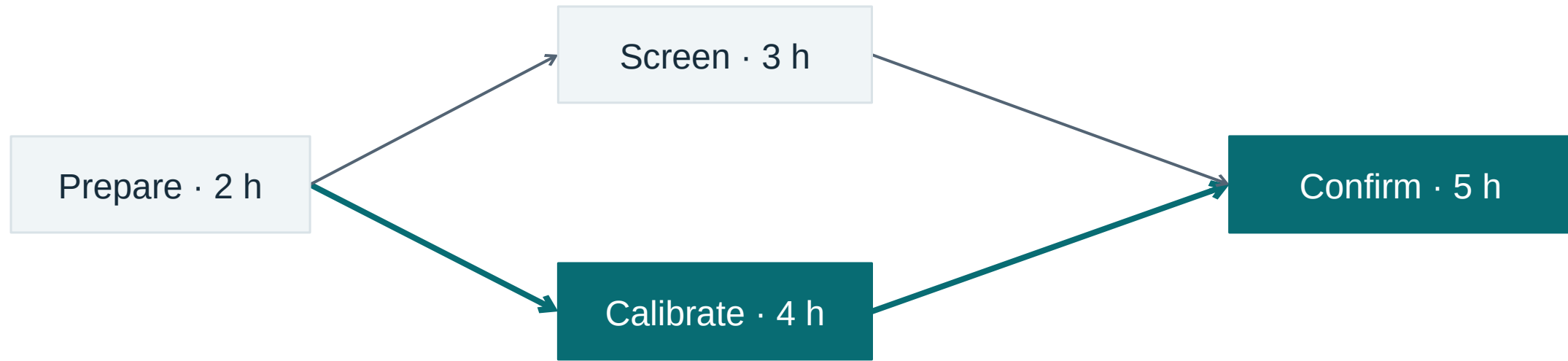


Sequential and Parallel Discovery

How much of a discovery process can more simultaneous work accelerate?

Complexity, Information, and Cross-Domain Transfer

A four-operation discovery workflow



Critical path: prepare → calibrate → confirm = 11 h

Prepare for 2 h; then screen for 3 h and calibrate for 4 h in parallel; confirm for 5 h after both finish. Only the middle operations can overlap. If more workers leave the finish time unchanged, the dependency graph suggests investigating waiting rather than worker count.

Which change can actually shorten this workflow?

Change from two workers	Finish time	Reason
Add more workers	11 h	No additional ready operation
Screen in 1 h instead of 3	11 h	Calibration still finishes later
Calibrate in 1 h instead of 4	10 h	Screening now determines the wait

Idealized durations; a faster procedure must preserve the same scientific acceptance standard.

Work, depth and wall-clock time

Total work W adds all durations: $2+3+4+5=14$ h. Weighted depth D follows the longest dependency chain: $2+4+5=11$ h.

Elapsed time also depends on workers and overhead. Experimental adaptivity adds another dependency: the next question may need an earlier result.

An ideal greedy schedule nearly meets the lower bound

$$\max(W/P, D) \leq TP \leq W/P + D$$

With P identical processors and no communication overhead, a greedy scheduler keeps processors busy whenever enough tasks are ready.

Full steps consume P units of work. In underfull steps, the remaining critical path shortens. This yields $TP \leq W/P + D$.

A numerical work–depth comparison

average parallelism = $W/D = 10$

For $W=1,000$ and $D=100$, ten processors give bounds $100 \leq T_{10} \leq 200$. One hundred processors give $100 \leq T_{100} \leq 110$.

Beyond about $W/D=10$ processors, sequential depth increasingly limits the gain.

