{Y}} \theta_{uv}^\phi(s',t) = \min_{t'\in\mathcal{Y}} \theta_{uv}^\phi(s,t') = 0\,. \tag{14}$$

Proof. **"Only if part"** .

For an optimal reparametrization ϕ, its corresponding tight minorant by Theorem 1 is $g(y) = \sum_u \theta_u^\phi(y_u)$. We need to prove the statement that minorant g is maximal only if the conditions in the theorem are fulfilled.

Recall that we are working with the constrained dual so that $\theta^\phi \geq 0$ component-wise. Assume for contradiction that one of the two zero minimum conditions is violated. Let it be the one with minimum over s'. Then $\exists uv \in \mathcal{E}'$ $\exists t$ such that $\lambda(t) := \min_{s'} \theta_{uv}(s',t) > 0$. We can then add $\lambda(t)$ to $\phi_{vu}(t)$. This will not destroy optimality of ϕ but will strictly increase $\theta_v^\phi(t)$, therefore leading to a strictly greater minorant, which contradicts maximality of g.

"If part" We need to show that if the conditions of the theorem are fulfilled then g is maximal.

Assume for contradiction that g is not maximal, *i.e.* there is a modular function $h(y)$ such that it is also a minorant for $E_{\mathcal{G}'}$ and it is strictly greater than g: $h(y) \geq g(y)$ for all y and $h(y') > g(y')$ for some y'.

The inequality $h(y) \geq g(y)$ for modular functions without constant terms is equivalent to component-wise inequalities:

$$h_u(y_u) \geq g_u(y_u), \ \forall u, \forall y_u. \tag{29}$$

From the inequality $h(y') > g(y')$ we conclude that there exists u and y'_u such that $h_u(y'_u) > g_u(y'_u)$. By the conditions of the theorem, and assuming a tree graph, a labeling y' can be constructed such that it takes label

y'_u in u and all costs $\theta^{\phi}_{u,v}(y'_u, y'_v)$ are zero. The construction starts from y'_u, finds labels in the neighbouring nodes such that edge costs with them is zero and proceed recurrently with the neighbours and their unassigned neighbouring nodes. For the labeling y' constructed in this way we have that

$$g(y') = \sum_u \theta^{\phi}_u(y'_u) = \sum_u \theta^{\phi}_u(y'_u) + \sum_{uv} \theta^{\phi}_{u,v}(y'_u, y'_v) = E_{\mathcal{G}'}(y'). \tag{30}$$

At the same time, $h(y') > g(y')$ and therefore $h(y') > \mathcal{E}_{\mathcal{G}'}(y')$, which contradicts that h is a minorant of $E_{\mathcal{G}'}$. □

B. ALGORITHMS DETAILS

B.1 Maximal Monotonic Chains

In this section we describe how we selected a collection of monotonic chains (`MMC`), on which `TRW-S` can run in its full efficiency and at the same time subgraph-based updates of `TBCA` and `HM` can be computed.

A chain is a subgraph of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that is completely defined by enumerating the sequence of nodes it contains, *i.e.* a chain $\mathcal{C}$ is denoted as $\mathcal{C} = (n_1, \dots, n_M), n_i \in \mathcal{V}$, with $(n_i, n_{i+1}) \in \mathcal{E}$ for $i = 1 : M-1$ denoting the edges it contains. Therefore, for every pair of consecutive nodes (n_i, n_{i+1}) there must also exist a corresponding edge in $\mathcal{E}$ for a chain to be a subgraph of $\mathcal{G}$.

Let there be a partial order defined on the nodes $\mathcal{V}$ such for each edge $uv \in \mathcal{E}$ the nodes are comparable: either $u > v$ or $v < u$. This can be always completed to a total order as was used for simplicity in Kolmogorov (2006). A chain $\mathcal{C}$ is said to be *monotonic* if $n_i < n_{i+1}$ holds for its nodes. A chain $\mathcal{C}$ is *maximal monotonic* if it is monotonic and not a a proper subgraph of some other monotonic chain.

