Evolutionary Algorithms:
Convergence Time & Population Size

October 30, 2019

Prof. Chang Wook Ahn

Meta-Evolutionary Machine Intelligence (MEMI) Lab.
Electrical Eng. & Computer Sci.
Gwangju Inst. Sci. & Tech. (GIST)

Convergence

Time from

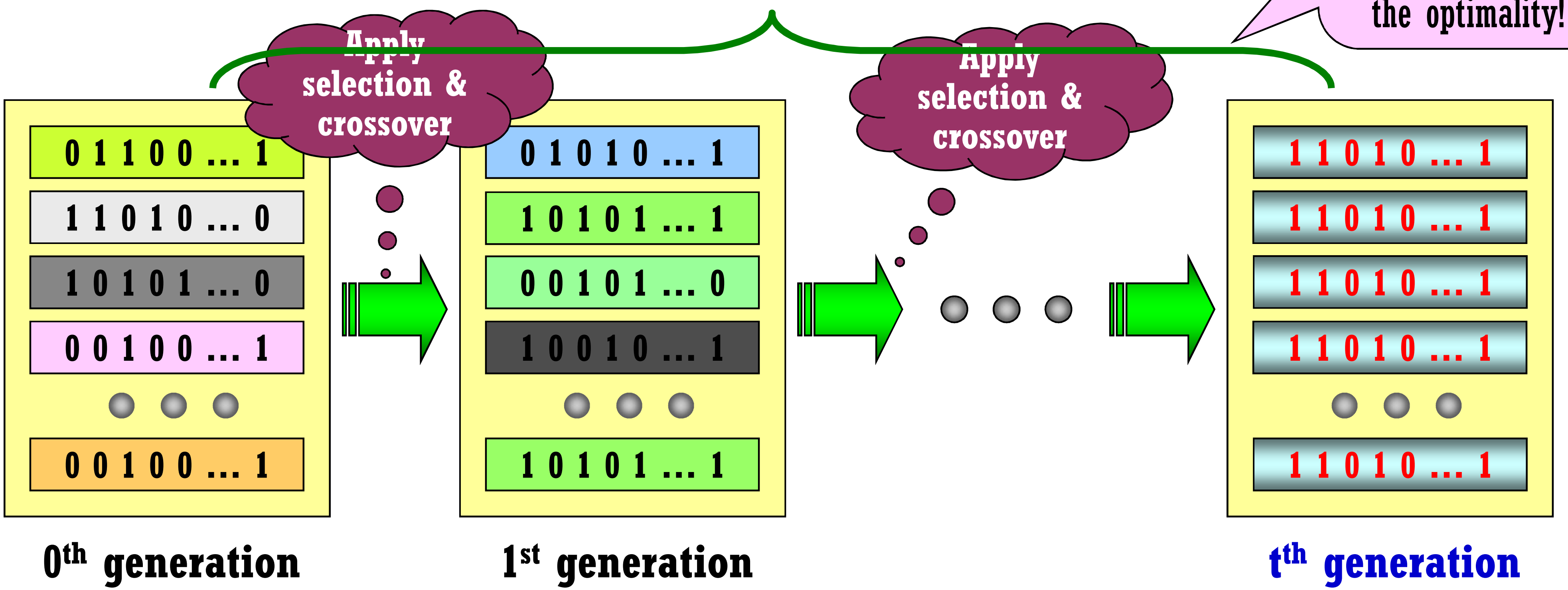
Population genetics

Convergence Time (1)

❖ Convergence Time

- ➢ Time (i.e., number of generations) until the population is converged.
- ➢ Assumptions
  - ✓ Two alternatives: 0 and 1
  - ✓ OneMax problem is assumed.
  - ✓ Uniform crossover is used, and no mutation is employed.

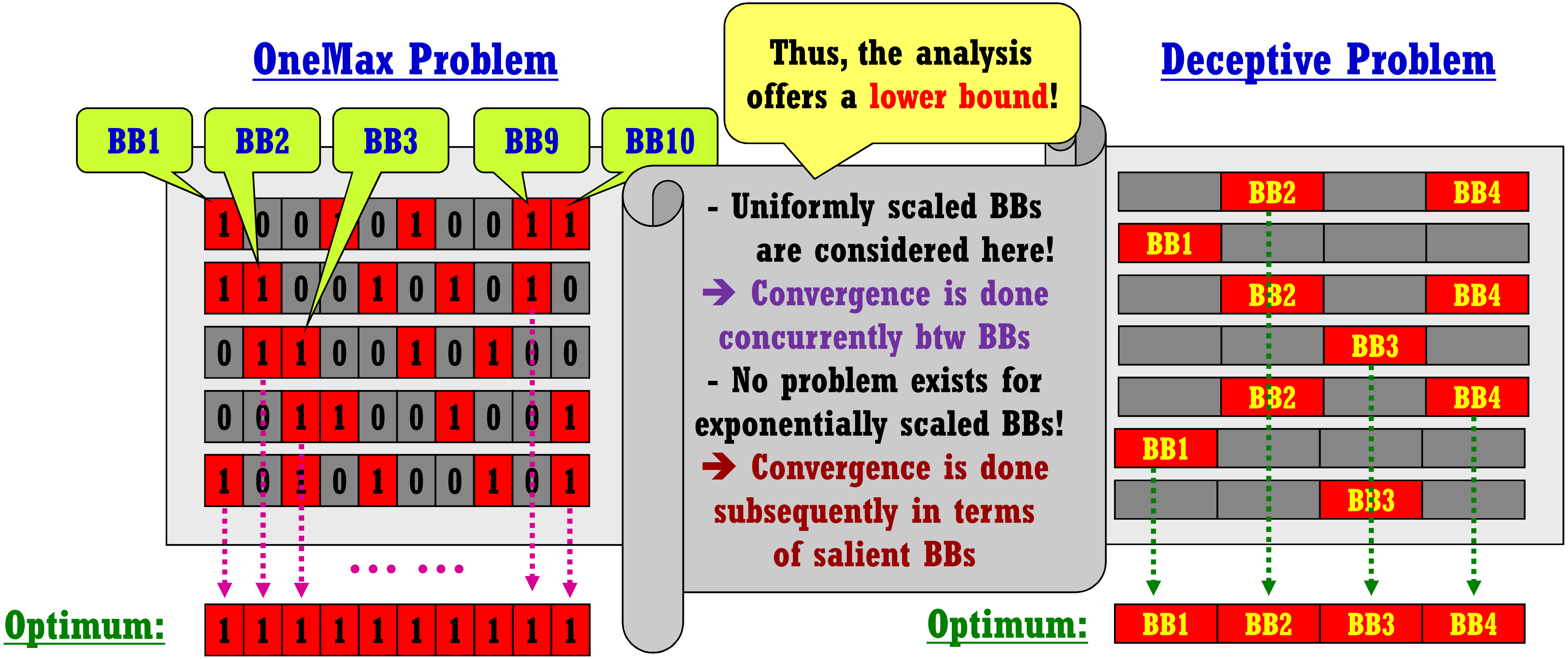
Convergence Time: But it does not say anything for the optimality!



