# 1008 Meeting

Cheng-Han Huang

# Global Setting

- Running on AMD EYYC 9654 / 9684X with A100, thread setting = 192
- $50$ seeds on $ER(500,0.5)$ for testing with fixed number of initialization ($10$ by default), and $10$ seeds on $ER(n, 0.5)$ for testing with fixed time.
- Maximum iteration set to $15000$ for fixed initialization testing and $150$ for MGD fix time testing when $n \leq 2000$ and $300$ otherwise. For Projected Newton and L-BFGS-B, we use $15000$ for all experiments.

# Methods

- CP-SAT: Widely used as baselines in combinatorial optimization studies. Take the integer programming form of the maximum independent set problem and solve.

- MGD (pCQO) / Projected Newton / Scipy's L-BFGS-B: Solve a relaxed optimization problem of maximum independent set that is continuous and differentiable with gradient based methods. Objective function are taken from pCQO paper with the clique informed penalty term.

# Methods

- Newton-Ours (single threaded): Projected Newton with a ‘free-set’ masking to determine points that can move legally. Only these points are computed in the algorithm. Maintain a sub-Hessian matrix for computation efficiency and recomputes only when the set of legal points or the damping matrix change. Armijo update with initial steps and backtracking scalar finetuned by grid search.

- L-BFGS-Ours (single threaded): Handcrafted L-BFGS also with the free set. Armijo update with initial steps and backtracking scalar finetuned by grid search.

| Graph, Time | MGD | CP-SAT | Projected Newton | L-BFGS-B | Newton-Ours | L-BFGS-Ours-10 | L-BFGS-Ours-20 | Yongliang's -5 (with / without clique penalty) | Yongliang's -10 (with / without clique penalty) |
|---|---|---|---|---|---|---|---|---|---|
| ER(500,0.5), 5 sec | 12.1 | 12.1 | 11.5 | 10.6 | 11.9 | 12.1 | 12.0 | 11.3 / 11.3 | 12.1 / 12.0 |
| ER(1000,0.5), 5 sec | 12.4 | 12.0 | 11.6 | 11.2 | 12.8 | 11.8 | 11.8 | 12.2 / 12.6 | 13.0 / 13.1 |
| ER(1000,0.5), 10 sec | 12.7 | 12.5 | 11.9 | 11.8 | 13.0 | 12.2 | 12.2 | 12.7 /12.6 | 13.3 / 13.3 |
| ER(2000,0.5), 10 sec | 12.9 | 0 | 7.2 | 12.3 | 13.2 | 12.5 | 12.5 | 13.8 / 14.0 | 13.5 / 13.9 |
| ER(2000,0.5), 20 sec | 13.2 | 11.2 | 10.7 | 12.4 | 13.4 | 12.7 | 12.7 | 14.1 / 14.1 | 13.7 /13.9 |
| ER(3000,0.5), 30 sec | 13.3 | 0 | 7.9 | 0 | 13.9 | 13.1 | 13.4 | 14.1 / 14.2 | 14.6 / 15.0 |
| ER(5000,0.5), 50 sec | 13.3 | 0 | 5.1 | 0 | 14.4 | 13.8 | 13.9 | 14.7 /15.0 | 15.1 / 15.3 |

# Gorubi Doesn’t Perform

- Compared to CP-SAT, Gorubi performs worse for the task of solving integer programming here unless the given time is too little for CP-SAT to produce even an initial sub-optimal solution.

| Graph, Time | CP-SAT | Gurobi |
|---|---|---|
| ER(500,0.5), 5 sec | 12 | 8.7 |
| ER(1000,0.5), 5 sec | 0 | 9.5 |
| ER(1000,0.5), 10 sec | 11.4 | 9.5 |

This comparison is run on my local environment.

# Free-Mask for Our Newton / L-BFGS

- Determine variables that are able to make meaningful moves.

Build a set $F$, for each coordinate $i$:

1. Interior: If $0 < x_i < 1$, *then* $x_i \in F$
2. At boundaries: $x_i$ at boundaries and the gradient is moving inward, *then* $x_i \in F$
3. Otherwise, $x_i \notin F$.

# Free-Mask for Our Newton

- In normal Newton, we compute $(H + \lambda I)d = -g$.
- Here, we consider $S_F \in \mathbb{R}^{n \times |F|}$ whose columns are the standard basis vectors $e_i$ for $i \in F$.
- $d = S_F d_F$ where $d_F \in \mathbb{R}^{|F|}$ is the step on free variables.
- Our method do $(H_{FF} + \lambda I)d_F = S_F^T (H + \lambda I) S_F d_F = -S_F^T g = -g_F$

# Free-Mask for Our Newton

Now, caching the Cholesky of $H_{FF} + \lambda I$ for each pair of $(F, \lambda)$.

Define $M_F(\lambda) = H_{FF} + \lambda I$, compute$M_F(\lambda) = LL^T$ once, which unlock the ability to compute$d_F = L^{-T}L^{-1}(-g_F)$ for steps with the same pair of $(F, \lambda)$.

Often, $|F| \ll n$, so even if we must refactor often, it's still cheaper than doing computation on the full space.

# Free-Mask for Our L-BFGS

- Similarly, with free mask we are doing the problem on the active coordinates.

- Avoid meaningless or harmful computation; projections break the secant equation at boundaries, and we don’t need to compute steps that we are going to zero out anyway.

# Scipy's L-BFGS-B and Our L-BFGS

- Comparison by solution size: (L-BFGS-B indicates Scipy's solver)

| | L-BFGS-B | Ours |
|---|---|---|
| Average solution size | 11.28 | 10.80 |

- Comparison by iteration number/time:

| | L-BFGS-B | Ours |
|---|---|---|
| Average iteration number | 167.1 | 23.1 |
| Average time | 2.2564 | 0.0613 |

# Projected Newton and Our Newton

- Comparison by solution size: (L-BFGS-B indicates Scipy's solver)

| | Projected Newton | Ours |
|---|---|---|
| Average solution size | 11.08 | 11.1 |

- Comparison by iteration number/time:

| | Projected Newton | Ours |
|---|---|---|
| Average iteration number | 76.91 | 20.69 |
| Average time | 0.1665 | 0.0155 |

# Next Meeting

- Determining how to choose a good max iter
- Determining or controlling memory length for L-BFGS if using
- Threading for our methods
- ReduMIS / iSCO baseline comparison
- Try on Max Cut / TSP

# Appendix

# Outlier Case

- We have one run taking much more steps

```
init  0 → MIS size   9, iters    37, time  0.0278s
init  1 → MIS size   8, iters    19, time  0.0152s
init  2 → MIS size   9, iters    29, time  0.0216s
init  3 → MIS size  10, iters    18, time  0.0123s
init  4 → MIS size  10, iters    31, time  0.0220s
init  5 → MIS size  10, iters    14, time  0.0099s
init  6 → MIS size  10, iters    21, time  0.0140s
init  7 → MIS size  10, iters    15, time  0.0101s
init  8 → MIS size  10, iters    16, time  0.0109s
init  9 → no MIS flagged (iters 15000, time 23.4302s)
```

# Single Thread

- The same piece of Newton / L-BFGS code.

- Average time with same setting

| Environment / Method | Newton | Scipy's L-BFGS-B |
|---|---|---|
| My Local Env | 0.1350 | 0.7411 |
| AMD24 | 0.1665 | 2.2564 |