Additional Experiments for Paper 2758

March 20, 2023

In this short report we provide additional experiment results for online LP.

1 Running time of online algorithm

In this section, we test the running time of online algorithm in practice.

Since running time of our algorithm only depends the the number of nonzeros in the LP coefficient matrices, we generate MKP data in the same way as in section 5.1 of our main paper.

The following table summarizes the CPU time (in seconds) when $K = 1$. Since $K > 1$ exactly repeats the algorithm K times, it suffices to multiply the running time by K to estimate the running time of our explicit update method. Results are obtained on a Mac Mini with M1 apple silicon. We also report solution time of `Gurobi v9.5.2` excluding crossover time (if barrier solver solves the problem).

Size (m, n)	Sparsity	Online	Gurobi	Size (m, n)	Sparsity	Online	Gurobi
$(8, 10^5)$	1.0	0.009	0.56	$(128, 10^6)$	0.100	0.28	79.56
	0.1	$< 10^{-3}$	0.44		0.050	0.20	86.75
$(16, 10^5)$	1.0	0.021	0.92	$(1024, 10^6)$	0.050	0.49	> 100
	0.1	0.002	0.25		0.010	0.23	> 100
$(32, 10^6)$	0.5	0.022	20.42	$(4096, 10^7)$	0.001	2.20	> 100
	0.1	0.005	10.58		0.0005	1.53	> 100

Table 1: Running time of our methods

From the above results we can see that our method scales linearly with nnz($\mathbf{A}$) and it takes less than one second to complete for sparse problem with one million variables.

Remark 1 *We note that Gurobi solves the problem to very high accuracy and is dependent on the numerical condition of problem data. Thus our method is not directly comparable with Gurobi, and*

we record running time only for reference.

2 Additional experiment

In this section, we carry out additional experiment to show that our methods work in practice when the assumptions do not necessarily hold. Specifically we consider the MKP problem with $\mathbf{b} = \mathbf{1}$.

$$\begin{aligned} \max_{\mathbf{x}} \quad & \langle \mathbf{c}, \mathbf{x} \rangle \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{1} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \end{aligned}$$

When n is large, **A1** becomes asymptotically violated since $\underline{d} = \frac{1}{n}$ becomes closer to one.

Testing configuration and setup

We configure the algorithms as follows.

1. **Dataset**. We take $(m, n) \in \{(5, 100), (8, 1000), (16, 2000), (32, 4000)\}, \sigma = 1$.
2. **Initial point**. We let online algorithms start from $\mathbf{0}$.
3. **Feasibility**. We force the algorithm to respect constraint violation.
4. **Duplication**. We allow $K \in \{1, 2, 4, 8, 16, 32, 64, 128\}$
5. **Stepsize**. We take $\gamma = (Kmn)^{-1/2}$.

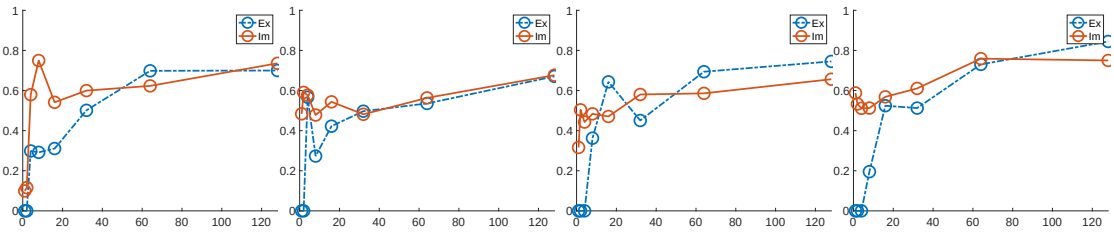

Figure 1: From left to right: $(m, n) \in \{(5, 100), (8, 1000), (16, 2000), (32, 4000)\}$ The axis represents K parameter ranging from 1 to 128. y-axis, relative optimality gap

We see that as suggested by our theory, as $\underline{d} = \frac{1}{n}$ gets smaller, online algorithms perform poorly when $K = 1$, but when we increase K, both implicit and explicit update gradually retrieve optimality gap.