Convergence Time (2)

❖ Why are the Assumptions Reasonable?

- If we successfully discover BBs, any problem can be interpreted as OneMax Problem at the level of BBs
- Otherwise, it offers useful bounds of how quickly solutions are expected to take



Convergence Time (3)

❖ Fisher's Theorem & Proportional Selection

- We calculate the **change in expected fitness** in a population as a function of the **current average fitness** (μ_t) and **the fitness variance** (σ_t^2) as follows:

$$\mu_{t+1} - \mu_t = \sigma_t^2 / \mu_t$$

❖ Convergence Time for the OneMax problem

OneMax Problem
N: Pop. size
n: Indiv. length
$\mu_t = n \cdot p_t$
$\sigma_t^2 = n \cdot p_t \cdot (1 - p_t)$

- The Fisher's theorem yields a **linear difference** equation

$$p_{t+1} - p_t = n^{-1}(1 - p_t)$$

- The **exact solution** to the equation is given as follows:

$$p_t = 1 - (1 - p_0)(1 - n^{-1})^t$$

- By an **approximation** of $(1 - r/k)^k \cong e^{-r}$, we obtain the following

$$p_t = 1 - (1 - p_0)\exp(-t/n)$$

- Getting to $p_t = 1 - \varepsilon$, the **convergence time** is given as follows:

$$t_c = -n \ln \frac{\varepsilon}{1 - p_0} = O(n)$$

Convergence Time (4)

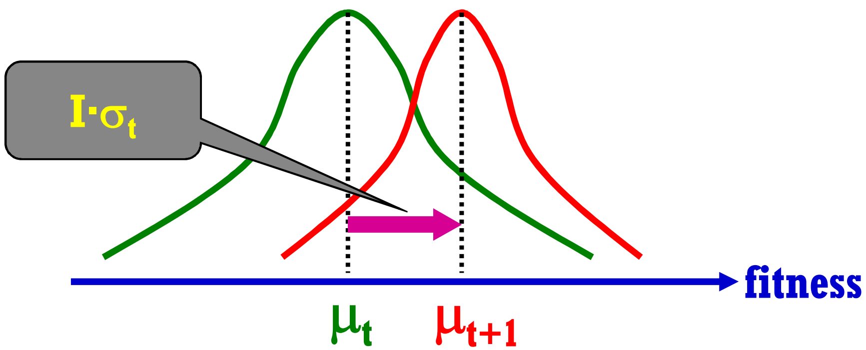
❖ Selection Intensity & Ordinal Selection

- **Ordinal selection** makes selections based on **the ranking** in the population
  - ➔ we need a somewhat **different analysis** for such schemes
- The **distribution** of fitness values may often be assumed to be **Gaussian**.
- Ordinal selection may be envisioned as **truncating the Gaussian distribution** and **shifting the fitness** to a somewhat higher value in the next generation.
- **The shift** is measured with the **fitness variance** σ_t^2 and the **selection intensity** I

$$\mu_{t+1} = \mu_t + I \cdot \sigma_t$$

Note: Different selection schemes have **different I values** as a function of their parameters

- **E.g.,** the **s-wise tournament selection** has the following selection intensity:



$$I = \mu_{s:s} = s\int_{-\infty}^{\infty} x\phi(x)\left(\int_{-\infty}^{x}\phi(z)dz\right)^{s-1} dx$$

$$I \approx \sqrt{2(\ln s - \ln\sqrt{4.14\ln s})}$$

Convergence Time (5)

❖ Convergence Time for the OneMax Problem

- The previous equation yields a difference equation:

$$p_{t+1} - p_t = I \Big/ \sqrt{n \cdot p_t \cdot (1 - p_t)}$$

- It can be approximated by a differential equation, and yields the following:

$$p_t = \frac{1}{2}\left[1 + \sin\left(\frac{I}{\sqrt{n}} t - \arcsin(2p_0 - 1)\right)\right]$$

