Mutation-Set Discovery for Cell-Fate Control: Nested Monte Carlo Search on Boolean Network Ensembles

Xichen Zhang
LAMSADE,
Université Paris-Dauphine,
Université PSL,
Paris, France

Loïc Paulevé
LaBRI,
Université de Bordeaux,
Bordeaux, France

Tristan Cazenave
LAMSADE,
Université Paris-Dauphine,
Université PSL,
Paris, France

Abstract

We study the efficient discovery of mutation sets in Boolean-network ensembles, where robust evaluation is costly, and steady-state estimates are noisy with few trajectories. We first derive a variance decomposition that separates across-model diversity from within-model simulation noise, yielding a simple tolerance-to-budget rule for allocating M (models) and n (trajectories). We then benchmark NMCS, Lazy NMCS, and NRPA, and introduce Bi-Lazy NMCS: a two-level scheme that prunes with a small, cheap ensemble and confirms with a large, accurate one. On the Cohen model, Bi-Lazy NMCS offers strong anytime performance and better scalability under fixed time budgets.

1 Introduction

Boolean networks (BNs) are classical models to capture qualitative aspects of gene-regulatory dynamics (see Figure 1) and cell-fate decisions such as apoptosis, invasion, and metastasis (Cohen et al., 2015). A BN is a dynamical model in which each gene (node) is a binary variable indicating whether it is active (1) or inactive (0), and each node has a logical rule specifying its next state as a function of its regulators. Given an initial configuration, one simulates the system by applying these rules according to an update mode: in the synchronous mode, all nodes update simultaneously; in the asynchronous mode, a single node updates at a time, chosen non-deterministically. Logical rules aim to reflect biomolecular regulation, which, in practice, is only partially known.(Chevalier et al., 2020; Naldi et al., 2018; Stoll et al., 2012).

To consider this uncertainty, recent work advocates ensembles of BNs that share the same topology but differ in local update rules (Schwab et al., 2021; Chevalier et al., 2020, 2025). Ensemble modeling improves robustness by averaging predictions over BNs, but this operation raises computational costs. Throughout, we use asynchronous simulations to approximate BN's steady-state, controlled by two parameters: the number of trajectories per model n_{traj} and the ensemble size M.

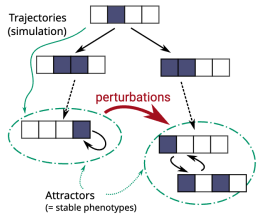
A central challenge is search: finding mutation sets that drive the ensemble toward a target phenotype is combinatorial. Exhaustive evaluation is infeasible for two reasons: (i) large graphs explode the candidate space, and (ii) large ensembles make each evaluation costly. Monte-Carlo–based search may help: Monte-Carlo Tree Search (MCTS) and UCT provide the general framework (Coulom, 2007; Kocsis and Szepesvári, 2006; Browne et al., 2012; Gelly and Silver, 2012); Nested Monte-Carlo Search (NMCS) specializes in single-agent optimization by recursively searching for the best-sequence (Cazenave, 2009); NRPA adds a learned stochastic rollout policy adapted to the best sequence (Rosin, 2011); and "lazy" pruning variants reduce branching by cheaply screening moves before deeper search (Roucairol and Cazenave, 2023; Roucairol et al., 2024). We also integrate search with simple variance budgeting for noisy phenotype estimates based on concentration bounds (Hoeffding, 1963; Audibert et al., 2009).

In this ensemble setting, We ask:

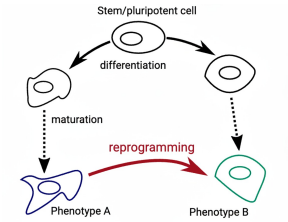
> ***Q1:*** *What really influences the variability of an ensemble of BNs?*
>
> ***Q2:****How can we efficiently search mutation sets in Boolean-network ensembles?*

We bridge steady-state estimation and Monte Carlo search, comparing NMCS, LNMCS, and NRPA while proposing *Bi-Lazy NMCS* (BILNMCS), a two-level scheme that uses a small, inexpensive ensemble for

pruning and a large, accurate one for confirmation.



(a) Dynamical model view: discrete states evolve under update rules and settle into attractors (fixed points or cycles).



(b) Cell-fate view: regulatory programs guide differentiation, maturation, and reprogramming decisions.

Figure 1: Boolean networks as a classic modeling paradigm for regulatory systems.

