ENGINEERING CASE STUDY / MIXED-INTEGER OPTIMIZATION

Fix the Formulation Before You Tune the Solver

Making production optimization models fast and numerically stable enough to trust

EXECUTIVE CONCLUSION

When a model cannot solve inside the decision window, the business quietly stops using it. Faster hardware rarely fixes that. Better formulations usually do.

Enterprise teams across industries had mixed-integer programs in production for planning, scheduling and allocation. The models worked, but they were slow, and each new business requirement made them slower and more fragile. This note sets out the engineering sequence that made them fast and stable: strengthen the formulation, control the numerics, remove what the business does not need, and only then tune the solver, measuring every change on the team's own instances. Computational experiments on three model families show which weaknesses a modern solver repairs by itself and which ones only the modeler can fix.

30–70%
reduction in solve times

2–5×
growth in model size, numerically stable

Formulation
first, solver tuning last

Measured
on production instances

Work led by Adam DeJans Jr. in an earlier industry role, engineering production solver deployments with enterprise optimization teams, not as a Bit Bros client engagement. The ranges are project-reported across engagements and not independently audited. Exhibits 1 and 2 are experiments run for this note on synthetic instances with an open-source solver; they are not engagement data.

Why speed and stability are business requirements

A mixed-integer program (MILP) that takes four hours in a two-hour planning window is not slow; it is unusable, and planners route around it with spreadsheets. A model that is numerically fragile is worse: a small change in input produces a large, unexplained change in the plan, and trust erodes faster than it can be rebuilt. Both failures look like technical problems and behave like adoption problems.

THE DECISION

Given a production model that is correct but too slow or too fragile, which changes will make it solve reliably inside the decision window, and in what order should they be made?

1. The relaxation decides how hard the problem is

Branch-and-bound lives or dies by its bound

A MILP solver explores a tree of subproblems and prunes any branch whose linear-programming (LP) relaxation bound cannot beat the best solution found. For a minimization with integer optimum z^{IP} and root relaxation value z^{LP}, the *root gap* $(z^{\mathrm{IP}} - z^{\mathrm{LP}})/z^{\mathrm{IP}}$ measures how much work the tree must do. Two formulations with exactly the same integer solutions can have very different relaxations, and therefore very different trees. That is why formulation strength is the first lever, not the last.

Formulations, defined

Let $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ be the set of feasible solutions. A polyhedron $P = \{(x, w) : Ax + Gw \le b\}$ is a *formulation* of X if $P \cap (\mathbb{Z}^n \times \mathbb{R}^p) = X$. Formulation P_1 is *stronger* than P_2 if $P_1 \subseteq P_2$; then for any objective c, $\min_{P_1} c^\top x \ge \min_{P_2} c^\top x$, so the root bound of P_1 is never worse. The strongest possible formulation is *ideal*, $P = \mathrm{conv}(X)$: every vertex is integral and the LP alone solves the problem.

Ideal and extended formulations

Ideal formulations are rarely available for a whole model, but they often are for its building blocks. Disjunctive programming (Balas) gives the convex hull of a union of polyhedra $P^k = \{x : A^k x \le b^k\}$, $k \in K$, by copying the variables:

$$x = \textstyle\sum_k x^k, \quad A^k x^k \le b^k \lambda_k, \quad \textstyle\sum_k \lambda_k = 1, \quad \lambda \ge 0.$$

For a semi-continuous quantity, $x \in \{0\} \cup [\ell, u]$, this yields $\ell y \le x \le u y$ with $y \in [0, 1]$, which is ideal; the familiar big-M form with an oversized u is a relaxation of it. Such *extended formulations* add variables whose projection onto the original space is much tighter than any compact model of similar size. The modeler's craft is to find, for each recurring structure in a production model, the strongest formulation the solver can still process quickly.

Aggregated big-M constraints are the classic weakness

In facility location, binary y_i opens facility i and $x_{ij} \in [0, 1]$ is the share of customer j served from it. The rule that customers use only open facilities can be written in aggregated form,

$$\textstyle\sum_j x_{ij} \le m\, y_i \qquad \forall i,$$

or disaggregated, one constraint per pair,

$$x_{ij} \le y_i \qquad \forall i, j.$$

Both admit the same integer solutions. **Proposition 1.** *The disaggregated relaxation is strictly stronger.* Summing $x_{ij} \le y_i$ over the m customers gives the aggregated row, so every point of the strong relaxation satisfies the weak one. The converse fails: let facility i serve one customer fully, $x_{i1} = 1$, with $y_i = 1/m$. Then $\sum_j x_{ij} = 1 \le m\, y_i$ holds, but $x_{i1} \le y_i$ does not. In the weak relaxation the facility is opened at one m-th of its fixed cost while serving that customer completely, and this is exactly the mechanism behind the gaps in Exhibit 1. It is not ideal in general, since uncapacitated facility location is NP-hard, but its relaxation is known to be very strong, and on all of our test instances its LP optimum was already integral.