- The time to full convergence (i.e., p_t=1.0) can be obtained as follows:

$$t_c = \left(\frac{\pi}{2} - \arcsin(2p_0 - 1)\right)\frac{\sqrt{n}}{I}$$

- For the initial condition, i.e., randomly generated individuals,
  - ✓ p_0=0.5 and thus the arcsin term is zero

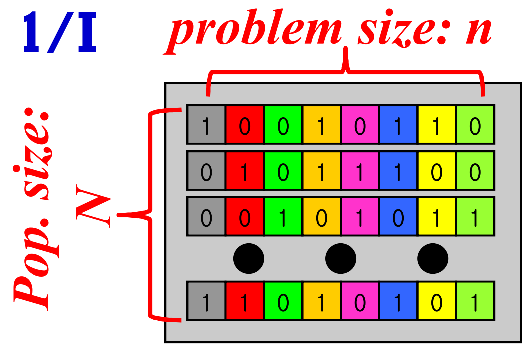
$$t_c = \frac{\pi}{2}\frac{\sqrt{n}}{I} = O(\sqrt{n})$$

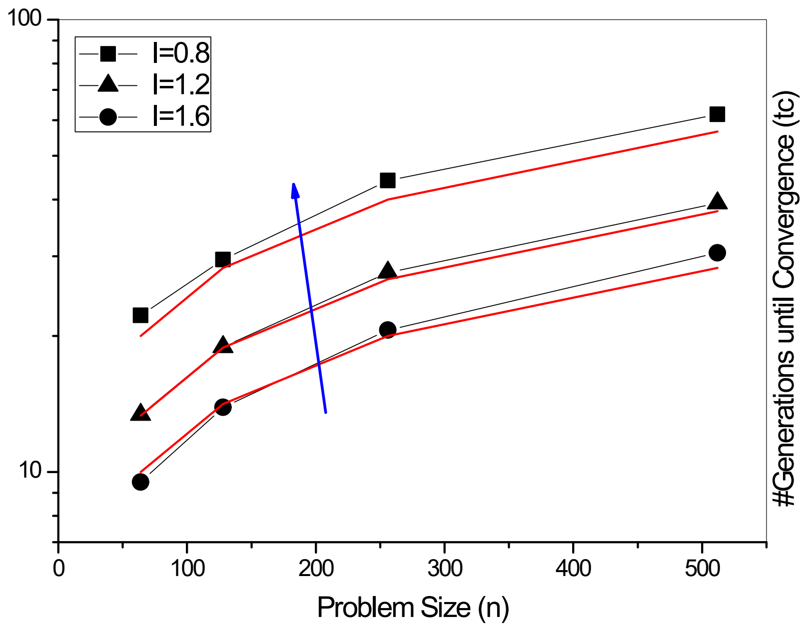
OneMax Problem
N: Pop. size
n: Indiv. length
$\mu_t = n \cdot p_t$
$\sigma_t^2 = n \cdot p_t \cdot (1-p_t)$

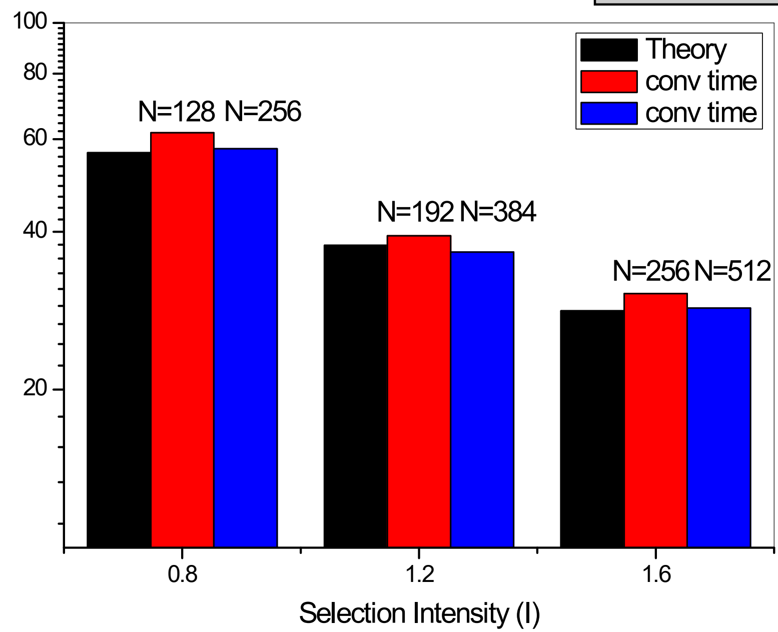
Convergence Time (6)

❖ Experimental Investigation

- Theoretical model is well matched with Experimental results
- Convergence time(tc) is proportional to sqrt(problem size); tc ∝ sqrt(n)
- tc is inversely proportional to the selection intensity I; tc ∝ 1/I
- tc is nearly independent of the population size N;
  - ✓ It implies that parallelism is embedded in GA!