2 Background and Problem Setup

2.1 Boolean Networks

Let $n \in \mathbb{N}$, $[n] = \{1, \dots, n\}$, and $\mathbb{B} = \{0, 1\}$. A Boolean network (BN) is a function $F : \mathbb{B}^n \to \mathbb{B}^n$,

$$F(x) := \big(f_1(x), \dots, f_n(x)\big) \qquad (x \in \mathbb{B}^n)$$

where $f_i : \mathbb{B}^n \to \mathbb{B}$ is the local Boolean function of component $i \in [n]$, and any $x \in \mathbb{B}^n$ is a configuration (see the toy example Figure 2). We write $\Delta(x, y) = \{i \in [n], x_i \neq y_i\}$ the nodes that differ in two configurations. Its *influence graph* $\mathcal{G}(F)$ is a signed digraph over its nodes $[n]$, with $i \xrightarrow{+} j$ (resp. $i \xrightarrow{-} j$) whenever there exists $x, y \in \mathbb{B}^n$ such that $\Delta(x, y) = \{i\}$, $x_i = 0$ and $f_j(x) < f_j(y)$ (resp. $f_j(x) > f_j(y)$). In this work, we adopt asynchronous semantics to define the transition relation $\xrightarrow[F]{a1}$ between pairs of configurations $x, y \in \mathbb{B}^n$:

$$x \xrightarrow[F]{a1} y \iff \exists i \in [n] : \ \Delta(x, y) = \{i\} \text{ and } y_i = f_i(x),$$

with $\xRightarrow[F]{a1}$ its reflexive–transitive closure. The relation $\xrightarrow[F]{a1}$ forms a digraph between configurations $\mathbb{B}^n$. Attractors (including fixed points) summarize long-run behavior under this update mode and correspond to the inclusion-smallest sets of configurations closed by $\xRightarrow[F]{a1}$. A phenotype is a Boolean condition ϕ over configurations.

2.2 Mutated Boolean Networks

A mutation fixes a node i to $v \in \mathbb{B}$: (i, v). For a set $\mathcal{M}$ with at most one pair per node,

$$(F/\mathcal{M})_i(x) = \begin{cases} v, & (i, v) \in \mathcal{M}, \\ f_i(x), & \text{otherwise.} \end{cases}$$

Restricting to modifiable regulators $\mathcal{I} \subseteq [n]$ ($g := |\mathcal{I}|$), the k-mutant search space is

$$\mathcal{S}_k = \{\mathcal{M} \subseteq \mathcal{I} \times \mathbb{B} : \ |\mathcal{M}| = k\}, \qquad |\mathcal{S}_k| = \binom{g}{k} 2^k.$$

This combinatorial growth motivates the search strategies used later.

Toy BN (n=3). $f_1(x) = \neg x_2$, $f_2(x) = \neg x_1$, $f_3(x) = \neg x_1 \wedge x_2$. Nodes 1 and 2 form a mutual-inhibition toggle; node 2 activates 3 while node 1 inhibits 3. Under asynchronous updates, this yields two fixed points: $(1, 0, 0)$ and $(0, 1, 1)$.

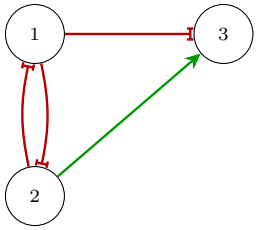


Figure 2: Influence graph $\mathcal{G}(F)$ for the toy BN. Green: activation; red: inhibition.

In Figure 3, Asynchronous single-step trajectories on a toy Boolean network under single gain-of-function mutations $\mathcal{M} = \{(i, 1)\}$. Each cube displays the configuration graph; thick arrows indicate the enabled update from the annotated start state, and boxed vertices represent fixed points. The three cases illustrate how mutating one node can reprogram the system by changing dynamics to different attractors: fixing node 1 locks the network at 100, fixing node 2 drives 010 → 011, and fixing node 3 drives 001 → 101.

2.3 Problem Setup

In the following, we consider $\mathscr{F} = \{F_{(m)}\}_{m=1}^{M}$ an *ensemble* of M distinct BNs over the same fixed nodes $[n]$, and an initial configuration $x^0 \in \mathbb{B}^n$.

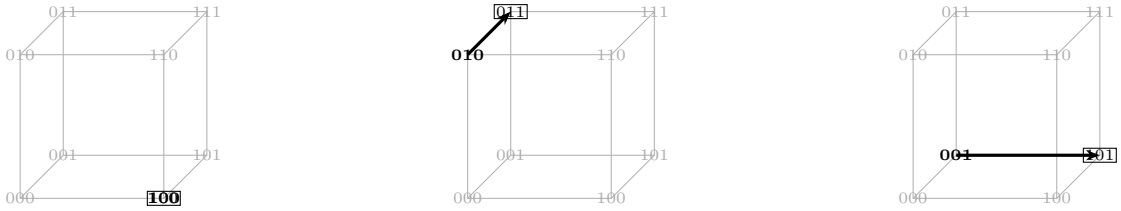
Given a mutation set $\mathcal{M}$, for any $m \in [M]$, $X_m^{(j)} \in \mathbb{B}$ is true if the j-th uniform random walk of length n_{steps} in the digraph generated by $\xrightarrow[F_m/\mathcal{M}]{a1}$ from x^0 verifies the phenotype ϕ.