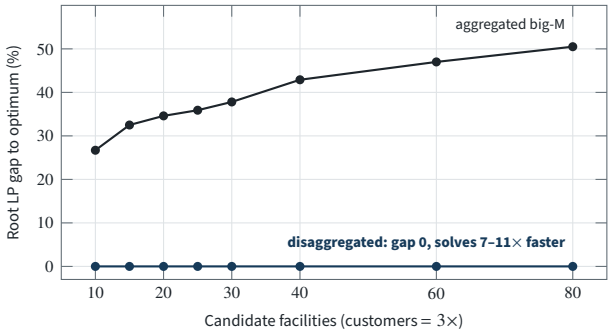


Exhibit 1. Gap between the LP relaxation and the integer optimum for random uncapacitated facility-location instances, mean of two or three instances per size. HiGHS through SciPy; synthetic instances.

Exhibit 1 shows the aggregated relaxation 27% below the optimum at the smallest size and 51% at the largest, while the disaggregated version solves 7 to 11 times faster. Both versions nevertheless finished without branching (a node count of one): HiGHS's presolve and root cutting planes closed the weak formulation's gap before the tree began. That observation deserves its own experiment.

2. What the solver repairs, and what it cannot

Presolve and cuts are formulation tools

Before branching, a modern solver rewrites the model. *Presolve* removes redundant rows and columns, tightens variable bounds by propagation, strengthens coefficients (shrinking an oversized big-M where the bounds of other variables allow it), and *probes* binaries by tentatively fixing them to learn implications. At the root, *cutting planes* then add valid inequalities that cut off the fractional LP solution: Gomory mixed-integer and mixed-integer rounding (MIR) cuts from the simplex tableau, knapsack cover cuts from single rows, flow cover cuts from fixed-charge structure, and implied-bound cuts such as $x_{ij} \le y_i$ derived from variable bounds. Much of what a modeler might add by hand, the solver adds by itself.

A cut, derived

The Chvátal–Gomory procedure shows how integrality creates inequalities. Take $4x_1 + 6x_2 \le 9$ with $x \in \mathbb{Z}^2_{\ge 0}$. The LP point $(0, 1.5)$ is feasible. Multiply by $\frac{1}{2}$ to get $2x_1 + 3x_2 \le 4.5$; because the left side is integral for integral x, the right side can be rounded down: $2x_1 + 3x_2 \le 4$, which cuts off $(0, 1.5)$ and

lowers the LP maximum of x_2 to 4/3. For bounded pure integer programs, every valid inequality is implied by finitely many rounds of this rounding (Chvátal). Mixed-integer rounding generalizes it to continuous variables: for $\{(x, s) : x + s \geq b,\ x \in \mathbb{Z},\ s \geq 0\}$ with $f = b - \lfloor b \rfloor > 0$, the inequality $s \geq f(\lceil b \rceil - x)$ is valid, and Gomory mixed-integer cuts apply it to rows of the optimal simplex tableau.

A three-family experiment

To separate what the solver repairs from what it cannot, we compared a weak and a strong formulation of three classic structures on random instances (two to five per size), with identical integer solutions in each pair.

Model and strong form	Weak root gap	Strong root gap	Effect on solving
Uncapacitated location, disaggregated	27–51%	0%	7–11× faster
Capacitated location, adding $x_{ij} \leq y_i$	6–12%	under 1.2%	roughly equal; the solver adds these as cuts
Capacitated lot sizing, facility-location extended form	83–94%	under 1.2%	4–6× faster; up to 21 nodes against 1

Exhibit 2. Root LP gaps and solving effort for three model families. Capacitated location: 20 to 40 sites, five instances per size. Lot sizing: 8 to 16 items, 12 to 18 periods, shared capacity with setup times, five instances per size. Times are shifted geometric means (shift 0.1 s). HiGHS through SciPy; synthetic instances; scripts in the note's repository folder.