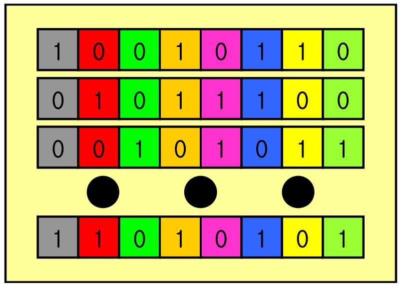
Population
Size of GAs

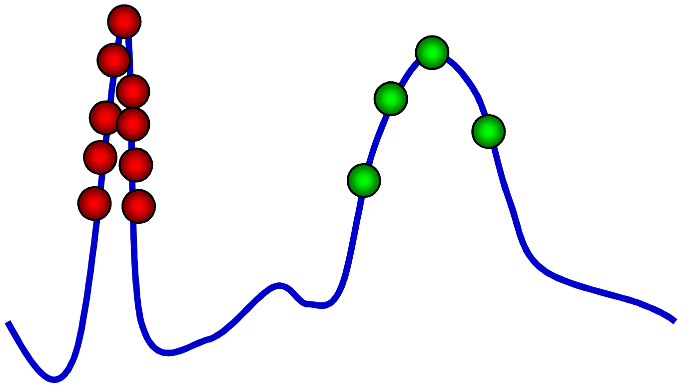
Population Size (1)

❖ Population Size of GA

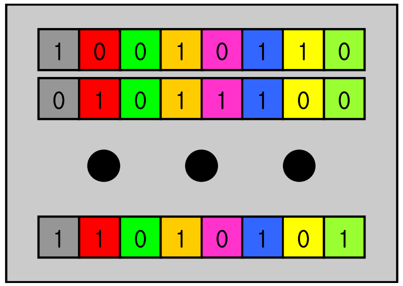
- ➢ How many individuals do we need to discover the solution of interest?
- ➢ Why important? We can optimize the solution quality and the computation cost.
- ➢ Assumptions: similar to those in the convergence time analysis
  - ✓ OneMax-type problem is considered.
  - ✓ Pair-wise tournament selection is used.
  - ✓ Uniform crossover is used, and No mutation is employed.

Large Population





Small Population



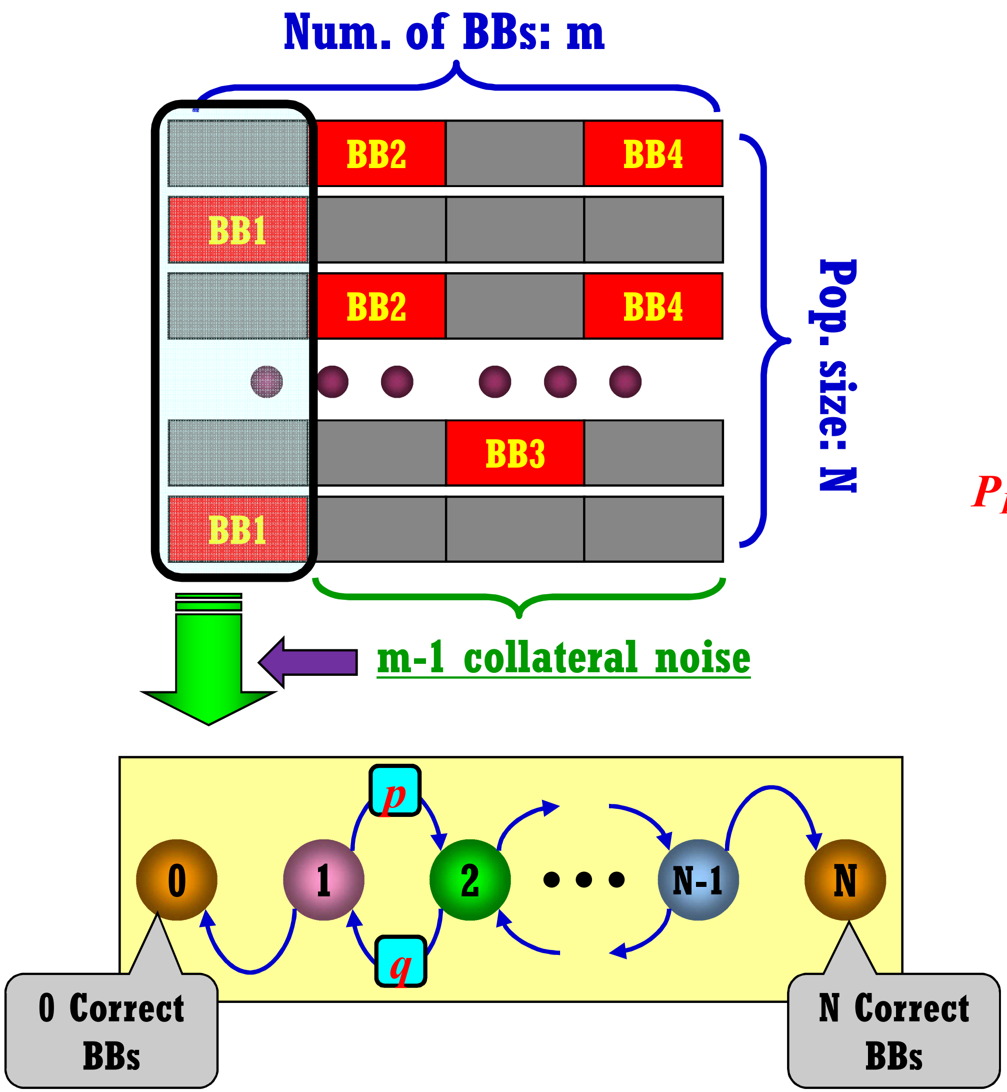
- It can discover the optimum
- But, it wastes computation cost

- It uses small computation cost
- But, it cannot find the optimum

Population Size (2)

❖ Gambler's Ruin Model

➢ Population behavior of GA can be represented by the Gambler's ruin model



- P_{BB} is the prob. that population corresponding to each BB is successfully converged
- $P_{BB}(i)$ is the prob. that the population starting from the i-th state (i.e., i correct BBs) is converged.
- $P(i)$ is the prob. that the population has i correct BBs.
- k is the number of bits of each BB.

