Estimating the Number of Local Optima in Multimodal Pseudo-Boolean Functions: Validation via Landscapes of Triangles

Xavier F. C. Sánchez-Díaz, Ole Jakob Mengshoel

Department of Computer Science
Norwegian University of Science and Technology

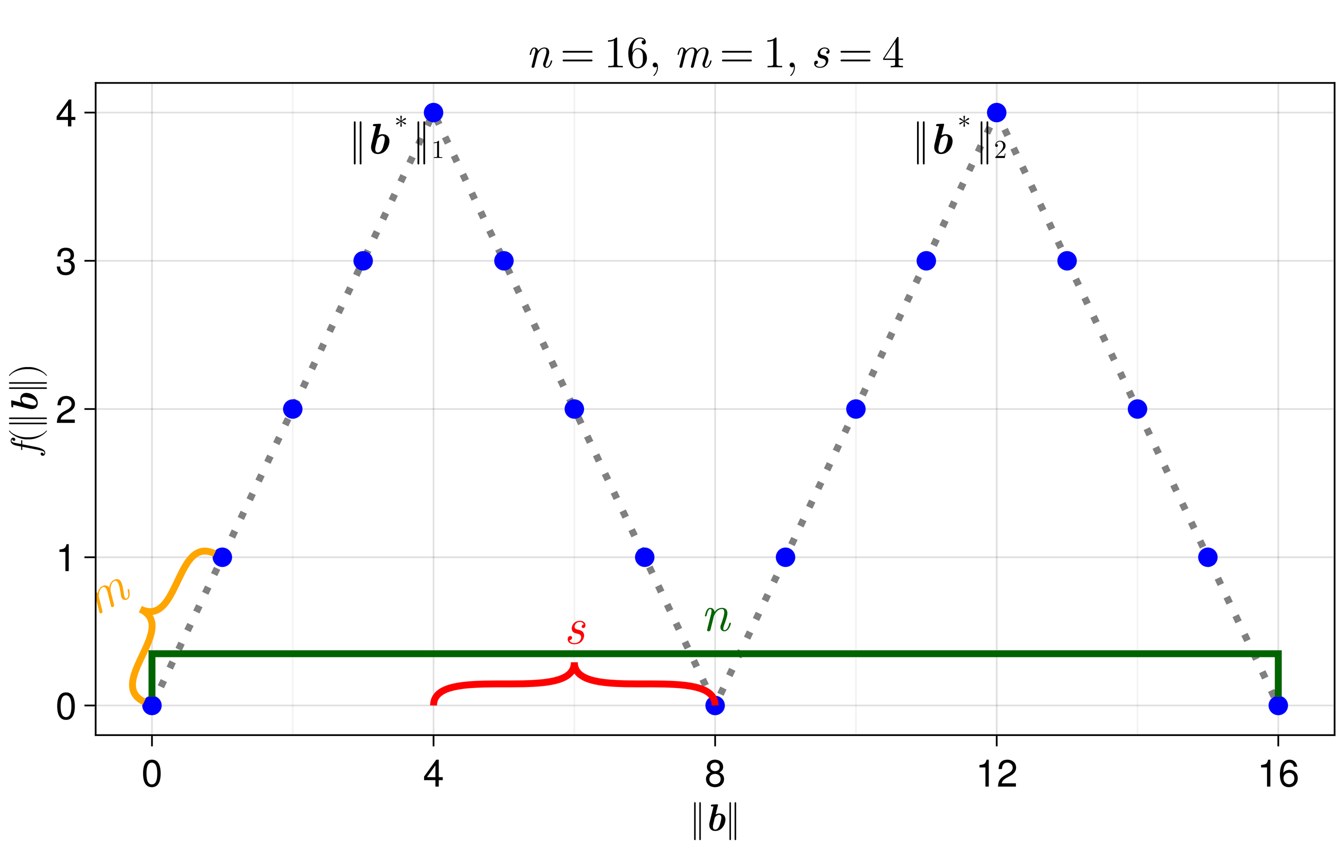
xavier.sanchezdz@ntnu.no, ole.j.mengshoel@ntnu.no