Evaluation objectives. For a mutation set $\mathcal{M}$ and an ensemble $\{F^{(m)}\}_{m=1}^{M}$, we estimate the mean success probability $\hat{p}_{\text{ens}}$ as follows:

$$\hat{p}_m = \frac{1}{n_{\text{traj}}} \sum_{j=1}^{n_{\text{traj}}} \mathbb{1}[X_m^{(j)} = 1], \quad \hat{p}_{\text{ens}} = \frac{1}{M} \sum_{m=1}^{M} \hat{p}_m \quad (1)$$

Intuitively, M averages over model diversity; n_{traj} reduces simulation noise per model. Both contribute to costs.

Search problem. Given depth D (number of mutations) and a time budget, our goal is to find $\mathcal{M} \in \mathcal{S}_D$ that maximizes $\hat{p}_{\text{ens}}$. Because $|\mathcal{S}_D| = \binom{g}{D} 2^D$ grows



(a) $\mathcal{M} = \{(1,1)\}$, start 100 (fixed). (b) $\mathcal{M} = \{(2,1)\}$, 010→011 (fixed). (c) $\mathcal{M} = \{(3,1)\}$, 001→101 (fixed).

Figure 3: **Asynchronous single-step trajectories on the toy BN (Figure 2).** Each panel shows the configuration cube under one gain-of-function mutation $\mathcal{M} = \{(i,1)\}$. Thick arrows mark the enabled asynchronous move; boxed states are fixed points.

rapidly, we later employ Nested Monte Carlo search with pruning and noise control, but here we formalize the modeling ingredients used throughout.

Experiments on biological case study In this work, we investigate the BN of Cohen et al. (2015) modeling molecular mechanisms involved in tumor progression. This model has been used to study mutations that impact metastasis development. The BN is composed of 32 nodes with 159 direct edges. We generated ensembles of Boolean networks using BoNesis (Chevalier et al., 2024), an Answer-Set Programming–based synthesis framework that samples diverse logical models sharing the same influence graph while satisfying user-specified dynamical constraints, following the work of Chevalier et al. (2020).

3 Variance Decomposition: Ensemble Diversity Dominates

Given a fixed ensemble size M, how many trajectories n per network are needed to estimate a target-phenotype probability accurately? Larger M averages model diversity and typically reduces how large n must be (See Figure 4).

3.1 Setup

Let $\mathscr{F} = \{F_m\}_{m=1}^{M}$ be an ensemble of Boolean networks on nodes $[n]$. Fix a simulation protocol (initial state or law, asynchronous updates, and a step cap n_{steps}). For phenotype ϕ, each model F_m has a success probability p_m: under the fixed protocol, it is the probability that a asynchronous update of the mutated model $F_m/\mathcal{M}$ reaches ϕ before the step cap n_{steps}.

We estimate p_m with n_{traj} independent simulations, Per-model and ensemble estimators $(\hat{p}_m, \hat{p}_{\text{ens}})$ are exactly those in Section 2.3, Equation (1).

Assumptions (i) Within each model, trajectories are i.i.d. given p_i; (ii) different models are independent given (p_i); (iii) models are sampled uniformly (with replacement) from $\mathscr{F}$.

3.2 Variance decomposition

Lemma 3.1. *Under the above assumptions,*

$$\mathrm{Var}(\hat{p}_{\text{ens}}) = \frac{\sigma_p^2}{M} + \frac{\eta}{Mn}, \qquad \eta := \mathbb{E}[p_i(1-p_i)], \;\; \sigma_p^2 := \mathrm{Var}(p_i).$$

Proof sketch. Law of total variance with $\mathrm{Var}(\hat{p}_i \mid p_i) = p_i(1-p_i)/n$ and $\mathbb{E}[\hat{p}_i \mid p_i] = p_i$.