$$P_{BB} = \sum_{i=0}^{N} P_{BB}(i)\, P(i) = \sum_{i=0}^{N} \left[\frac{1-(q/p)^i}{1-(q/p)^N} \right] \binom{N}{i} \left(\frac{1}{2^k}\right)^i \left(1-\frac{1}{2^k}\right)^{N-i}$$

$$\Rightarrow \quad P_{BB} = \frac{1-\left(1-\frac{2p-1}{2^k p}\right)^N}{1-(q/p)^N}$$

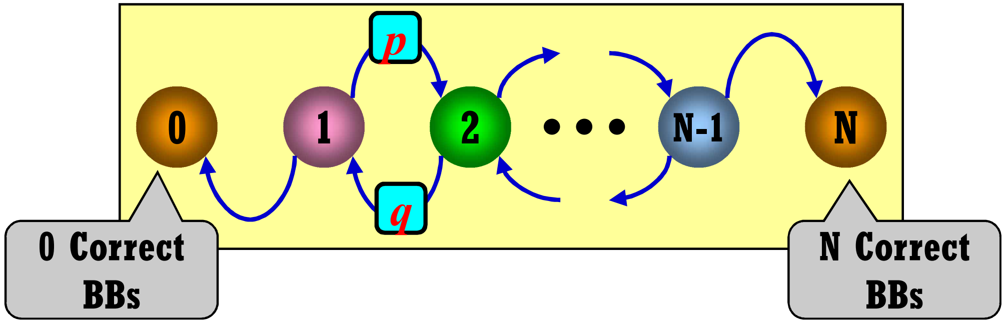
$$\Rightarrow \quad N = \frac{\ln(1-P_{BB})}{\ln\left(1-\frac{2p-1}{2^k p}\right)} \approx -2^k \ln(\alpha)\ \frac{p}{2p-1}$$

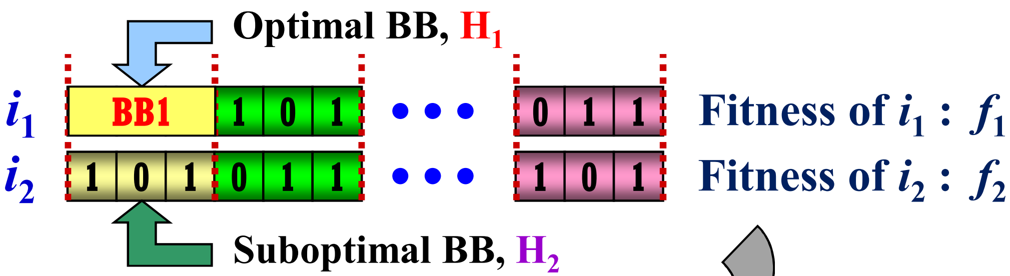
where $\alpha = 1 - P_{BB}$ (i.e., failure probability)

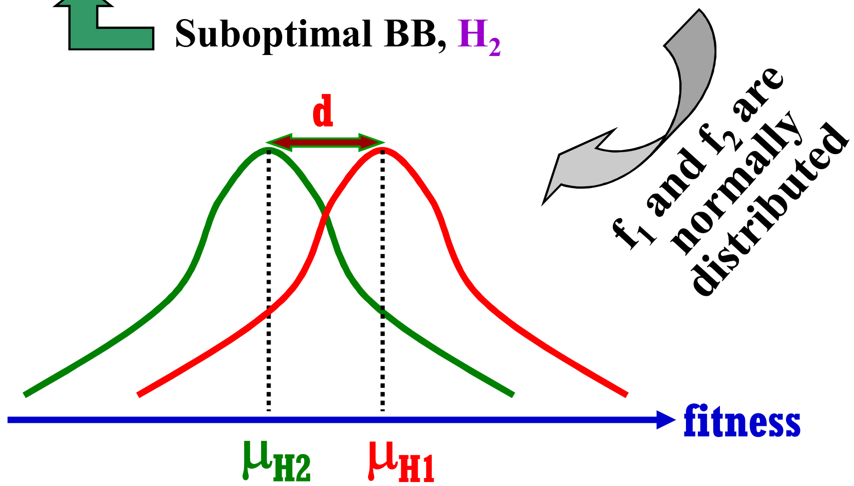
Population Size (3)

❖ Decision Making Model

- The state transition prob. of **GA** can be represented by the **decision making model**







$f_1 \sim \mathrm{N}(\mu_{H1}, \sigma_{H1}^2)$ $\quad$ $f_2 \sim \mathrm{N}(\mu_{H2}, \sigma_{H2}^2)$

$p = P[H_1 \text{ propagates}] = P[f_1 > f_2] = P[f_1 - f_2 > 0]$

Since f_1 and f_2 are normally distributed, $f_1 - f_2$ is also normally distributed with with mean $\mu_{H1}-\mu_{H2}$ and variance $\sigma^2_{H1}+\sigma^2_{H2}$

$$p = \Phi\left(\frac{\mu_{H1}-\mu_{H2}}{\sqrt{\sigma_{H1}^2+\sigma_{H2}^2}}\right) = \Phi\left(\frac{d}{\sqrt{2(m-1)}\sigma_{BB}}\right)$$

By the approximations, $p = \frac{1}{2} + \frac{1}{\sqrt{2\pi}}\frac{d}{\sigma_{BB}\sqrt{2(m-1)}}$

$$N = -2^{k-1}\ln(\alpha)\left(\frac{\sigma_{BB}\sqrt{\pi(m-1)}}{d} + 1\right)$$

Population Size (4)