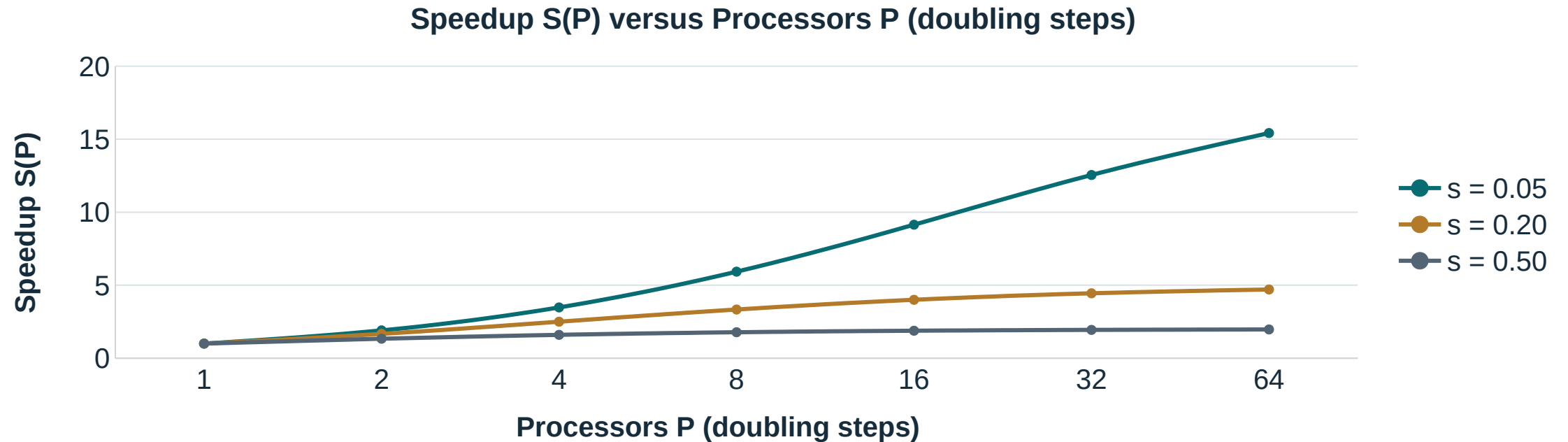
Amdahl's fixed-workload model

$$\text{Speedup}(P) = 1 / [s + (1-s)/P]$$

Let s be the serial fraction of a fixed workload and $1-s$ the ideally parallel fraction. Normalizing one-processor time to one gives the displayed speedup.

As P grows, speedup approaches $1/s$. The workload and algorithm are held fixed.

A fixed serial fraction creates a speedup ceiling



Illustrative Amdahl curves: $S(P) = 1/[s + (1-s)/P]$. With serial fraction $s=0.20$, even unlimited processors cannot exceed a fivefold speedup for this fixed workload. Horizontal spacing represents successive doublings.

More processors cannot remove the serial fraction

Processors P	Normalized time	Speedup
1	1.000	1.00
10	0.190	5.26
100	0.109	9.17
Unlimited	0.100	10.00

If $s=0.1$, speedup is about 5.26 at $P=10$ and 9.17 at $P=100$; the limiting speedup is ten.

Parallel search increases coverage

$$\text{Pr(at least one success)} = 1 - (1 - p)^k$$

With independent success probability p per attempt, k parallel attempts succeed at least once with probability $1 - (1 - p)^k$.

The formula presumes independent attempts with the same success probability and a valid success checker.

Correlated attempts can waste a large batch

copies: 0.10; independent attempts: $1 - 0.9^{10} \approx 0.651$

If all attempts are exact copies, k attempts still succeed with probability p . Under independence with $p=0.1$ and $k=10$, success is about 0.651.

More proposals are useful when they expand effective coverage, not merely when their text differs.

Adaptive rounds are a separate resource

An R -round policy chooses a batch, observes its outcomes, then chooses the next batch. All actions inside a batch precede feedback from that batch.

Fewer rounds can require more total queries because the policy cannot tailor each next action to the latest evidence.

Threshold search demonstrates an adaptivity advantage

adaptive: $\log_2(1024)=10$; one batch: $q+1 \geq 1024$

An unknown threshold lies among 1,024 ordered possibilities. Noiseless comparison queries support binary search in ten adaptive rounds.

In one nonadaptive batch, q comparison points create at most $q+1$ intervals. Exact identification requires at least 1,023 queries.

The comparison is about allowed questions

Ten arbitrary bit queries could identify a 1,024-valued index in one batch. Ten threshold comparisons cannot.

