

Scheduling Under Resource Limits, by Genetic Algorithm

Review of Chunlai Chai, *Modeling Resource-constrained Project Scheduling Problem and its Solution by Genetic Algorithm* (Journal of Digital Information Management, April 2013, 10 pp.)

TLDR / 80-20

The method is ordinary; one design principle is worth keeping.

Build your candidate so it *cannot express an illegal answer*, instead of filtering out broken ones later.
And never let the search destroy the best thing it has found.

The Keys

1. The problem, and why exact methods die.

Jobs must run in order and compete for the same limited people and machines. Gantt charts, CPM and PERT quietly assume unlimited resources; add the limits and exact answers cost exponentially more. Past a small size, you stop solving and start searching.

2. The trick: evolve the pecking order, not the timetable.

Each task carries a priority number. A decoder walks the network, starts the best-priority task whose predecessors are done, and waits a day if the resources are busy. So *every* priority list decodes to a valid schedule — the roof can never precede the walls.

3. The score comes out of the far end, and it is relative.

Play the ordering through and read the finish date, rescaled against the generation's best and worst; selection is a roulette wheel over those shares. Note what is missing: priorities start as **pure random numbers** — no critical path, no slack, no "this unblocks five others".

4. The experiment, and the failure the author names himself.

Twenty tasks, three resource pools, 50 generations, a 600-day schedule. Population 200 beat 100 (600 vs 620 days), and dropping mutation from 0.2 to 0.02 smoothed convergence. But his optimum keeps stumbling backwards, and he says why: **crossover and mutation can destroy his best candidate**. The rival paper keeps its champion untouched, and converges better.

AI's Take

As evidence this is thin: one example project, no benchmark, no fair comparison, and an abstract that contradicts its own model about whether tasks can be interrupted. The author is beaten by the paper he cites, over a fix worth one line of code.

What survives is the shape: **something proposes, something measures, the measurement bends what gets proposed next**. Evolutionary search, RL and RLHF share it, differing only in where the score comes from — a formula here, a human in RLHF, a verifier in between. This is not RL, though: nothing learns, candidates are killed or kept. Killing needs no gradient, so it works on any black box you can run and time.

The lesson to carry: **a ranked list without a measured outcome is an opinion**. Most of our priority lists are this paper's front half with the scoreboard missing — and the scoreboard does the work.