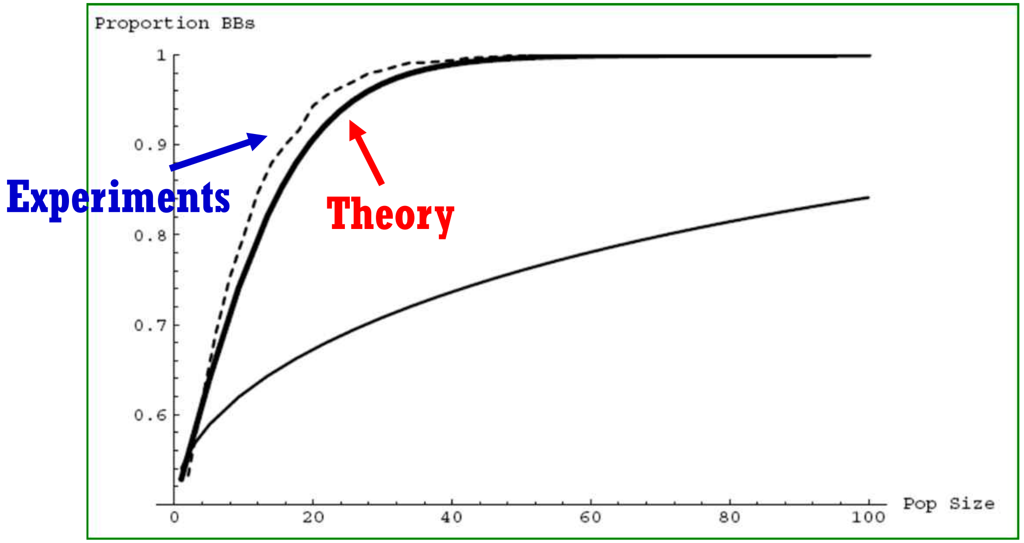
#BBs (i.e., m) is proportional to problem size (i.e., individual length n)!

❖ Experimental Verification

- Theoretical results agree to experiments quite well.
- Pop. size(N) is proportional to sqrt(m)
- Pop. size(N) is proportional to BB noise, σ_{BB}

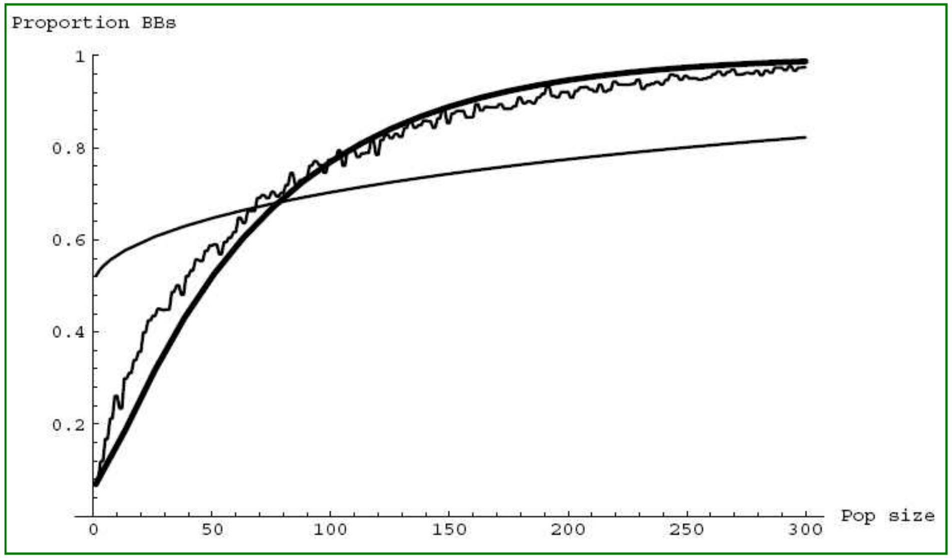
$$N = -2^{k-1}\ln(\alpha)\left(\frac{\sigma_{BB}\sqrt{\pi(m-1)}}{d}+1\right)$$

Results for OneMax Problem (100 bits)



Pairwise tournament selection
Uniform crossover

Results for 4-bit Deceptive Problem (80 bits)



Pairwise tournament selection
2-point crossover

Automatic Population Size (1)

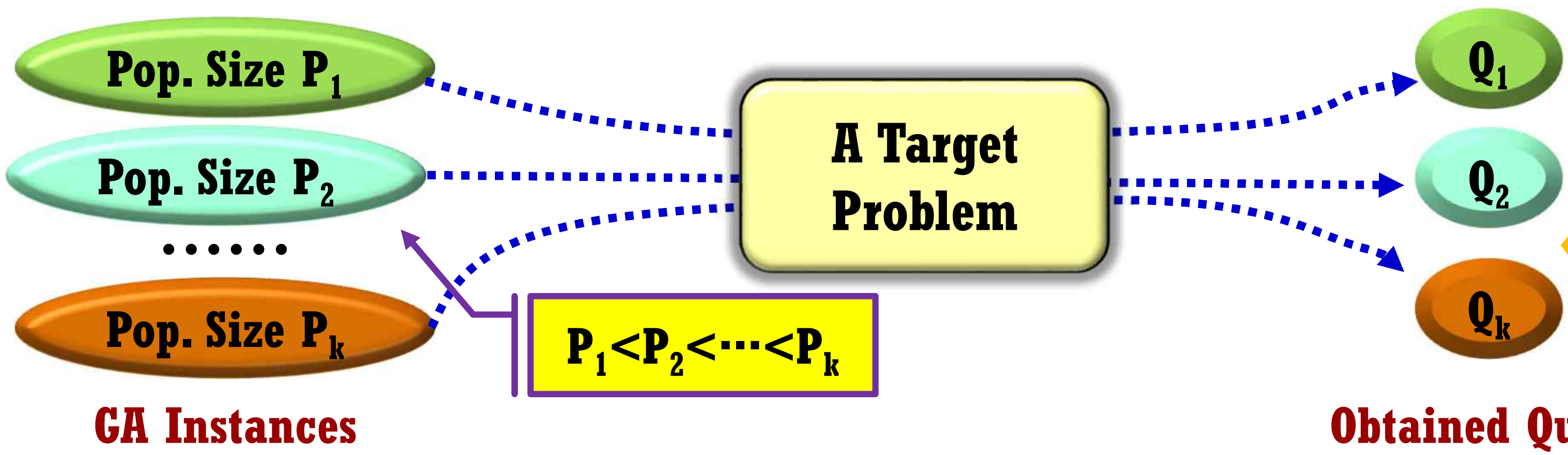
- ❖ **What's Wrong with the Theoretical Population-Sizing Model?**
  - ➢ The model says that population size should be $\propto$ sqrt(problem length) and BB's SNR ratio
  - ➢ But the model is difficult to apply in practice
    - ✓ Due to several assumptions that may not hold for real problems