The pattern is instructive. In capacitated facility location, the missing constraints $x_{ij} \leq y_i$ are exactly the implied-bound cuts the solver knows how to generate, so writing them explicitly changed little. In lot sizing, the weak model uses the convenient constant $x_{pt} \leq C_t\, y_{pt}$, with C_t the period's capacity. The strong model replaces production quantities by variables z_{pts}, the share of item p's demand in period s produced in period t, with $z_{pts} \leq y_{pt}$ (Krarup and Bilde). In full, for items p, periods $t \leq s$, demand d_{ps}, setup time σ_p and capacity C_t:

$$\begin{aligned} &\textstyle\sum_{t \leq s} z_{pts} = 1 && \forall p, s :\ d_{ps} > 0, \\ &z_{pts} \leq y_{pt} && \forall p,\ t \leq s, \\ &\textstyle\sum_p \left(\sum_{s \geq t} d_{ps} z_{pts} + \sigma_p y_{pt} \right) \leq C_t && \forall t, \end{aligned}$$

with holding cost $h_p(s - t)d_{ps}$ on z_{pts}. For a single uncapacitated item this formulation is ideal; it projects onto the classical (ℓ, S) inequalities of Barany, Van Roy and Wolsey,

$$\textstyle\sum_{t \in S} x_t \ \leq\ \sum_{t \in S} d_{t\ell}\, y_t + I_\ell, \qquad S \subseteq \{1, \ldots, \ell\},$$

where $d_{t\ell}$ is total demand from t to ℓ and I_ℓ is ending inventory. Its relaxation sits within about one percent of the optimum, where the big-M relaxation is off by more than four-fifths. No general-purpose cut family recovers that structure reliably, and on production models with many interacting rules, such gaps become search trees that do not finish.

THE PRACTICAL RULE

Let the solver add what it can derive from single rows and variable bounds. Spend modeling effort on structure it cannot see: extended formulations, problem-specific valid inequalities, and bounds that come from business knowledge.

3. Control the numerics

Tolerances decide what "feasible" and "integral" mean

Solvers work in floating point and accept a solution as feasible within a primal feasibility tolerance (commonly 10^{-6}), as optimal within a dual or optimality tolerance, and a variable as integral within an integrality tolerance (commonly 10^{-5} or 10^{-6}, depending on the solver). With $x \leq My$ and $M = 10^7$, a value $y = 10^{-5}$ can count as zero while permitting x up to 100: the solver reports a "closed" facility that ships 100 units. Formally, a binary with $y \leq \varepsilon_{\text{int}}$ is accepted as zero, so the constraint permits leakage up to $M\,\varepsilon_{\text{int}}$, and M must satisfy $M\,\varepsilon_{\text{int}}$ below the smallest quantity that matters. The fix is to derive each M as the smallest valid bound. For a row $a^\top x \leq b + M(1 - y)$ that must be inactive when $y = 0$, over variable bounds $\ell \leq x \leq u$,

$$M = \textstyle\sum_j \max(a_j \ell_j,\ a_j u_j) - b,$$

and business bounds tighten it further: production of item p in period t never needs to exceed $\min\left(C_t - \sigma_p,\ \sum_{s \geq t} d_{ps}\right)$. In the lot-sizing instances of Exhibit 2, replacing the capacity constant by this derived bound cut the mean root gap from 89% to 56%, a large gain from one line of arithmetic, yet still far from the 0.4% of the extended formulation. Indicator constraints, enforced logically by the solver, avoid the coefficient altogether.

Conditioning and coefficient ranges

Every LP solve inside branch-and-bound factorizes basis matrices B. The condition number $\kappa(B) = \|B\|\,\|B^{-1}\|$ bounds error amplification: to first order, a relative perturbation δ in the data can move the basic solution by a relative $\kappa(B)\,\delta$. With double precision ($\delta \approx 10^{-16}$) and $\kappa \approx 10^{10}$, results are trustworthy only to about six digits, the same order as the feasibility tolerance, and plans change with irrelevant perturbations. Scaling replaces A by $D_r A D_c$ with diagonal D_r, D_c chosen to bring $\max|a_{ij}|/\min|a_{ij}|$ toward one. Choosing units sensibly (thousands of units, millions of dollars), scaling rows and columns, and tracking the ranges of matrix, bound and objective coefficients as a routine diagnostic catches most problems before they surface as unstable plans; solver logs report these ranges and, on request, condition estimates.