Theorem 3.2 (minimal M)**.** *If* $\mathrm{Var}(\hat{p}_{\text{ens}}) \le \varepsilon^2$, *any* (M,n) *with*

$$M \;\ge\; \frac{\sigma_p^2 + \eta/n}{\varepsilon^2}$$

is sufficient. With only $p_i \in [0,1]$ *(hence* $\sigma_p^2 \le \frac{1}{4}$, $\eta \le \frac{1}{4}$*),*

$$M \;\ge\; \frac{\frac{1}{4} + \frac{1}{4n}}{\varepsilon^2}.$$

Corollary 3.3 (Fixed total budget $B = Mn$)**.** *With* B *fixed,*

$$M \;\ge\; \frac{\sigma_p^2}{\varepsilon^2 - \eta/B} \quad \textit{(requires } \varepsilon^2 > \eta/B\textit{).}$$

Remark 3.4 (Simulation-noise–only bound). Since $p(1-p) \le \frac{1}{4}$, enforcing $\eta/(Mn) \le \varepsilon_{\text{sim}}^2$ gives

$$n \;\ge\; \frac{1}{4M\varepsilon_{\text{sim}}^2}.$$

The $1/M$ term from *model diversity* (σ_p^2/M) often dominates once n is modest, so—under a fixed budget B—allocating more to M usually yields larger variance reductions than further increasing n.

4 Nested Monte Carlo Search on Boolean-Network Ensembles

We present three nested search algorithms—NMCS, NRPA, and BILNMCS—applied to ensembles of

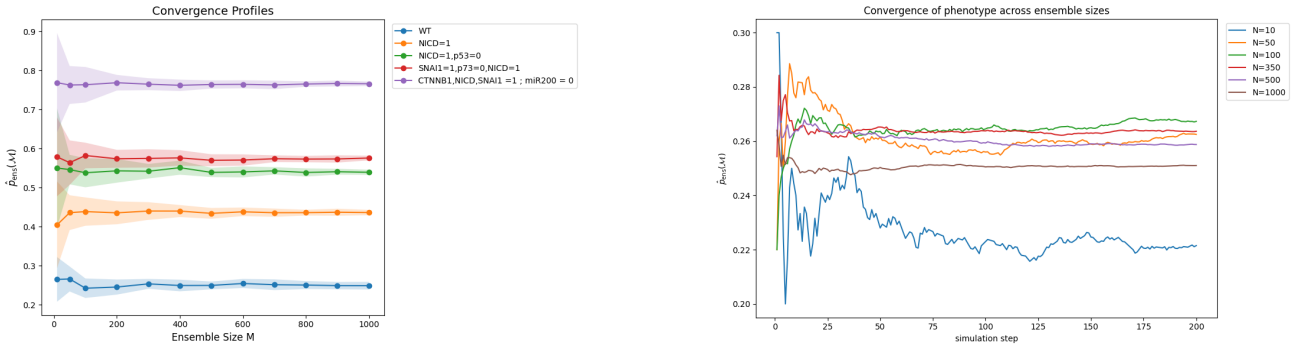


Figure 4: Convergence under ensemble simulations. Left: stability vs. trajectories per model n. Right: stability vs. ensemble size M.

Boolean networks with a fixed ensemble size M ($M = 1000$). All explore large mutation spaces via nested calls but differ in how they select and refine rollouts (playouts). Compared to the original versions, we add (1) Per-level caches (C_ℓ): at level L, save mutation sets with order-free keys to avoid re-evaluation. (2) Rollout Cache (`ec` results): store complete playout evaluations to skip identical simulations.

4.1 Nested Monte Carlo Search (NMCS)

Nested Monte Carlo Search(Cazenave, 2009) is a recursive algorithm originally used for single-player games and is here adapted to search over mutation sets in ensemble Boolean networks. At level 0, it runs a random rollout by picking mutations uniformly until the depth limit, returning the resulting score and mutation set. At higher levels, it tries each legal move from the current state, calls NMCS at one level lower, and keeps track of the best sequence found; after checking all moves, it advances along this best sequence and repeats the process. The key parameter is the nesting level, which balances search quality and computation. In our application, the state is the set of mutations applied so far, and NMCS scores each possible next mutation by recursive simulation to the target depth, reusing the best set found at each step to guide the search while still keeping some randomness for exploration.

In Algorithm 1, `lines 1--5` form a wrapper that sets the deadline T if a timeout is given, initializes the global best result $best = (\text{score} : -\infty, \text{set} : \emptyset)$, and prepares an empty cache C for evaluated states. `Line 6` calls the recursive `Core` function. Inside `Core`, `line 7` checks the deadline for anytime termination, and `line 8` converts S to a `frozenset` key, so caching is order-invariant. The base case in `lines 9--16` triggers when $L = 0$ or S is terminal: results are either retrieved from C or computed by a random playout, then cached and used to update *best* if improved. The recursive case (`lines 17--26`) starts by setting a local best, iterates over legal moves, rechecks T, generates S_1, and either uses the cache or calls `Core` at depth $L-1$; both local and global bests are updated if better scores are found. Finally, `lines 27--29` backtrack by extending S with the next move from *bestSet*, ensuring the search continues along the most promising path.

