From an inner point to a corner point: Smart Crossover

Zikai Xiong[1]

[1]MIT Operations Research Center
Joint work with Dongdong Ge, Chengwenjian Wang, and Yinyu Ye
Slides made by Chengwenjian Wang

October 15, 2022

Table of Contents

Table of Contents

LP Formulation

Consider a general LP problem:

$$\begin{aligned} \min_x \ & c^\top x \\ \text{s.t.}\ & Ax = b, \\ & x \geq 0 \end{aligned} \tag{1}$$

which is a powerful framework for describing and solving optimization problems.

- The set of applications of linear programming is literally too long to list;
- Everything from production scheduling to web advertising optimization to clothing manufacturing;
- LP touches nearly every commercial industry in some way.

Classic Algorithm: Simplex Methods

- The first algorithm solving LP, proposed by *George Dantzig* in 1947;
- Lead to an **exact(vertex) solution** each iteration;
- **Exponential convergence rate**, which makes it struggling when solving super large-scale problems.

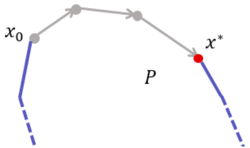


Figure: Geometric diagram of Simplex Method: find an exact(vertex) solution x^* with potentially exponentially many moves.

Classic Algorithm: Interior-Point Methods

- **polynomial-time complexity**;
- **Faster** than simplex for solving LP problems from scratch;
- Reach to an **interior-point** solution, unless the problem has a unique optimal solution.

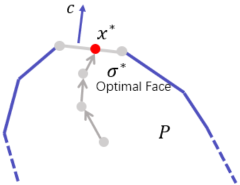


Figure: Geometric diagram of interior-point methods: quickly find a solution in the relative interior of the optimal face.

Prevalent First-Order Methods

- Traditional interior point method struggles for many **huge scale LP** problems due to high per iteration cost;
- Many successful first-order algorithms, such as
  - an ADMM based Interior Point Method (ABIP) (Lin, Ma, et al. 2020),
  - a primal-dual majorization-minimization method (Liu, Dai, and Huang 2022), etc.
- Strong methods for LP with special structure, especially network flow structure, e.g.
  - optimal transport (Cuturi 2013; Lin, Ho, and Jordan 2019),
  - Wasserstein barycenter (Benamou et al. 2015; Ge et al. 2019).
- **Low accurate** solutions and lack of dual information.
  (Thus the LP and mixer integer programming (MIP) solvers still cannot benefit from these emerging first-order methods.)

Significance for Basic(Vertex/Corner) Solutions

Interior-point methods $\rightarrow$ solutions in the interior of optimal face
First-order method $\rightarrow$ approximated sub-optimal solutions

There are many cases that a(n) basic (exact/vertex) solution could be **more valuable** than an interior-point (approximated) solution:

- **More accurate** than an interior-point solution,
- a basic solution can be used to **warm-start the simplex algorithm** in case of reoptimization,
- **More sparse**, i.e. more variables are fixed to zero. Particularly appealing when solving continuous relaxations of **mixed integer problems**.

Crossover

? How could we benefit from the speed of interior-point methods or first-order methods, and also obtain high-quality basic (vertex) solutions?

A crossover algorithm is a bridge from inner points to corner points.

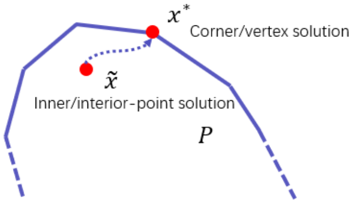
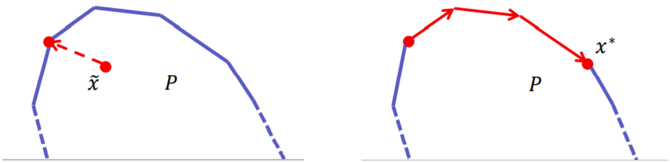


Figure: Crossover algorithm — a "jump" from an interior-point solution to a vertex solution

Crossover Methods — Diagrams



(a) Basis Identification (Primal/Dual Push)

(b) Reoptimization (Clean-up)

Figure: Crossover algorithm's process diagram

Crossover Research Review

- There are few papers about crossover research. It is an undisclosed technology developed by commercial solvers respectively.
- Previous research such as Megiddo (1991), Mehrotra and Ye (1993), Andersen and Ye (1996), and Andersen (1999), **only consider crossover from an optimal primal-dual pair, which is not attainable by first-order methods**;
- One reason is that crossover algorithms are **hard to do theoretical convergence analysis**. Since the crossover phase starts from an interior-point solution which we have no prior information. It is almost impossible to prove the convergence or show the convergence rate of a crossover algorithm.