For a given ordering, we select a collection of edge disjoint monotonic chains covering the graph by greedily finding and removing from the edge set maximal monotonic chains. Finding and removing one chain is specified by Algorithm 3. The algorithm works on the graph adjacency list representation. Let Ad be the adjacency list corresponding to the directed version of directed the graph $\mathcal{G}$: $Ad(i)$ contains all neighbours of node i in G that are greater than i, *i.e.* $\forall j \in Ad(i), j > i$. The operation $Ad(i).remove(j)$ removes element j from the list $Ad(i)$. The algorithm is executed until all Ad lists are empty (all edges have been covered).

Algorithm 3 Compute Maximal Monotonic Chain

function $(\mathcal{C}, Ad)$=COMPUTEMMC(Ad) ▷ Ad is the adjacency list of $\mathcal{G}$ as defined above.
  $\mathcal{C} = \emptyset$, `tail` $= \emptyset$, $done = false$ ▷ $\mathcal{C}$ is initially empty., `tail` is the last node added to the chain.
  Find the smallest in the order i such that $Ad(i)$ is not empty.
  $\mathcal{C}.add(i)$, `tail` $= i$. ▷ Add node i to $\mathcal{C}$. Update `tail`.
  while !$done$ **do**
    Find j in Ad(`tail`) such that $j >$ `tail`.
    if j is found **then**
      $\mathcal{C}.add(j)$, Ad(`tail`)$.remove(j)$, `tail` $= j$ ▷ The node j is added to $\mathcal{C}$, removed from Ad(`tail`). `tail` is updated.
    else if j is not found **then**
      $done = true$ ▷ The loop exit condition is satisfied.

The result of the algorithm is a collection of chains that are monotonic w.r.t. to the ordering. TRWS running on the respective ordering of nodes as introduced in Sec. 3.3 can be viewed also as optimizing the dual decomposition with monotonic chains Kolmogorov (2006). It can be shown that the number $\max(N_{\text{in}}(u), N_{\text{out}}(u)$ used to calculate weights in TRWS is exactly the number of different chains containing node u for any collection of monotonic chains found as above. Hence such a collection natively represent subproblems associated with TRWS.

B.2 Message Passing in Spanning Trees

The hierarchical minorant for chains involves passing messages from the ends of the chain to the central nodes, as shown in 2. For trees, the process is similar. Messages are passed from the leaf nodes to the central nodes. The centroid of a tree of size n is the node whose removal results in subtrees of size $\leq \lfloor \frac{n}{2} \rfloor$. The central nodes of a tree are defined as nodes connected by an edge whose removal gives trees that are similar in length. One of the

central nodes is always the tree-centroid. The other inode s selected keeping in mind minimum deviation between the different sub-trees that arise from the removal of this node. As the hierarchical minorant is recursive, the recursion is repeated with a subtree.

B.3 Generation of Spanning Trees in TBCA

For the static strategy, we compute a sequence of minimum weight spanning trees with the weights being the number of times an edge has already been included in a spanning tree. This weighing scheme ensures that un-sampled edges are prioritized in building spanning trees. The sampling is stopped when all the edges are covered. In the experiments (below) we observed that with the block update strategy that we chose, dynamic updates were not advantageous any more and performed slower overall.

Algorithm 4 Compute Strictly Shortest Path

function $(\mathcal{C})$=COMPUTESSP$(\mathcal{G} = (\mathcal{V}, \mathcal{E})$,src) ▷ *src* is the source node from which to grow the shortest path.
  Create Vertex Set $\mathcal{Q}$ from graph $\mathcal{G}$
  for Each Vertex v in $\mathcal{G}$ **do**
    $dist[v] := \infty$ ▷ Set distance of all vertices to ∞
    $prev[v] := UNDEFINED$ ▷ Initialize all previous nodes to default value.
  $dist[src] = 0$ ▷ Distance from the source node to the source node is 0
  while $\mathcal{Q}$ is not empty **do**
    $u :=$ vertex in $\mathcal{Q}$ with min $dist[u]$ ▷ u is assigned vertex in $\mathcal{Q}$ with minimum value in $dist[\,]$
    for each neighbor v of u **do** ▷ Only v that are still in $\mathcal{Q}$
      $alt := dist[u] + 1$ ▷ alt is $dist[u] + length(u, v)$, which equals 1
      if $alt < dist[v]$ **then** ▷ If $dist[v]$ is greater than alt update distance
        $dist[v] := alt$
        $prev[v] := u$
      else if $alt == dist[v]$ **then** ▷ Condition for strictness of shortest path is violated
        `break`
  Construct chain $\mathcal{C}$ from $prev[\,]$

C. DETAILED EXPERIMENTAL RESULTS

We show in Fig. C.1 results per individual application, with performance in both messages and time. Since in each application, there are still multiple instance, we apply the same normalization and averaging procedures as in the main paper.