Algorithm 1: NMCS for BN

```
1  Function NMCS(S, L, D, bestMoves, ec, timeout):
2      T ← (now + timeout) or ∅
3      best ← {score : −∞, set : ∅}
4      C ← [empty map]_{ℓ=0}^{L}
5      return Core(S, L, D, bestMoves, ec, C, T, best)
6  Function Core(S, L, D, M, ec, C, T, best):
7      if T ≠ ∅ and now > T then return (best.score, best.set)
8      k ← frozenset(S)                                  ▷ canonical key
9      if k ∈ C[L] then return C[L][k]
10     if L = 0 then
11         if k ∈ C[0] then
12             (sc, mutSet) ← C[0][k]
13         else
14             (sc, mutSet) ← RandomPlayout(S, M, D, ec)
15             C[0][k] ← (sc, mutSet)
16         return (sc, mutSet)
17     bestSc ← −∞; bestSet ← ∅                          ▷ local best
18     while ¬ isTerminal(S) do
19         for m ∈ legalMoves(S, M) do
20             if T ≠ ∅ and now > T then return
                (best.score, best.set)
21             S_1 ← S ∪ {m};   k_1 ← frozenset(S_1)
22             if k_1 ∈ C[L−1] then
23                 (sc, mutSet) ← C[L−1][k_1]
24             else
25                 (sc, mutSet) ←
                    Core(S_1, L−1, D, M, ec, C, T, best)
                                                  ▷ recursive call
26                 C[L−1][k_1] ← (sc, mutSet)
27             if sc > bestSc then bestSc ← sc;
                bestSet ← mutSet
28             if sc > best.score then best ← (sc, mutSet)
                                                  ▷ global best
29         S ← S ∪ {next(bestSet \ S)}
30     return (bestSc, S)
```

4.2 Nested Rollout Policy Adaptation (NRPA)

Nested Rollout Policy Adaptation (Rosin, 2011) changes NMCS by replacing uniform rollouts with a learned stochastic policy

$$\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, \quad (s, a) \mapsto \pi(s, a),$$

where $\pi(s, a)$ is a log-weight assigned to action $a \in \mathcal{A}(s)$ in state s.

At the search level $L = 0$, actions are sampled according to the Gibbs distribution

$$P_\pi(a \mid s) = \frac{\exp\big(\pi(s, a)\big)}{\sum_{a' \in \mathcal{A}(s)} \exp\big(\pi(s, a')\big)}, \tag{2}$$

which biases selection toward higher-weight actions while retaining stochastic exploration.

After each level-L search, the policy is adapted toward the best sequence $B = (a_1, \ldots, a_D)$ found so far. Let s_t be the state before executing a_t. The update rule is

$$\begin{aligned} \pi \leftarrow \pi + \alpha \sum_{t=1}^{D} \Big[& e_{\mathrm{code}(s_t, a_t)} \\ & - \sum_{a \in \mathcal{A}(s_t)} P_\pi(a \mid s_t)\, e_{\mathrm{code}(s_t, a)} \Big] \end{aligned} \tag{3}$$

where e_i is the i-th standard basis vector and $\alpha > 0$ is the learning rate. Equation (3) is a stochastic gradient-ascent step on $\log P_\pi(B)$ under (2), integrating NMCS's recursive best-sequence propagation with online policy learning.

4.3 Pruning the candidates: Bi-level lazy-NMCS

In the section Section 2.2, the challenging part is that the $|\mathcal{S}_k|$ will explode, which makes the branch factor of a search very large.

Therefore, we propose an extension of Lazy Nested Monte Carlo Search that introduces a bi-level evaluation strategy to accelerate the search while maintaining high-quality solutions. This strategy builds on recent advances in Lazy NMCS (Roucairol and Cazenave, 2023; Roucairol et al., 2024), extending it with a decoupled evaluation that we refer to as **Bi-Lazy NMCS**.

The key idea is to decouple the pruning phase and the evaluation phase of the search by introducing two evaluators: (1)A cheap evaluator (`ec_fast`, e.g., an ensemble of size 50) used exclusively for the b random playouts in the pruning step (i) at each node. (2) A high-fidelity evaluator (`ec_main`, e.g., an ensemble of size 1000) responsible for scoring the final full mutation sets and updating the global. There are two Per-level caches equipped for those evaluator(potentially introduce memory issues).