Abstract

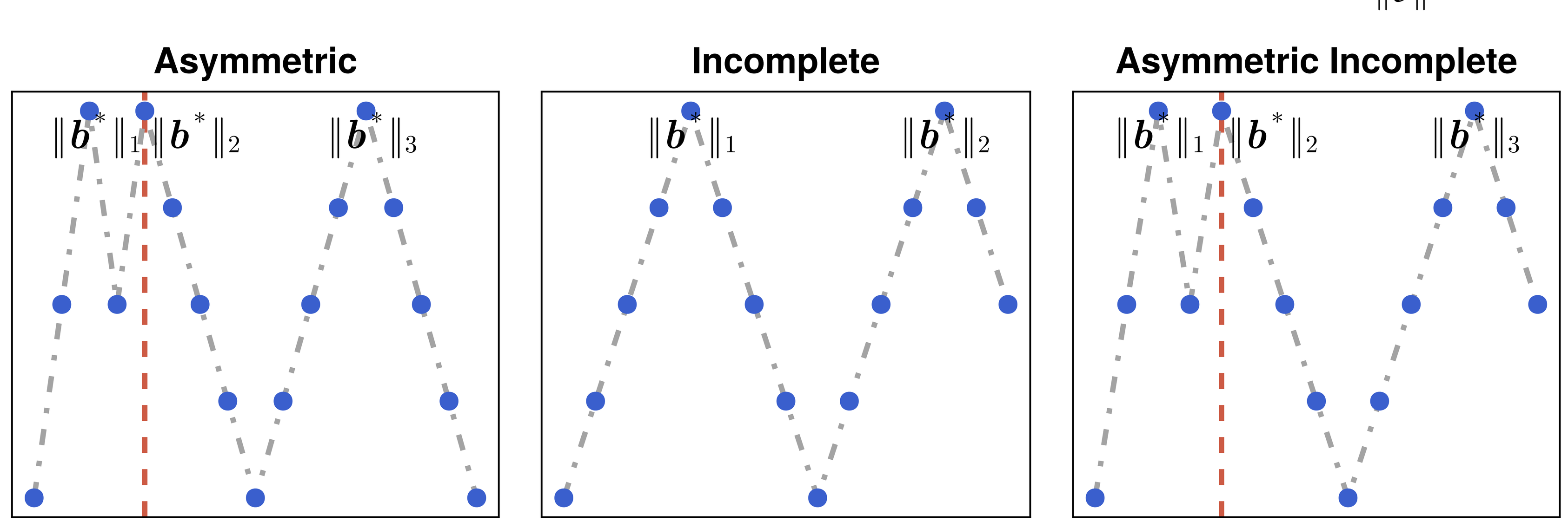
Pseudo-Boolean functions are often **multimodal**, and it is of interest to find **multiple optima**. However, the problem of **estimating the number of local optima** has not been much studied in the **combinatorial setting**. Since exhaustive enumeration is generally prohibitive, we study an alternative in this paper. Our method, which uses the celebrated **Birthday Paradox** "in reverse", enables us to estimate the number of local optima in fitness landscapes. We study the method analytically and experimentally, using a **new synthetic problem**, TRIANGLE. This problem allows us to vary the **number of optima and its distribution** easily but understandably, which enables **analytical validation** of our experiments.

A visual example: TRIANGLE

We focus on **Peaked-Local OPtima** (PLOP) functions where an optima $\boldsymbol{b}^* \in \mathbb{B}$ is strictly better than its neighbourhood. We create a PLOP function based on three parameters, n, m and s, that we can tune to control the number of optima. **Fig. 1** shows an example of this function, TRIANGLE with $n = 16, m = 1$ and $s = 4$.



(a) A symmetric TRIANGLE function with a bitstring length n, a slope m and a step size s. The number of optima for different values of $\|\boldsymbol{b}\|$ can be calculated as $\binom{n}{\|\boldsymbol{b}\|}$.



(b) Possible asymmetric landscapes

Figure 1: The TRIANGLE function assigns fitness to a bitstring $\boldsymbol{b}$ given the number of 1s in its chromosome. Different combinations of the n, m and s parameters give us different landscapes and different number of optima. In symmetric landscapes (cf. 1a) the number of optima is distributed evenly, as opposed to in asymmetric landscapes where two regions can be identified (separated by a red, dashed line in 1b).

Sampling the space of bitstrings

Using a **hill-climbing evolutionary algorithm** (HCEA) we sample the space:

1. Initialise **set of samples**
2. **Find a local optimum** via hill-climbing
3. **Store** it in the set
4. **Find another local optimum**
5. **If already found**: then it is a **repeated birthday**. Otherwise, continue searching.
6. **Record** how many tries it took to find the same optimum twice

We repeat this process until we collect a sample of size N to **estimate the number of optima.**

The Birthday Paradox

"How many people in a room (r) do we need to get two people sharing the same birthday with $p = 0.5$?"

After a certain sample of size N, we use the **reverse Birthday Paradox**:

$$d = \frac{(r - 0.5)^2 - 0.25}{2 \log(2)}$$

to **estimate the number of optima** ("*number of days in a calendar year*" or d) from the number of samples ("*people in the room*").

Experiments

We tested different landscapes of TRIANGLE, both **symmetric** and **asymmetric**.