Practical Results

For current crossover algorithms, quite often the crossover computation time is **significantly longer** than the interior-point method computation time. We list some of these problems.

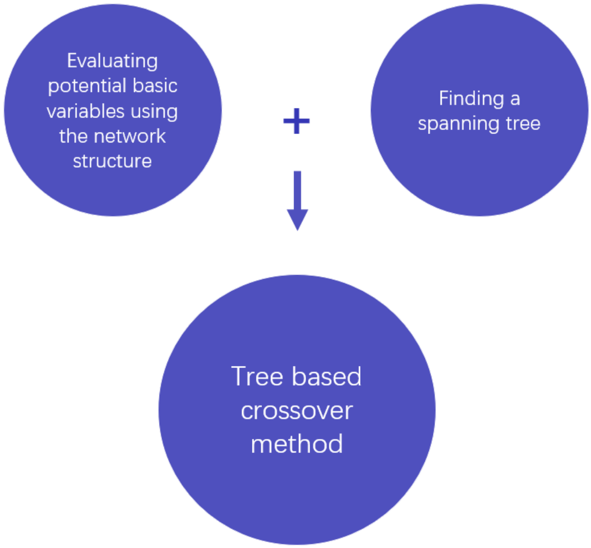
- Large-scale optimal transport problems and minimal cost flow problems could be solved faster by the barrier algorithm than the simplex method. But their crossover time is quite long and affect the solving efficiency;
- Benchmark of barrier LP solvers. Many problems here have long crossover time. Some typical problems like datt256 and graph40-40. The crossover run-time is hundreds of times over the barrier run-time.

Our Contributions

- For large scale LP with network structure, we propose
  - a criterion to evaluate the possibility of each variable being in the optimal basis, and a column generation based basis identification phase;
  - a spanning tree structure based basis identification method based on the spanning tree characteristics of basic solutions.
  - Speed-up crossover over commercial solvers and create fast algorithm combining with first-order methods, such as Sinkhorn algorithm.
- For general LP problems, we develope
  - a perturbation crossover to alleviate difficulties when the LP has a large optimal face;
  - This powerful technique has been used on the rapid-growing commercial optimizer COPT, leading to a breakthrough from its version *1.5* to *1.6*.

Table of Contents

The Tree Based Crossover Method



High-level Idea of Evaluating Potential Basic Variables

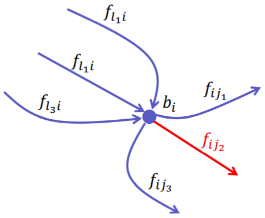


Figure: A flow f in G at node i, from the given interior-point solution f.

Measure the importance of each arc in a flow:

$$\frac{\text{flow on this arc}}{\max\{\text{the total in-flow, out-flow for the node}\}}$$

Spanning Tree Based Basis Identification

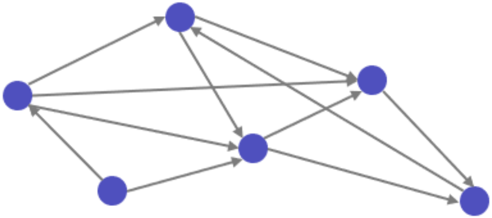


Figure: An interior-point solution on a network flow.

Spanning Tree Based Basis Identification

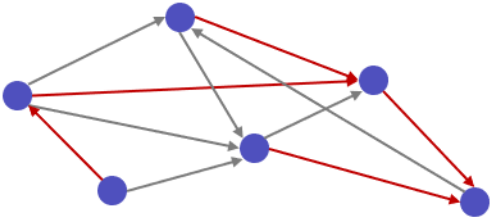


Figure: Basic solution for a network LP problem is a tree solution.

Experiment: Optimal Transport (MNIST dataset)

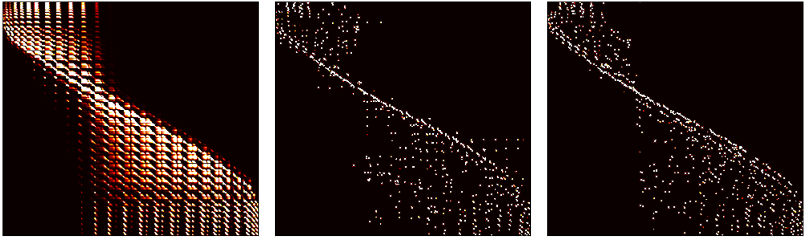


Figure: Transport plan of a randomly generated optimal transport problem from MNIST dataset. The left, the middle, and the right are the initial interior point solution, the tree solution from Tree-based basis identification, and the final-solved optimal solution respectively.