In standard LNMCS, evaluating b rollouts per candidate move becomes costly when using large ensembles. Bi-Lazy NMCS addresses this by performing these inexpensive evaluations using a small ensemble, thus accelerating the screening phase.(see Algorithm 2) The pruning threshold θ_d at each depth d is then computed from these fast estimates, enabling the early elimination of weak branches. Final evaluations and best-set selection, however, are still performed using a large, accurate ensemble to preserve solution fidelity. This decoupled structure allows pruning decisions to be guided by small ensembles while retaining the quality benefits of larger ones—achieving a better time-quality tradeoff, particularly under tight computational budgets.

Algorithm 2: Bi-Lazy NMCS for BN

Function CachePlayout(C, ℓ, S, ec):
  $k \leftarrow \mathrm{Key}(S)$
  if $k \in C[\ell]$ **then**
    return $C[\ell][k]$
  $(sc, S^\star) \leftarrow$ RandomPlayout(S, M, D, ec); $C[\ell][k] \leftarrow (sc, S^\star)$;
  return $(sc, S^\star)$
Function FastStats(S):
  $d \leftarrow |S|$; $tot \leftarrow 0$
  for $i \leftarrow 1$ **to** b **do**
    $(sc, \_) \leftarrow$ CachePlayout(C^{fast}, 0, S, ec_{fast}); $tot \leftarrow tot + sc$
  $mean \leftarrow tot/b$; $tr[d].\mathrm{count} \leftarrow tr[d].\mathrm{count} + 1$
  $tr[d].\mathrm{mean} \leftarrow \dfrac{tr[d].\mathrm{mean} \cdot (tr[d].\mathrm{count} - 1) + mean}{tr[d].\mathrm{count}}$;
  $trmax[d] \leftarrow \max(trmax[d], mean)$;
  return mean
Function BiLazyNMCS(S, L, D, M, ec_{main}, ec_{fast}, e, b, r, tr, $trmax$, $timeout$):
  $T \leftarrow (\mathrm{now} + timeout)$ or $\varnothing$;
  $best \leftarrow (\mathrm{score} = -\infty,\ \mathrm{set} = \emptyset)$
  $C^{\mathrm{main}} \leftarrow [\,\mathrm{Map}()\,]_{\ell=0}^{L}$;
  $C^{\mathrm{fast}} \leftarrow [\,\mathrm{Map}()\,]_{\ell=0}^{L}$;
  return
    Core($S, L, D, M, ec_{\mathrm{main}}, ec_{\mathrm{fast}}, e, b, r, tr, trmax, C^{main}, C^{fast}, T$)
Function Core(S, L, D, M, ec_{main}, ec_{fast}, ...):
  if $T \neq \varnothing$ *and now*$> T$ **then return** $(best.score, best.set)$
  if $L = 0$ **then**
    return CachePlayout (C^{main},0,S,ec_{main})
  $k_0 \leftarrow \mathrm{Key}(S)$; $bestSc \leftarrow -\infty$; $bestSet \leftarrow [\,]$
  while $\neg\, isTerminal(S, D)$ **do**
    if $T \neq \varnothing$ *and now*$> T$ **then return** $(bestSc, S)$
    $\mathcal{M} \leftarrow$ LegalMoves(S, M); **if** $|\mathcal{M}| > e$ **then**
      $\mathcal{M} \leftarrow$ uniform sample of e from $\mathcal{M}$
    $\mathcal{C} \leftarrow [\,]$; $d \leftarrow |S|$
    for $m \in \mathcal{M}$ **do**
      $S' \leftarrow S \cup \{m\}$; $mean \leftarrow$ FastStats(S');
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(mean, m)\}$
    $\theta_d \leftarrow tr[d].\mathrm{mean} + r \cdot (trmax[d] - tr[d].\mathrm{mean})$
    for $(mean, m) \in \mathcal{C}$ **do**
      $S' \leftarrow S \cup \{m\}$; $k' \leftarrow \mathrm{Key}(S')$
      $nextL \leftarrow (0$ **if** $mean < \theta_d$ **else** $L-1)$
      if $nextL = 0$ **then**
        CachePlayout (C^{main},0,S',ec_{main})
      else
        $(sc,\ s') \leftarrow$ Core(S', $L-1$, ...)
        $C^{\mathrm{main}}[L-1][k'] \leftarrow (sc,\ s')$
      if $sc > bestSc$ **then** $bestSc \leftarrow sc$; $bestSet \leftarrow mutSet$
      if $sc > best.score$ **then** $best \leftarrow (sc, mutSet)$
    $S \leftarrow S \cup \{\mathrm{next}(bestSet \setminus S)\}$
  return $(bestSc, S)$