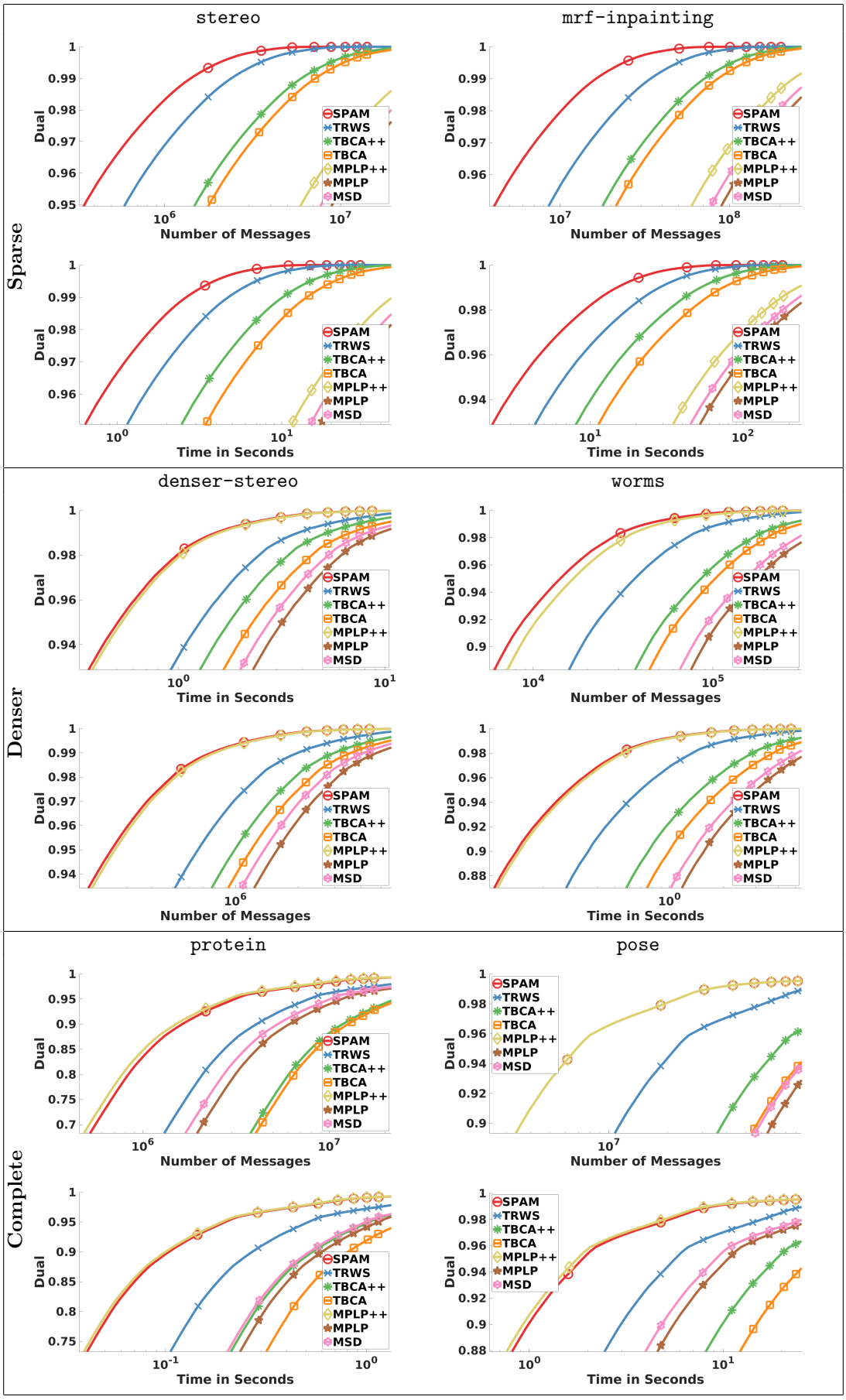

Figure C.1: The averaged plots for application-specific datasets: messages and time.