An adaptivity claim must state the query family and how outcomes depend on the hidden target.

Batch setup changes time, not the evidence available

Policy for ten assays	Choice information	Completion
Two batches of five	All chosen before results	5 h
Ten individual runs; two instruments	Refresh choice after each run	25 h
Two adaptive batches of five	Second follows first result	10 h

Two instruments; each batch needs 1 h setup plus 4 h assay time. A batch can contain up to five assays.

Delayed feedback changes allocation

Suppose 20 assays fit in a daily batch. A policy requiring five sequential rounds takes at least five days even if it uses only 100 assays.

With enough extra instruments, running 100 assays immediately may finish sooner but can test the wrong candidates because later-round information is absent.

Speculative work exchanges compute for latency

binary branching over R rounds $\rightarrow$ up to 2^R leaves

A policy can compute next steps for multiple possible outcomes before the current experiment returns. Once the outcome is known, it keeps the matching branch.

This can reduce response latency at the price of work on branches never used.

Sequential errors accumulate unless checked

$$\Pr(\text{chain succeeds})=q^R; \quad 0.99^{100} \approx 0.366$$

If each of R necessary steps independently succeeds with probability q , the whole chain succeeds with probability q^R .

At $q=0.99$ and $R=100$, success is about 0.366. Local checks, recovery and redundancy change this model.

Checkpoints can shorten recovery

A verified intermediate result can be reused after a later failure. Without a checkpoint, the entire chain may need repetition.

Checkpointing costs time and storage, but can reduce expected wasted work when failures occur.

Mathematical search mixes parallel and sequential work

Many candidate constructions can be evaluated independently. A proof or constructive improvement can also require a sequence of dependent lemmas.

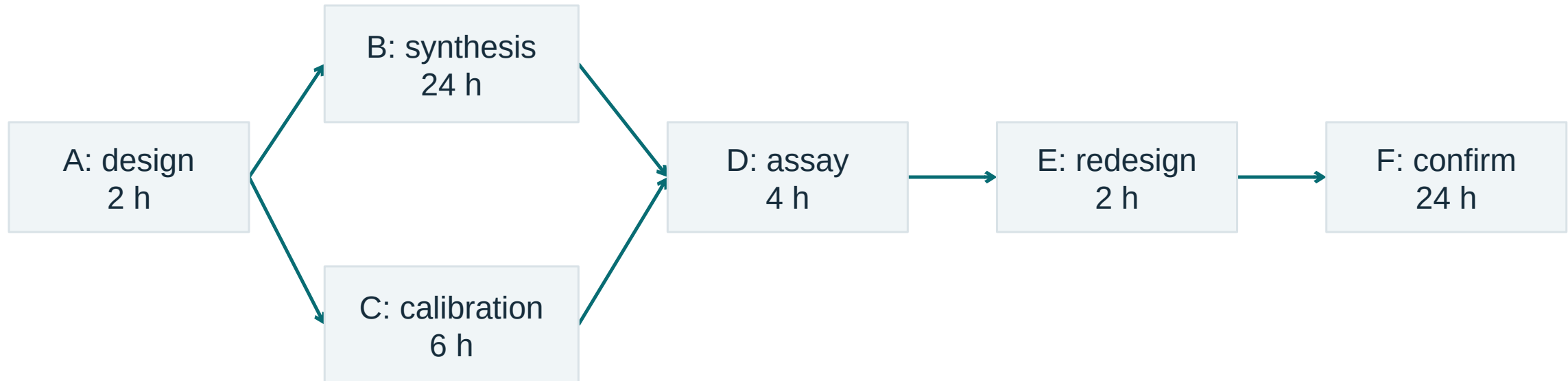
The strongest candidate found in parallel may become the starting point for a later adaptive search phase.

Synthesis depends on the feedback granularity

Counterexample-guided refinement naturally creates adaptive rounds. Enumerating independent programs can run in parallel before feedback.

A parallel verifier farm improves throughput, while richer counterexamples may reduce the number of future rounds.

A two-round experimental dependency graph



Baseline: critical path A–B–D–E–F is 56 h; total work is 62 activity-hours. If D takes 28 h instead of 4 h, completion takes at least 80 h and work rises to 86 activity-hours. Extra parallel capacity cannot remove that wait.

