Examining Forms of Inductive Bias Towards 'Simplicity' in Genetic Algorithms to Enhance Evolvability of Boolean Functions

Hetvi Jethwani [1] Sumeet Agarwal [1]

[1]Indian Institute of Technology, Delhi

Background

One key factor in enabling evolvability appears to be some form of prior constraint or *inductive bias* on the search space, which typically represents selectivity for low-complexity structures- one can think of these constraints as approximations to physical laws which drive evolution. **We propose a common simulation framework to examine how two forms of inductive bias might relate to each other in terms of their facilitation of evolution, and whether they might be unified.**

Learning theoretic setting

Valiant [3] suggests that Darwinian evolution can be formulated as a learning problem, where learning is only guided by aggregate fitness. Restricting the search space to a **chosen class of functions** (such as conjunctions) here serves as a form of *inductive bias*. This allows him to obtain results on which function classes are evolvable in polynomial time and resources. Simpler classes like conjunctions are shown to be more evolvable than less simple classes like parity functions.

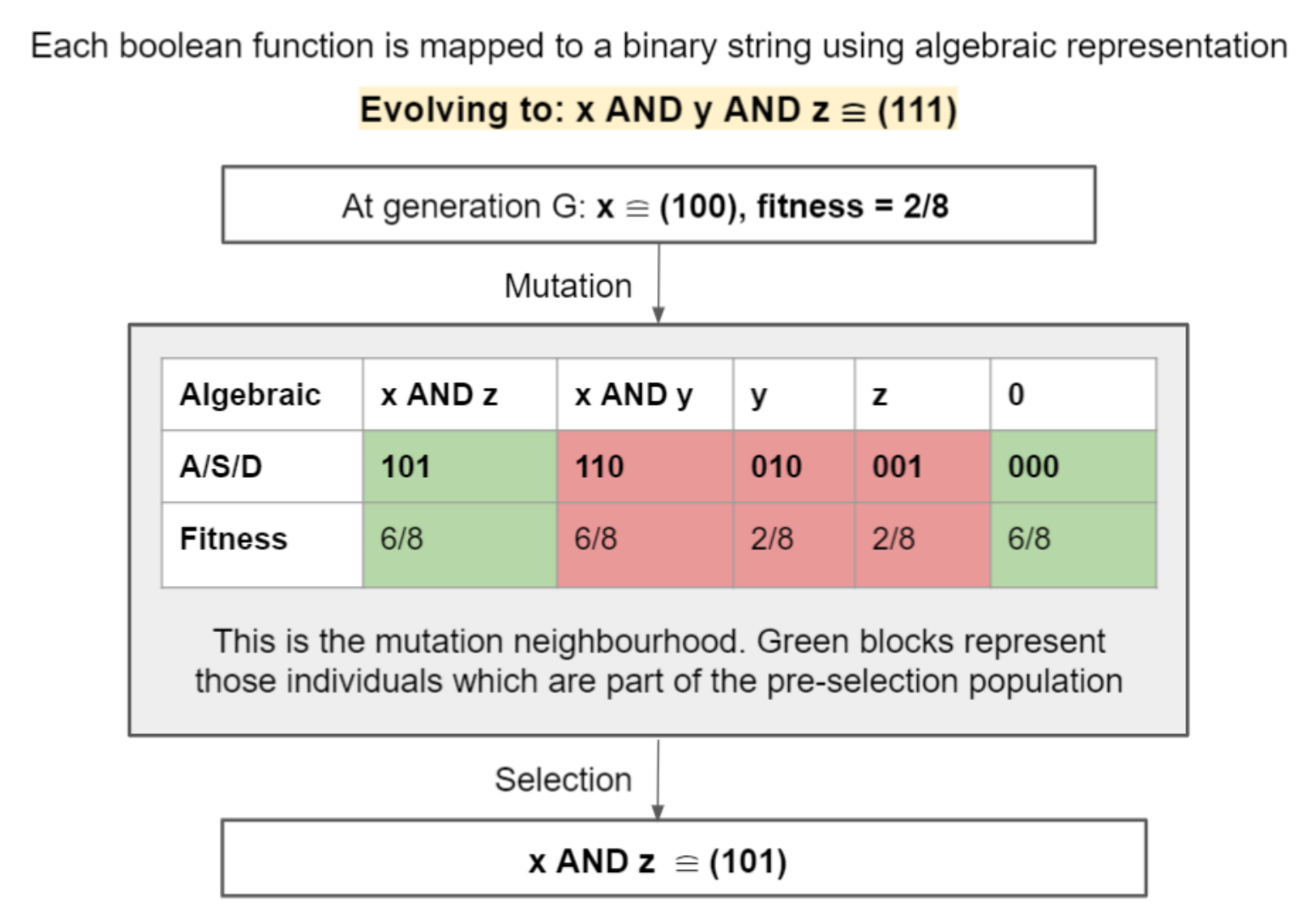
Each boolean function is mapped to a binary string using algebraic representation
Evolving to: x AND y AND z ≘ (111)
At generation G: x ≘ (100), fitness = 2/8
Mutation
Algebraic | x AND z | x AND y | y | z | 0
A/S/D | 101 | 110 | 010 | 001 | 000
Fitness | 6/8 | 6/8 | 2/8 | 2/8 | 6/8
This is the mutation neighbourhood. Green blocks represent those individuals which are part of the pre-selection population
Selection
x AND z ≘ (101)