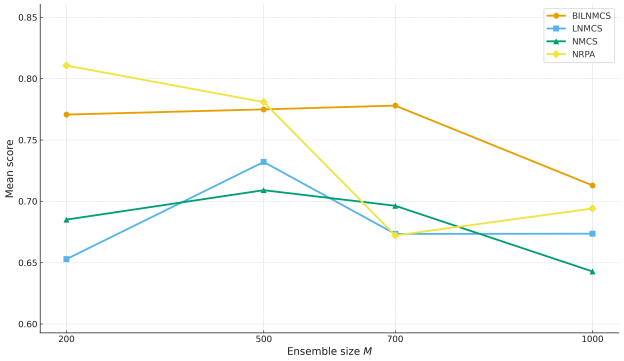


Figure 5: Scalability with ensemble size under a fixed 90 s timeout and depth $D=10$. Points show the mean score over 20 trials for each $M \in \{200, 500, 700, 1000\}$.

5 Experimental Evaluation: Scalability and Anytime Performance

We evaluate NMCS, LNMCS, NRPA, and our BILNMCS along two dimensions: (1) scalability and (2) anytime performance.

Experimental setup. Unless stated otherwise, all methods run at level 2. LNMCS/BILNMCS use $b=3$ playouts with pruning $r=0.4, e = 10$; BILNMCS decouples evaluation via a small-ensemble filter ($M=50$, `ec_fast`) and a full-ensemble scorer (M, `ec_main`). NRPA uses an untuned policy (implicit $\tau=1$, $N=100$). Using Remark 3.4, the simulation budget scales per model as $\lceil 2500/M \rceil$, keeping $\varepsilon_{\text{sim}} = 0.01$.

5.1 Scalability with Ensemble Size

We fix the timeout to 90 s and the search depth to $D=10$, and vary the ensemble size $M \in \{200, 500, 700, 1000\}$. Figure 5 summarizes the mean score over 20 trials.

BILNMCS is the most stable as M increases: it holds near 0.77 for $M=200$–700, dips slightly at $M=1000$. NRPA achieves the highest scores at small/medium M (peaking around 0.811 at $M=200$) but degrades to ≈ 0.69 at $M=1000$. LNMCS/NMCS is fairly flat (~ 0.65–0.7) across M.NMCS is the lowest at $M=1000$ reflecting the increase in evaluation costs when no pruning is used.

Under a fixed budget, growing M reduces the number of rollouts that an algorithm can afford. By decoupling fast screening of small ensemble (`ec_fast`) from high-fidelity confirmation on the full ensemble (`ec_main`), BILNMCS preserves stable and strong performance at large-M.

5.2 Anytime Performance

In Table 1, across depths $d \in \{4, 5, 6\}$ at fixed $M=1000$. For d = 4, BILNMCS is strongest at 10s, while NRPA leads at 30–60,s. For d = 5, BILNMCS wins at 30s, but NRPAis slightly ahead at 10s and 60s. For d = 6, BILNMCS attains the best mean across all shown budgets (10–60s). NMCS and LNMCS are generally behind these two. There is meaningful room to improve late-budget performance by organizing caches more carefully.

6 Conclusion

We studied how to search for mutation sets in Boolean-network (BN) ensembles when evaluations are expensive and noisy.

Limitations. There is a budget–accuracy trade-off. Small $M \times n$ is fast but noisy; large $M \times n$ is accurate but costly. Our current control uses a uniform worst-case bound for the simulation variance, which is conservative. Tighter, data-dependent estimates would better indicate when an estimate is "good enough," enabling smarter allocation between `ec_fast` and `ec_main`. Policy-adapted methods (NRPA/GNRPA) also have room for improvement (temperature schedules, bias design, and caching).

Future work. (i) *Variance-aware allocation:* use adaptive stopping with online variance estimates (with finite-population corrections) to adjust n_{traj} and M during the run and decide when to switch from `ec_fast` to `ec_main`. (ii) *Learning priors from the influence graph:* train a simple GCN to score nodes/moves and use it as bias β in GNRPA; then transfer the learned policy from small to large ensembles. (iii) *Broader case studies:* test on more BN ensembles to check scalability across variance regimes. (iv) *Benchmarking:* build an open benchmark (datasets, protocol, variance reporting, reproducible scripts) for fair comparisons and faster progress.

References

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Table 1: Anytime performance across depths $d \in \{4, 5, 6\}$ at M=1000 (trials = 10). Best per timeout in **bold**.