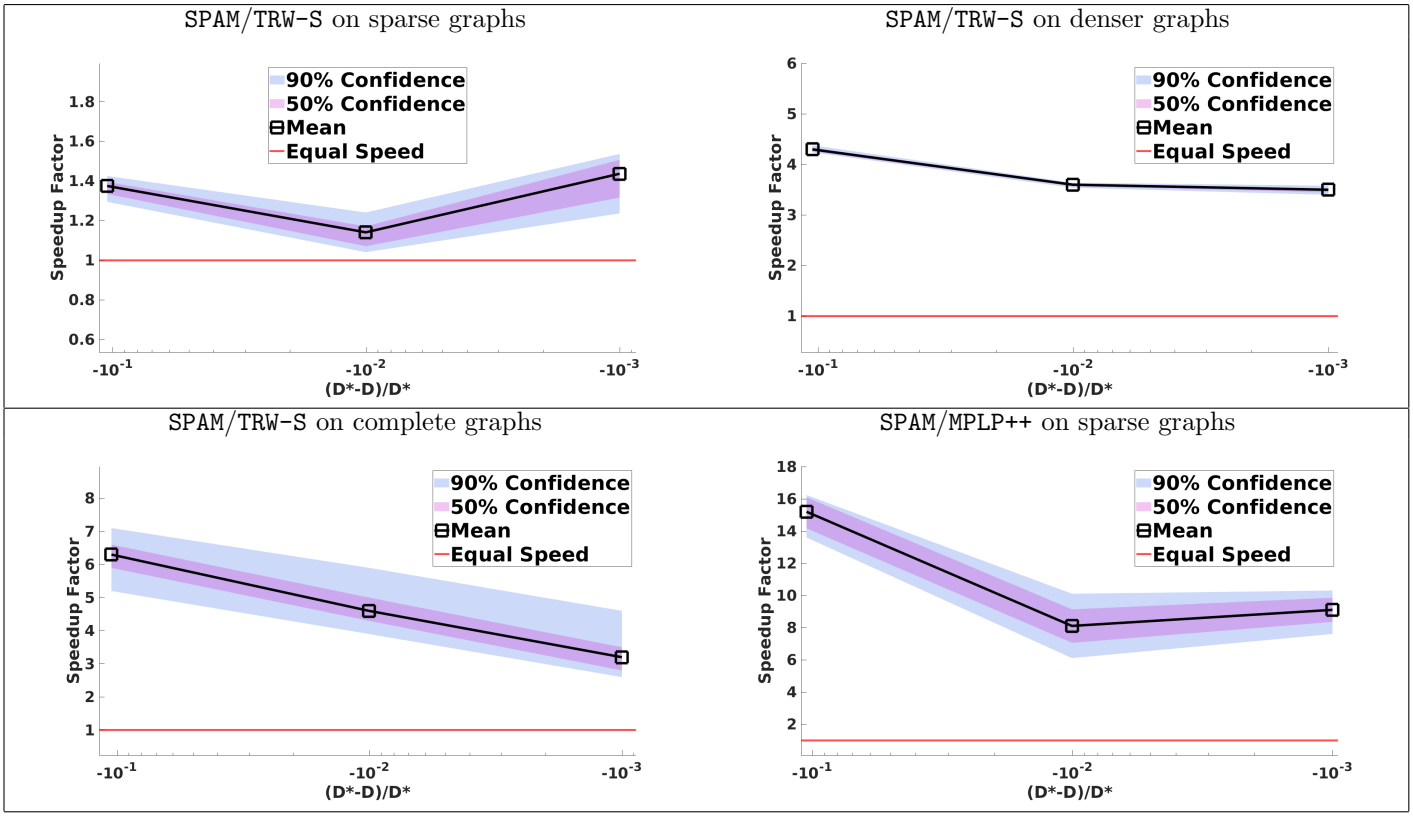

Figure C.2: Speed-up factors of SPAM w.r.t. TRW-S and MPLP++ with confidence intervals for the different datasets. The x-axis shows the normalized dual value and the y-axis the speed-up to achieve the same dual. The statistics are computed over all instances in a dataset. We show asymmetric confidence intervals with the equal percentage around the mean.