Experiment: Optimal Transport (MNIST dataset)

Table: Crossover procedure comparison with **high precision** interior-point solution.

	scale	gurBarr	gurCross	CNET	TNET
1	1	0.41 s	0.71 s	**0.28 s**	0.32 s
2	1	1.79 s	0.64 s	0.60 s	**0.28 s**
3	1	0.90 s	0.27 s	**0.20 s**	0.21 s
4	2	1.81 s	**0.25 s**	0.64 s	0.71 s
5	2	3.05 s	**0.60 s**	0.86 s	0.82 s
6	2	1.67 s	**0.34 s**	1.05 s	0.96 s
7	3	17.76 s	**1.74 s**	2.44 s	1.85 s
8	3	11.09 s	**0.50 s**	3.45 s	2.02 s
9	3	7.10 s	**0.48 s**	1.53 s	1.58 s
10	4	7.40 s	**0.12 s**	1.58 s	1.22 s
11	4	24.86 s	32.06 s	8.95 s	**3.21 s**
12	4	29.86 s	9.09 s	4.05 s	**3.46 s**
13	5	233.28 s	215.84 s	43.82 s	**17.65 s**
14	5	43.02 s	105.77 s	48.68 s	**8.76 s**
15	5	184.96 s	246.66 s	24.38 s	**15.85 s**

Experiment: Optimal Transport (MNIST dataset)

Table: Crossover procedure comparison with **low precision** interior-point solution.

	scale	gurBarr	gurCross	CNET	TNET
1	1	0.88 s	**0.33 s**	1.25 s	1.26 s
2	1	0.39 s	0.26 s	**0.17 s**	0.18 s
3	1	0.45 s	0.20 s	0.20 s	**0.19 s**
4	2	1.17 s	**0.24 s**	0.49 s	0.54 s
5	2	0.93 s	**0.31 s**	0.50 s	0.55 s
6	2	1.22 s	0.57 s	0.86 s	**0.56 s**
7	3	14.98 s	6.16 s	6.54 s	**5.04 s**
8	3	6.76 s	4.06 s	**1.11 s**	1.25 s
9	3	6.23 s	5.80 s	1.42 s	**1.25 s**
10	4	40.28 s	51.75 s	**4.68 s**	6.47 s
11	4	15.58 s	19.07 s	5.61 s	**5.37 s**
12	4	18.51 s	67.67 s	10.25 s	**4.92 s**
13	5	114.45 s	231.84 s	16.01 s	**14.16 s**
14	5	110.92 s	282.51 s	**8.58 s**	11.97 s
15	5	50.52 s	142.90 s	12.68 s	**8.01 s**

Experiment: Optimal Transport (MNIST dataset)

Table: Total run-time comparison among Simplex, Barrier, network Simplex, and Sinkhorn plus our crossover methods.

	scale	gurSimplex	gurBarrier	cplNetSplx	Skh+CNET	Skh+TNET
1	1	0.19 s	1.10 s	**0.05 s**	0.17 s	0.26 s
2	1	0.35 s	1.17 s	**0.05 s**	0.06 s	0.07 s
3	1	0.33 s	1.28 s	0.06 s	**0.05 s**	0.07 s
4	2	4.20 s	2.28 s	0.23 s	**0.21 s**	0.36 s
5	2	9.29 s	3.12 s	0.36 s	**0.25 s**	0.44 s
6	2	0.54 s	1.29 s	0.09 s	**0.08 s**	0.13 s
7	3	32.04 s	6.17 s	0.67 s	**0.44 s**	0.81 s
8	3	141.20 s	9.10 s	1.13 s	**0.64 s**	1.03 s
9	3	247.53 s	16.28 s	1.98 s	**1.32 s**	2.46 s
10	4	98.81 s	11.25 s	1.56 s	**1.23 s**	2.02 s
11	4	997.06 s	51.23 s	5.05 s	**2.39 s**	4.43 s
12	4	t[1]	123.42 s	4.89 s	**3.19 s**	5.33 s
13	5	t	280.12 s	10.50 s	**7.73 s**	15.14 s
14	5	t	177.75 s	6.53 s	**5.26 s**	11.74 s
15	5	t	245.74 s	14.16 s	**5.21 s**	8.58 s
16	6	t	862.93 s	48.20 s	**11.03 s**	30.59 s
17	6	t	823.22 s	56.16 s	**21.87 s**	37.57 s
18	6	t	536.69 s	42.94 s	**12.20 s**	27.46 s

[1] Time limit exceeded (over 1000 seconds);

