Simplex vs. Interior Point Methods for LP

Neither dominates. The honest answer is that it depends on problem size, structure, and what you need out of the solve.

1. Theory vs. practice

Interior point methods (IPMs) have polynomial complexity — roughly O(√n) iterations for a primal-dual path-following method, each iteration costing a Newton solve. Simplex has an exponential worst case (Klee–Minty), yet in practice takes a number of iterations that grows roughly linearly in the number of constraints. So the complexity theory favours IPM but badly mispredicts typical behaviour. The theoretical bound also does not drive real IPM performance: implementations converge in 20–60 iterations almost regardless of size, far better than the bound suggests.

2. Where each wins

Simplex tends to win on:

• Small and medium problems, where its cheap iterations beat one expensive Newton step.

• Problems with special structure it exploits well — network flow, staircase, highly sparse but poorly-factorizable constraint matrices.

• Anything needing warm starts. This is the decisive one: in branch-and-bound for MILP each node reoptimizes after a bound change and dual simplex restarts in a handful of pivots, whereas IPM essentially cannot warm start. That is why simplex is what MILP solvers use at nodes.

• When you want a basic (vertex) solution or clean sensitivity / dual information.

IPM tends to win on:

• Large problems — hundreds of thousands to millions of variables and constraints — where simplex iteration counts grow and IPM's fixed ~30 iterations pay off.

• Dense or highly degenerate problems. Degeneracy hurts simplex badly and does not bother IPM at all.

• Problems where the normal equations / KKT system factorizes well (good sparsity, low fill-in).

• Parallel hardware. The linear algebra in an IPM iteration parallelizes; simplex pivoting is inherently sequential.

3. Where IPM struggles

If the KKT system has dense columns or bad fill-in, one factorization can be catastrophically expensive, and a moderate-size problem that simplex solves in seconds may be intractable for IPM. Also, IPM converges to the analytic center of the optimal face, not a vertex; if you need a basic solution you must run a crossover step, which is itself simplex-like work and can take a large fraction of the total time.

4. What solvers actually do

Commercial solvers ship all three (primal simplex, dual simplex, barrier) and default to a concurrent solve: run dual simplex and barrier in parallel on separate threads and take whichever finishes first. That is essentially an admission that the question cannot be answered a priori. Dual simplex is the usual default for single-method LP because it handles the reoptimization patterns that show up most often.

	Simplex	**Interior point (barrier)**
Worst-case complexity	Exponential (Klee–Minty)	Polynomial, ~O(√n) iterations
Typical iterations	Grows ~linearly with #constraints	20–60, nearly size-independent
Cost per iteration	Cheap pivot + basis update	Expensive Newton / KKT factorization
Warm start	Excellent (dual simplex)	Essentially none
Degeneracy	Hurts badly (stalling, cycling)	Largely unaffected
Solution returned	Basic / vertex solution	Analytic center; crossover for a vertex
Parallelism	Poor — sequential pivoting	Good — parallel linear algebra
Inside MILP nodes	Standard choice	Rarely used

5. Rule of thumb

> Under roughly 10^4–10^5 constraints, or inside a MILP, use dual simplex. Above that, on a one-shot LP with good sparsity, try barrier first — and turn crossover off if you do not need a vertex solution. If you have the cores, just run both concurrently.