Depth	timeout (s)	NMCS	LNMCS	BILNMCS	NRPA
4	10	0.465 ± 0.079	0.505 ± 0.105	**0.576 ± 0.108**	0.566 ± 0.088
	30	0.615 ± 0.049	0.603 ± 0.062	0.600 ± 0.053	**0.634 ± 0.088**
	60	0.702 ± 0.065	0.698 ± 0.033	0.747 ± 0.046	**0.755 ± 0.050**
5	10	0.549 ± 0.079	0.549 ± 0.113	0.627 ± 0.141	**0.629 ± 0.040**
	30	0.627 ± 0.074	0.623 ± 0.118	**0.684 ± 0.129**	0.666 ± 0.091
	60	0.709 ± 0.052	0.701 ± 0.052	0.718 ± 0.069	**0.724 ± 0.045**
6	10	0.558 ± 0.072	0.613 ± 0.060	**0.619 ± 0.086**	0.577 ± 0.092
	30	0.664 ± 0.085	0.649 ± 0.049	**0.707 ± 0.102**	0.703 ± 0.066
	60	0.679 ± 0.082	0.716 ± 0.076	**0.729 ± 0.081**	0.725 ± 0.057

Tristan Cazenave. Nested monte-carlo search. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 456–461, 2009.

Stéphanie Chevalier, Vincent Noël, Laurence Calzone, Andrei Zinovyev, and Loïc Paulevé. Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision. In *CMSB 2020 - 18th International Conference on Computational Methods in Systems Biology*, volume 12314 of *Lecture Notes in Computer Science*, pages 193–209, Cham, 2020. Springer. doi: 10.1007/978-3-030-60327-4_11.

Stéphanie Chevalier, Déborah Boyenval, Gustavo Maga textasciitilde na López, Théo Roncalli, Athénaïs Vaginay, and Loïc Paulevé. BoNesis: a Python-based declarative environment for the verification, reprogramming, and synthesis of Most Permissive Boolean networks. In *22th International Conference on Computational Methods in Systems Biology (CMSB 2024)*, LNCS, Pisa, Italy, 2024. Springer. doi: 10.1007/978-3-031-71671-3_6.

Stéphanie Chevalier, Julia Becker, Yujuan Gui, Vincent Noël, Cui Su, Sascha Jung, Laurence Calzone, Andrei Zinovyev, Antonio del Sol, Jun Pang, Lasse Sinkkonen, Thomas Sauter, and Loïc Paulevé. Data-driven inference of Boolean networks from transcriptomes to predict cellular differentiation and reprogramming. *npj Systems Biology and Applications*, 11:105, 2025. doi: 10.1038/s41540-025-00569-z.

Daniel Paul Cohen, Giorgos Kourou, Ioana-Maria Ioana, Hidde de Jong, Evangelia S. T., et al. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLoS Computational Biology*, 11 (11):e1004571, 2015. Metastasis/invasion Boolean model used in many BN papers.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games (CG 2006)*, volume 4630 of *LNCS*, pages 72–83. Springer, 2007.

Sylvain Gelly and David Silver. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113, 2012.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML 2006)*, volume 4212 of *LNCS*, pages 282–293. Springer, 2006.

Aurélien Naldi, Céline Hernandez, Nicolas Levy, Gautier Stoll, Pedro T. Monteiro, Claudine Chaouiya, Tomáš Helikar, Andrei Zinovyev, Laurence Calzone, Sarah Cohen-Boulakia, Denis Thieffry, and Loïc Paulevé. The CoLoMoTo Interactive Notebook: Accessible and Reproducible Computational Analyses for Qualitative Biological Networks. *Frontiers in Physiology*, 9:680, 2018. doi: 10.3389/fphys.2018.00680.

Christoph D. Rosin. Nested rollout policy adaptation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

Milo Roucairol and Tristan Cazenave. Solving the hydrophobic-polar model with nested monte carlo search. In *Advances in Computational Collective Intelligence - 15th International Conference, ICCCI 2023, Budapest, Hungary, September 27-29, 2023, Proceedings*, volume 1864 of *Communications in Computer and Information Science*, pages 619–631. Springer, 2023.

Milo Roucairol, Jérôme Arjonilla, Abdallah Saffidine, and Tristan Cazenave. Lazy nested monte carlo search for coalition structure generation. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART 2024, Volume 2, Rome, Italy, February 24-26, 2024*, pages 58–67. SCITEPRESS, 2024.

Julian D. Schwab, Nensi Ikonomi, Silke D. Werle, Felix M. Weidner, Hartmut Geiger, and Hans A. Kestler. Reconstructing boolean network ensembles from single-cell data for unraveling dynamics in the aging of human hematopoietic stem cells. *Computational and Structural Biotechnology Journal*, 19:5321–5332, 2021. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2021.09.012.

Gautier Stoll, Eric Viara, Emmanuel Barillot, and Laurence Calzone. Continuous time boolean modeling for biological signaling: application of gillespie algorithm. *BMC Systems Biology*, 6(1):116, 2012. ISSN 1752-0509. doi: 10.1186/1752-0509-6-116.