Symptom	Usual cause and remedy
Plan swings on tiny input changes	Alternative optima with no tie-break; add a change penalty against the previous plan
"Closed" assets carrying flow	Oversized big-M against the integrality tolerance; derive M or use indicators
Different answers across runs	Ill-conditioning or performance variability; check ranges and repeat with seeds
Infeasible with no clear reason	Conflicting rules; compute an irreducible infeasible subsystem

4. Remove what the business does not need

Every constraint should have a provenance

Production models accumulate constraints: a minimum added after one bad cycle, a buffer that compensated for a data problem since fixed, a rule nobody can trace to an owner. Each enlarges the model and may weaken or complicate it. Reviewing every constraint family against its provenance (physical limit, contract, policy or habit) routinely removes rules the business no longer requires, shrinking the model without changing its answers. Soft preferences belong in the objective with an explicit price, not in hard constraints.

Handle symmetry deliberately

Identical machines, vehicles or shifts create many equivalent solutions that branch-and-bound explores separately. Modern solvers detect symmetry and apply orbital fixing or orbitopal reductions automatically. Hand-written symmetry-breaking constraints, such as lexicographic ordering of identical resources, can help further, but they can also conflict with the solver's own symmetry handling and heuristics, so they should be benchmarked rather than assumed.

5. When reformulation is not enough

Some models remain too large or too loosely coupled for any compact formulation. Decomposition then exploits structure explicitly: *Benders decomposition* when a few integer decisions, such as which sites to open, drive many continuous ones; *Lagrangian relaxation* when a handful of coupling constraints link otherwise independent subproblems, giving strong bounds and good heuristics; and *column generation* (branch-and-price) when solutions are naturally built from a huge set of patterns, such as routes, shifts or cutting patterns.

Benders decomposition, written out

In capacitated facility location with site capacities s_i, demands d_j and transport costs c_{ij}, fix the binary decisions $\hat{y}$ and the rest is a transportation LP, $Q(\hat{y}) = \min\{\sum c_{ij} x_{ij} : \sum_i x_{ij} = d_j,\ \sum_j x_{ij} \leq s_i \hat{y}_i,\ x \geq 0\}$. Its dual, with u_j free and $w_i \geq 0$, is

$$Q(\hat{y}) = \max\left\{ \textstyle\sum_j d_j u_j - \sum_i s_i \hat{y}_i w_i :\ u_j - w_i \leq c_{ij} \right\},$$

whose feasible region does not depend on $\hat{y}$. Each dual vertex $(\bar{u}, \bar{w})$ therefore gives an *optimality cut* $\eta \geq \sum_j d_j \bar{u}_j - \sum_i s_i \bar{w}_i y_i$, and each extreme ray a *feasibility cut* $\sum_j d_j \bar{u}_j \leq \sum_i s_i \bar{w}_i y_i$. The master problem keeps y and a growing set of cuts.

BENDERS LOOP

```
C ← ∅,  LB ← −∞,  UB ← +∞
while UB − LB > ϵ |UB| do
    solve master min{f⊤y + η : (y, η) satisfies C}; set LB, ŷ
    solve dual subproblem at ŷ
    if unbounded then
        add feasibility cut from the ray to C
    else
        add optimality cut; UB ← min(UB,  f⊤ŷ + Q(ŷ))
    end if
end while
```

Modern implementations solve the master once by branch-and-cut and add cuts at integer incumbents, rather than re-solving it from scratch. Decomposition is powerful but costs engineering effort and maintenance, so it is warranted only after the compact model has been made as strong as it can be.

6. Tune the solver last

Once the formulation is sound, targeted tuning helps. A *MIP start* from the previous plan is often the single most useful setting in recurring planning: the solver begins with a good incumbent, prunes from the first node, and tends to return a plan close to the last one. Emphasis settings trade finding good solutions early against proving optimality; heuristic and cut aggressiveness, presolve level and solution polishing each matter on some models. The time limit and gap target should match the decision: a 1% optimality gap is irrelevant when input data carry 10% error, and chasing it wastes the planning window. Tuning a weak formulation, by contrast, mostly buys noise.