$$N = -2^{k-1}\ln(\alpha)\left(\frac{\sigma_{BB}\sqrt{\pi(m-1)}}{d}+1\right)$$

How to estimate?

- ❖ **What are Users actually doing to solve a problem with a GA?**
  - ➢ Start by trying a small population size and then try a larger one…
  - ➢ In the end, user has tried several different population sizes to have a feeling of
    - ✓ how the problem responds to different population settings

➔ **Why not design the algorithm that does this steps automatically?**



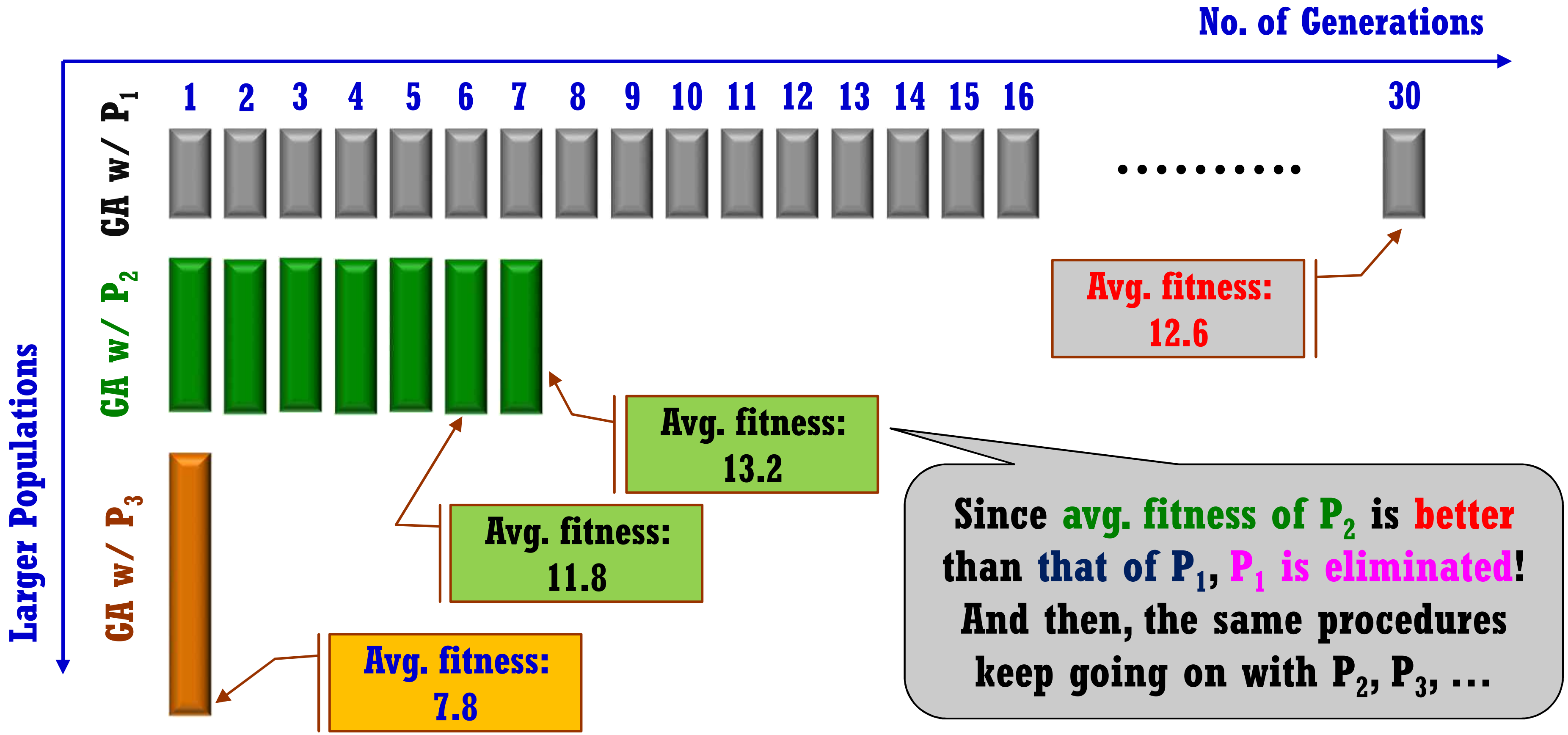
If the results with bigger populations are similar to those with smaller ones, take the best solution!