Experiment: Minimum Cost Flow

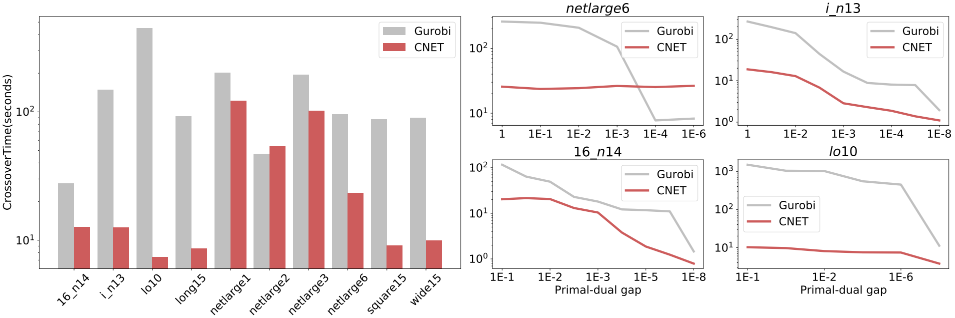


Figure: Computation time of crossover on the large network-LP benchmark problems.

Experiment: Minimum Cost Flow

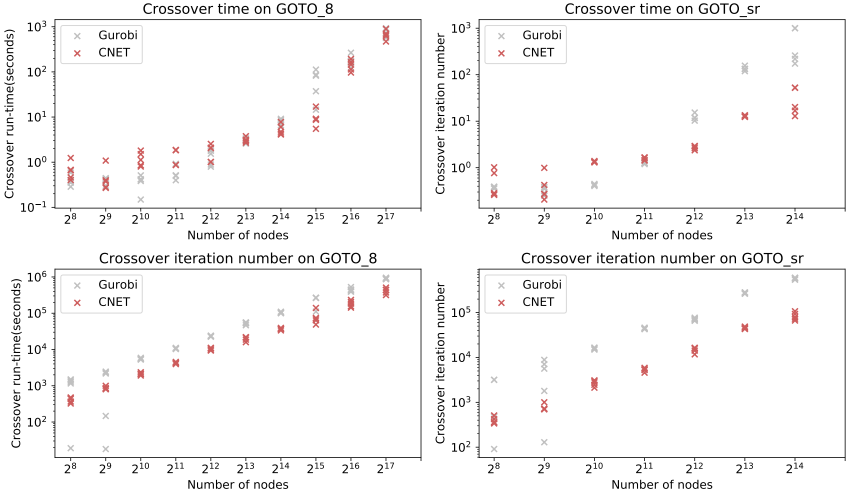


Figure: Computation time and iteration number of crossover on GOTO_8 and GOTO_sr.

Perturbation Crossover: a High Viewpoint

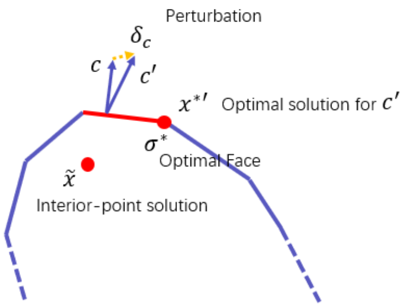


Figure: For a certain type of problems, there are infinite optimal solution gathered on an "optimal face". In this case, a perturbation on c will let the optimal face degenerate to a single point. This will largely reduce the difficulty of crossover.

Experiment: Perturbation Crossover on LP Benchmark

Table: Test perturbation crossover on **Mosek** on the barrier LP benchmark problems with long crossover time.

	problem	mskBarr	Original	Perturbed
1	datt256	3.61 s	349.36 s	**10.08 s**
2	ns1688926	3.53 s	**87.55 s**	90.86 s
3	stat96v1	9.20 s	104.72 s	**65.99 s**
4	graph40-40	18.20 s	158.17 s	**37.11 s**
5	savsched1	17.19 s	113.30 s	**48.73 s**
6	self	1.02 s	5.80 s	**1.44 s**

Experiment: Perturbation Crossover on LP Benchmark

Table: Test perturbation crossover on **Cplex** on the barrier LP benchmark problems with long crossover time.

	problem	cplBarr	Original	Perturbed
1	graph40-40	0.69 s	92.89 s	**45.44 s**
2	datt256	3.49 s	284.44 s	**23.70 s**
3	nug08-3rd	1.25 s	88.75 s	**64.77 s**
4	cont11	7.24 s	**221.69 s**	279.61 s
5	cont1	2.50 s	70.41 s	**64.23 s**
6	shs1023	15.42 s	365.36 s	**298.28 s**
7	savsched1	7.64 s	108.45 s	**25.03 s**
8	chrom1024-7	0.20 s	2.63 s	**2.50 s**
9	neos3	1.45 s	**15.48 s**	73.80 s
10	fhnw-bin0	1.53 s	**12.84 s**	20.75 s
11	self	0.99 s	5.87 s	**3.38 s**
12	qap15	0.39 s	**2.06 s**	3.03 s

Thank you!