7. Benchmark like an experiment

Performance variability is real

MILP run times change substantially with seemingly irrelevant changes, such as row order, a random seed or a different machine, because small differences alter the branching path. A single run before and after a change proves nothing. Two summaries make comparisons honest. The *shifted geometric mean* of times $t_1, \ldots, t_n$ with shift ς,

$$\mathrm{SGM}_\varsigma = \exp\Big(\tfrac{1}{n}\textstyle\sum_i \ln(t_i + \varsigma)\Big) - \varsigma,$$

damps both trivially easy instances and single outliers. A *performance profile* for method m on instance set $\mathcal{P}$ plots

$$\rho_m(\tau) = \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} : \frac{t_{p,m}}{\min_{m'} t_{p,m'}} \le \tau\right\}\right|,$$

the share of instances method m solves within a factor τ of the best; $\rho_m(1)$ is how often it wins and $\lim_{\tau\to\infty} \rho_m(\tau)$ how often it solves at all.

Practice	Why
Production instance set	Synthetic or single instances do not represent the structure that matters
Several seeds or permutations	Separates real improvement from performance variability
Shifted geometric mean	Summarizes times without letting easy instances or one outlier dominate; state the shift
Gap at the time limit	Captures progress on instances that do not finish
Performance profiles	Show, for each method, the share of instances solved within a factor of the best
Separate tuning and test sets	Prevents parameter settings that fit the benchmark rather than the business

Diagnose infeasibility for planners, not just modelers

When a plan is infeasible, an irreducible infeasible subsystem (IIS), a minimal set of constraints that cannot hold together, tells the modeler which rules conflict. Planners usually need a different answer: the cheapest way to restore feasibility. An *elastic* formulation adds penalized slack to soft rules and reports which rule to bend, by how much, at what cost, which turns an error message into a decision.

REFERENCE ENGINEERING SEQUENCE

Strengthen the formulation → control numerics → remove unneeded rules → decompose if needed → tune and warm-start → benchmark on real instances

8. What the work reported

Across engagements, solve times fell by 30 to 70 percent, and models grew two to five times larger while remaining numerically stable, giving teams room to add the business detail they had previously deferred. The largest gains usually came from the formulation itself, followed by targeted solver tuning and numerical diagnostics. These ranges are project-reported and have not been independently audited.

9. Lessons that transfer

The relaxation is the product

Two models with the same feasible set can differ by orders of magnitude in solve time. Judge a formulation by its relaxation, not only by its correctness.

Know what the solver already does

Presolve and cuts repair local weaknesses derived from single rows and bounds. Modeling effort pays where they cannot reach: extended formulations and structure only the business knows.

Every big-M is a numerical decision

Derive it from the data or replace it with an indicator constraint. A convenient large constant is a latent bug.

Speed buys model detail

A model that solves in a third of the time can carry the business rules that were deferred for performance. Engineering speed is how a model keeps up with the business.

A USEFUL EXECUTIVE DIAGNOSTIC

If your planning model's run time doubled next quarter, would your team reformulate it, buy hardware, or quietly stop using it?

Further reading

Vielma, J. P. Mixed integer linear programming formulation techniques. *SIAM Review* 57(1), 2015.

Pochet, Y. and Wolsey, L. A. *Production Planning by Mixed Integer Programming*. Springer, 2006.

Balas, E. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic and Discrete Methods* 6(3), 1985.

Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4, 1962.

Achterberg, T. et al. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing* 32(2), 2020.

Klotz, E. and Newman, A. M. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science* 18(1), 2013.

Lodi, A. and Tramontani, A. Performance variability in mixed-integer programming. *INFORMS TutORials*, 2013.

Dolan, E. D. and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming* 91(2), 2002.

Chinneck, J. W. *Feasibility and Infeasibility in Optimization*. Springer, 2008.

DeJans Jr., A. *The MILP Optimization Handbook*. Bit Bros, 2025.

About the case and Bit Bros

This paper draws on formulation and solver engineering work led by **Adam DeJans Jr.** in an earlier industry role, working with enterprise optimization teams on production deployments. It was not a Bit Bros client engagement. Clients, models and data are not disclosed, and the experiments in Exhibits 1 and 2 use synthetic instances. Bit Bros is a boutique applied economics and optimization firm that helps operating teams price their decisions and build the systems that make them repeatable.



Adam DeJans Jr.
Technical Managing Director, Bit Bros LLC
Decision scientist specializing in mixed-integer optimization, solver engineering and decision systems.

Strengthen the model. Then tune the solver.