Automatic Population Size (2)

- **How about Establishing a Race among multiple populations?**
  - It gives smaller populations more function evaluations, thereby converging faster
  - If a larger population has an average fitness better than that of a smaller one
    - ✓ The smaller population is eliminated due to no reason in continuing it.

Counter

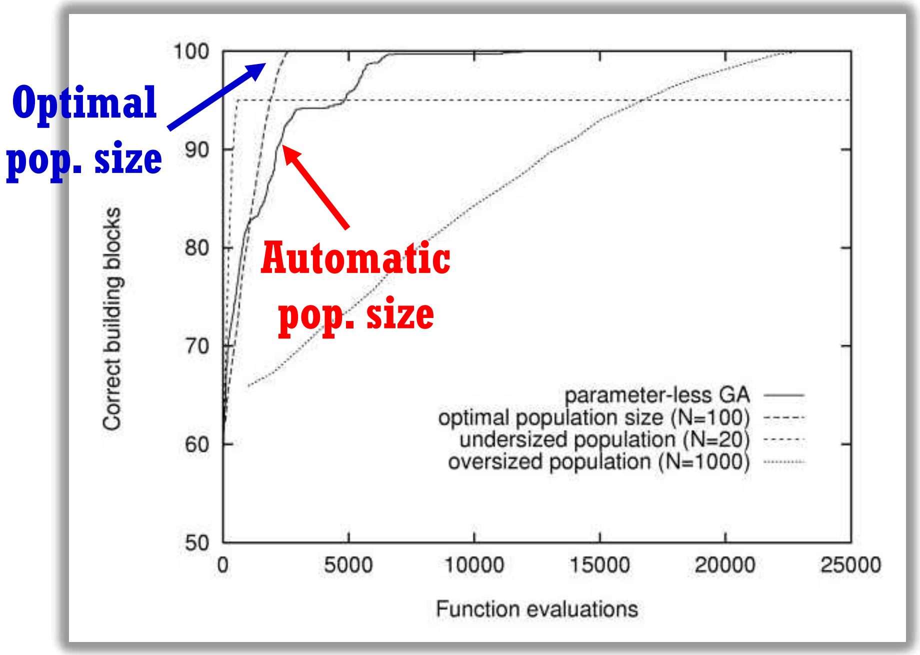
0 → -
1 → P_1
2 → P_1
3 → P_1
10 → P_2
11 → P_1
12 → P_1
13 → P_1
20 → P_2
21 → P_1
22 → P_1
23 → P_1
30 → P_2
31 → P_1
32 → P_1
33 → P_1
100 → P_3
101 → P_1
....



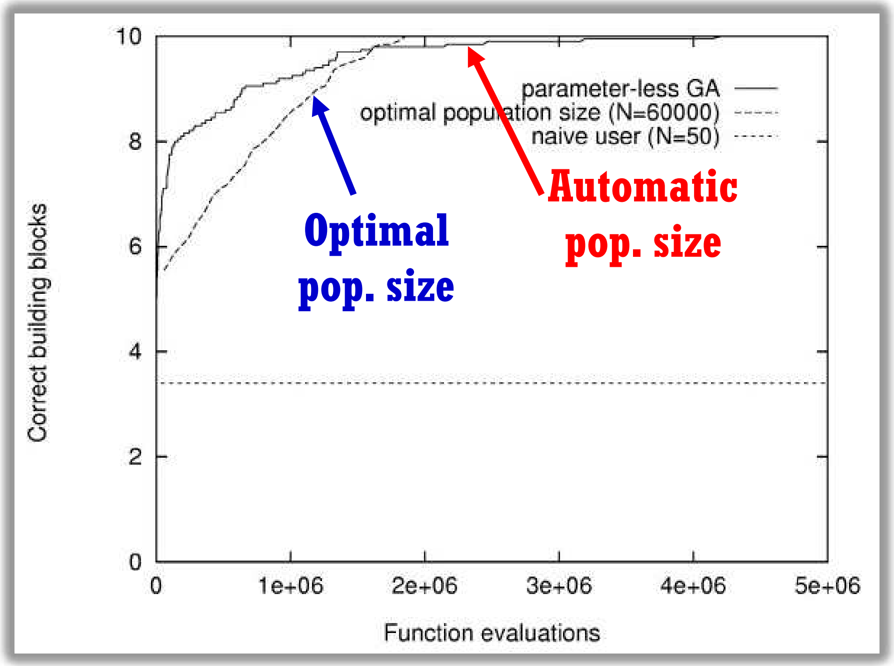
Automatic Population Size (3)

- **Experimental Verification**
  - The automatic population-sizing scheme does not require much computing resources comparing to the optimal population setting.
    - If problem is harder, their performances are similar

Results for OneMax Problem (100 bits)



Results for 4-bit Deceptive Problem (40 bits)



Summary

- **Two Issues** of GAs have been investigated!
  - **Convergence Time**; i.e., #generations until the population is converged
  - **Population Size**; i.e., #individuals required for the specific quality of solution
- **Convergence Time** (i.e., #generations until convergence)
  - Proportional selection: $O(n)$, Ordinal selection: $O(\sqrt{n})$ where n is the problem size
  - It is inversely proportional to the selection intensity
  - It is not dependent on the population size! ➔ Parallelism
- **Population Size** (i.e., #individuals to get a desired quality of solution)
  - It is proportional to sqrt(problem size)
  - It is inversely proportional to signal-to-noise ratio

When running GAs, we can set the population size required for obtaining a certain quality of solution, and estimate its required computing costs!