Mechanism discovery can require sequential redesign

A first perturbation may reveal that the original model class is inadequate. The next useful experiment may not be definable until that failure is interpreted.

Parallel experiments from the original class can then have low value despite high throughput.

2026: explicitly allocating width and depth

$$N = C \times L \times K$$

SimpleTES, or Simple Test-time Evaluation-driven Scaling, organizes queries by global trajectories C , refinement depth L and local candidate count K .

Its nominal evaluator budget is $N=C \times L \times K$. Equal N can correspond to different feedback structures and different elapsed time.

Equal query budgets can have unequal elapsed time

Full-run budget: 12,800 queries	SimpleTES to matched score	OpenEvolve full run
Third autocorrelation task	11.2 minutes	132.7 minutes
Sum-difference task	3.3 minutes	59.4 minutes

Both use up to 128 concurrent requests. OpenEvolve is an evolutionary coding agent; times compare reaching its whole-run best score.

An equal-budget comparison must preserve the cost boundary

At $N=1,000$, $(C,L,K)=(100,10,1)$ and $(10,10,10)$ both use 1,000 evaluations, but compare different local and global diversity.

A valid comparison also includes evaluator duration, rejected proposals, training and communication overhead.

What a work–depth model leaves out

Communication, queueing, hardware heterogeneity, sample deterioration and human interpretation can dominate real time.

Treat ideal scheduling bounds as diagnostic baselines. They identify possible bottlenecks without estimating every practical delay.

Cross-domain resource comparison

Domain	Often parallel	Potential critical path
Construction	Candidate checking	Dependent proof or redesign
Program	Independent tests	Counterexample refinement
Molecule	Library screening	Physical preparation and assay
Mechanism	Perturbation panel	Model revision after outcomes

The same number of attempts has different temporal implications in each domain.

Open question: irreducible discovery depth

Can a task require many adaptive rounds under realistic experimental access even when total compute is enormous?

Formal oracle examples establish that this can happen. Measuring such limits for real scientific domains remains substantially harder.

Conclusion: parallelism still needs feedback

Work determines how much activity is performed; depth and adaptive rounds constrain when useful outcomes become available.

Next: how to model scaling when compute, evidence, experiments and time can substitute only partially.

References

Richard P. Brent (1974). The Parallel Evaluation of General Arithmetic Expressions. Journal of the ACM 21, 201–206; journal article

Gene M. Amdahl (1967). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. AFIPS Spring Joint Computer Conference, 483–485; conference paper

Daniel Golovin and Andreas Krause (2011). Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. JAIR 42, 427–486 (2011); inspected arXiv v5, 6 December 2017

Claude E. Shannon (1948). A Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423, 623–656; journal article

References · continued

[Kevin Ellis et al. \(2021\). DreamCoder: Bootstrapping Inductive Program Synthesis with Wake-Sleep Library Learning. PLDI 2021; conference paper; preprint has earlier title](#)

[Thomas Hubert et al. \(2025\). Olympiad-level formal mathematical reasoning with reinforcement learning. Nature; journal article, 2025](#)

[Bernardino Romera-Paredes et al. \(2024\). Mathematical discoveries from program search with large language models. Nature 625, 468–475; online 14 December 2023](#)

[Alexander Novikov et al. \(2025\). AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv:2506.13131; white paper/preprint](#)

References · continued

Sumit Gulwani, Oleksandr Polozov and Rishabh Singh (2017). Program Synthesis. Foundations and Trends in Programming Languages 4, 1–119; survey monograph

Judea Pearl and Elias Bareinboim (2014). External Validity: From Do-Calculus to Transportability Across Populations. Statistical Science 29, 579–595; journal article; arXiv posted 2015

Jonas Peters, Peter Bühlmann and Nicolai Meinshausen (2016). Causal inference by using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society B 78, 947–1012; journal article

Haotian Ye et al. (2026). Structured Scaling of AI Discovery Across Diverse Scientific Domains. arXiv:2604.19341v2; preprint, 27 July 2026