Cycling in the Simplex Method

Cycling happens when degenerate pivots — a basic variable sitting at value 0, so the step size is 0 — let the algorithm revisit the same basis without improving the objective. The objective never changes, the basis sequence repeats, and the method never terminates.

1. Anti-cycling methods

Bland's rule (smallest-subscript rule). Among all columns with negative reduced cost, enter the one with the smallest index; among rows tied in the minimum ratio test, let the basic variable of smallest index leave. This provably prevents cycling because the sequence of bases can never repeat. Cost: it often converges slowly, so it is typically used as a fallback — switched on only after a run of degenerate pivots is detected.

Lexicographic (Charnes) perturbation rule. Break ties in the ratio test lexicographically: instead of comparing only the ratios b_i/a_{ik}, compare the whole rows of $[b \mid B^{-1}]$ divided by a_{ik} and pick the lexicographically smallest. Equivalently, replace b_i by $b_i + \varepsilon^i$ for a symbolic infinitesimal ε, which makes every basic solution nondegenerate. Also provably finite, and usually much less damaging to speed than Bland's rule.

Random tie-breaking / random perturbation. Choose randomly among tied rows, or numerically perturb the right-hand side by tiny random amounts, solve, then restore. Cycling then has probability essentially zero. Not a proof of termination, but effective in practice.

Practical tricks used in real solvers.

- • *Harris ratio test with expanding tolerance*: allow slightly infeasible basic variables within a tolerance, giving more freedom in choosing the leaving variable and fewer degenerate steps.
- • *Perturbation on the fly*: CPLEX, Gurobi and HiGHS detect stalling, automatically perturb bounds, finish, then remove the perturbation and clean up.
- • *Switching pivot rules or restarting* from a different basis when stalling is detected.

> Cycling is rare in practice; degeneracy causing *stalling* — many zero-length steps without a true cycle — is the more common practical problem. That is why solvers favour perturbation and Harris-type ratio tests over Bland's rule.

2. Bland's rule vs. lexicographic: which is more efficient?

Short answer: the lexicographic rule is generally more efficient in practice; Bland's rule is cheaper to implement but tends to make the algorithm crawl.

Why Bland's rule is slow. It forces the entering variable to be the smallest-index column with negative reduced cost, throwing away the information the reduced costs carry. Dantzig's rule (most negative reduced cost) or steepest-edge pricing pick columns that promise real progress; smallest-index picks essentially an arbitrary one. Empirically this produces far more iterations, and there are constructed families of problems where Bland's rule takes exponentially many pivots on instances other rules solve quickly. It also has no notion of how far one can move — it just needs a valid improving direction.

Why lexicographic is better. It changes only the *ratio test* (the leaving-variable choice), and only when there are ties. The entering variable is still chosen by whatever pricing rule you like — Dantzig, steepest edge, devex. You keep all the normal pricing machinery and pay extra only when degeneracy actually appears. In the nondegenerate case there are no ties, so the rule costs nothing beyond the tie check.

The cost side of lexicographic. The tie-breaking itself is more expensive per occurrence: rows of B^{-1} (or of the perturbation vector) are compared component by component rather than comparing single indices. With a dense tableau that is fine; with a revised simplex and a factored basis, obtaining the needed rows of B^{-1} requires extra BTRAN solves. That is why textbook lexicographic order is not what most production solvers literally implement — they use equivalent-in-spirit bound perturbation instead, which is cheaper and achieves the same anti-degeneracy effect.

How they are actually used. The common pattern: run a good pricing rule with a normal ratio test; detect stalling (a long run of zero-length steps); switch on an anti-cycling mechanism — perturbation, lexicographic ordering, or Bland's rule — until you escape; then switch back. Bland's rule survives in this role because it is trivial to implement and provably terminates, not because it is fast. For permanently-on protection, lexicographic/perturbation is the better default.

	Bland's rule	**Lexicographic**
What it constrains	Entering *and* leaving variable	Leaving variable only (ties)
Pricing rule kept?	No — overridden	Yes — any pricing rule
Cost per iteration	Negligible	Extra row comparisons / BTRAN
Iteration count	Often very high	Close to unmodified simplex
Finite termination	Guaranteed	Guaranteed
Typical use	Emergency fallback on stalling	Always-on anti-degeneracy

Both guarantee finite termination, so the choice is purely about iteration count and implementation overhead, not about correctness.