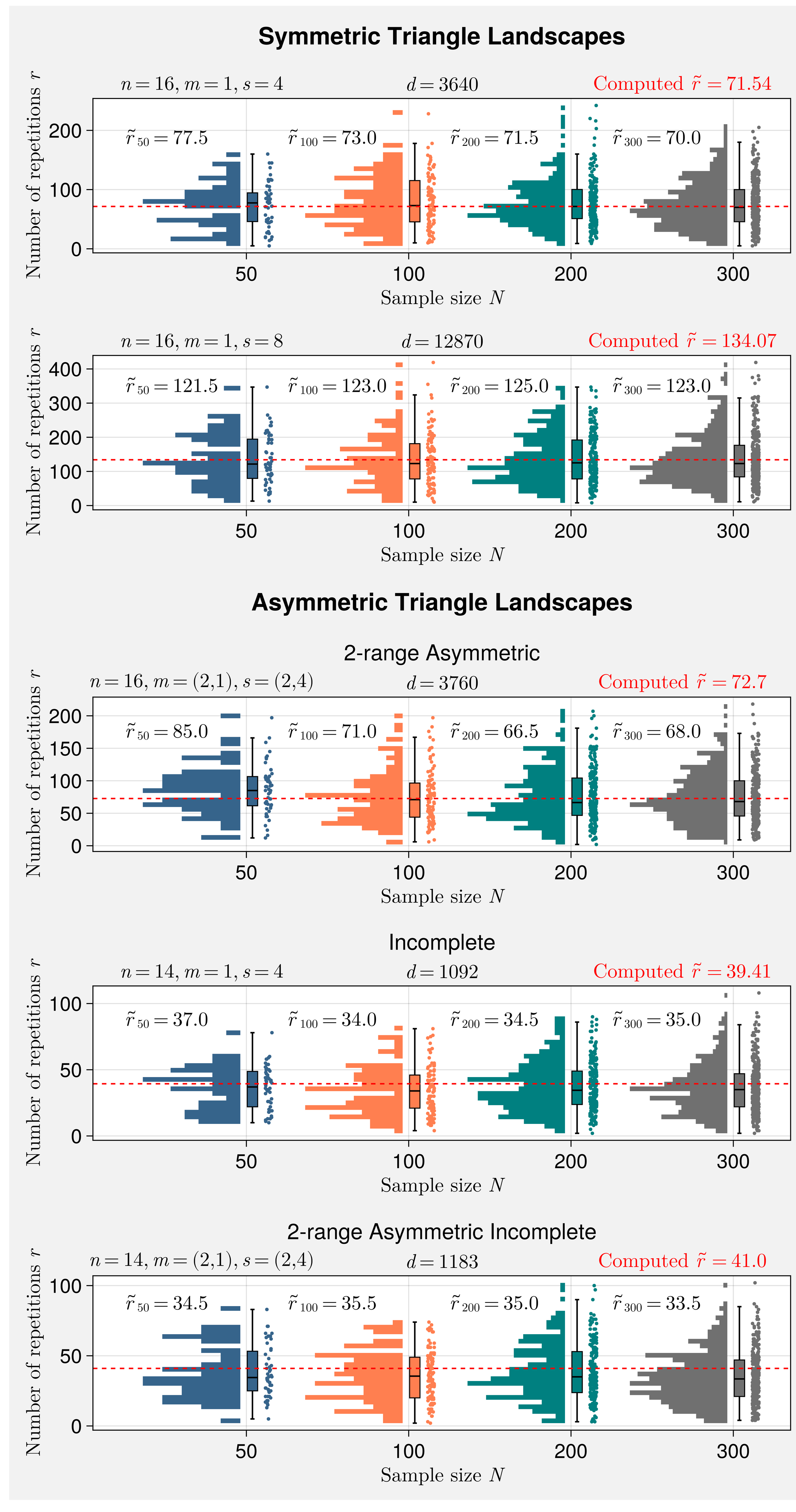


Figure 2: Sampling results on different landscapes. For each trial, we present a histogram, a box plot and a scatter plot of the set of samples. The median $\tilde{r}$ is shown with a red dashed line.

Findings and continuing the work

The estimation procedure **gives very accurate estimates in symmetric landscapes** where regions of attraction are approximately of the same size. However, the estimation is **not as accurate on asymmetric incomplete landscapes**. Still, the estimate falls inside the IQR and **provides a good sense of the number of optima** given the small number of samples taken. This is **useful** when dealing with **highly multimodal landscapes**, or when **finding all optima** is desirable.

Some avenues for future work include improving the estimation procedure by trying out **other sampling methods**, and testing on **real-world landscapes** is another opportunity to build on this work.

Read the full paper

The Genetic and Evolutionary Computation Conference
July 14–18, 2024, Melbourne, Australia

Coded with EvoLP.jl