Figure 1. Toy-example of G.A. in [3]

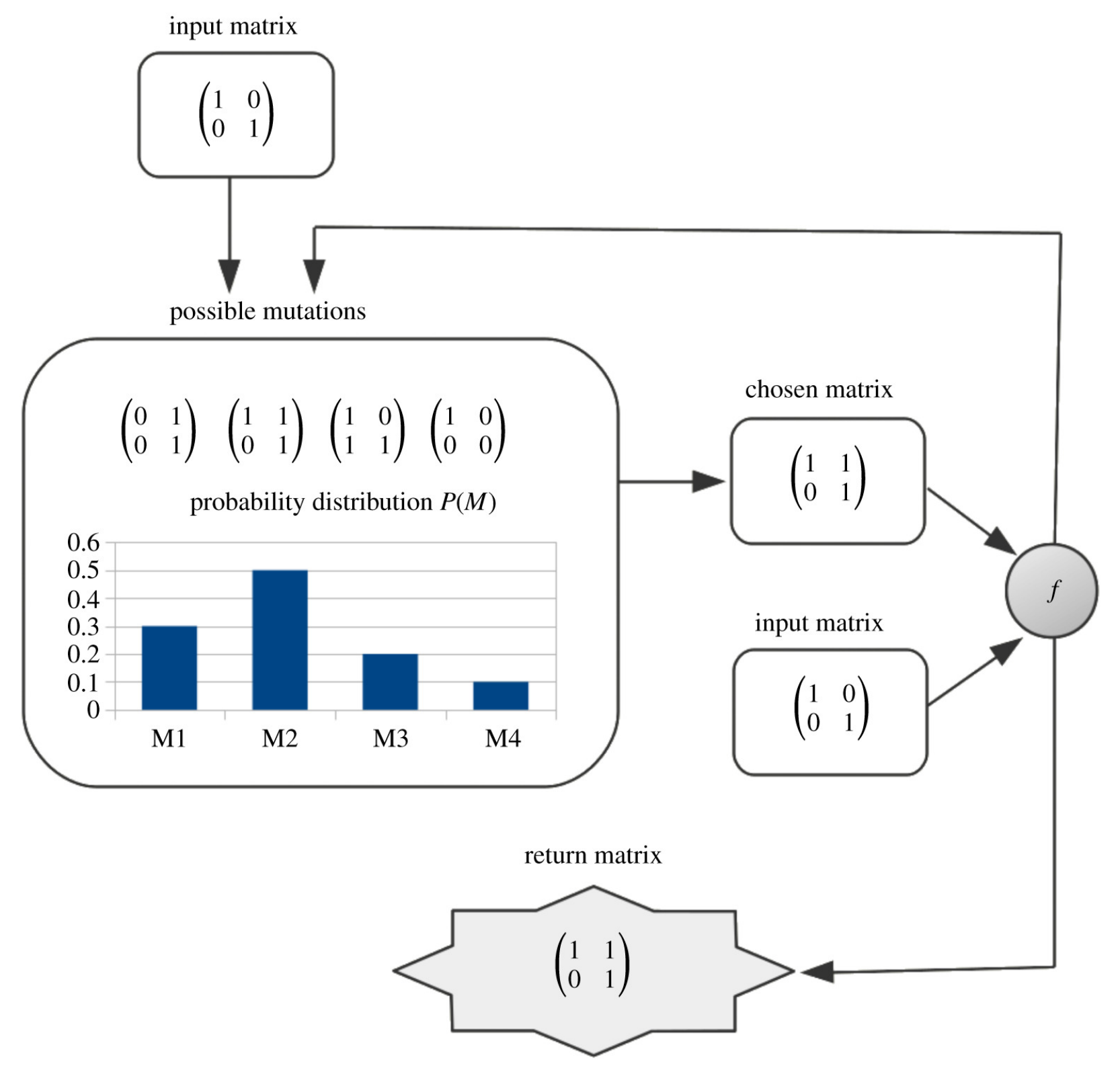

Figure 2. Toy-example of G.A. in [2], Fig. 1 from [2]

Algorithmic Information theoretic setting

The framework proposed by Orozco et al. [2] explores the setting proposed by Chaitin [1] experimentally, where they impose the **universal distribution** over the mutation neighbourhood, thus favouring mutations with low algorithmic complexity- showing that convergence to 'simpler' target matrices can be significantly faster in this case. Here, simplicity is described using BDM (Block Decomposition Method) values, which are an approximation to the Kolmogorov complexity.

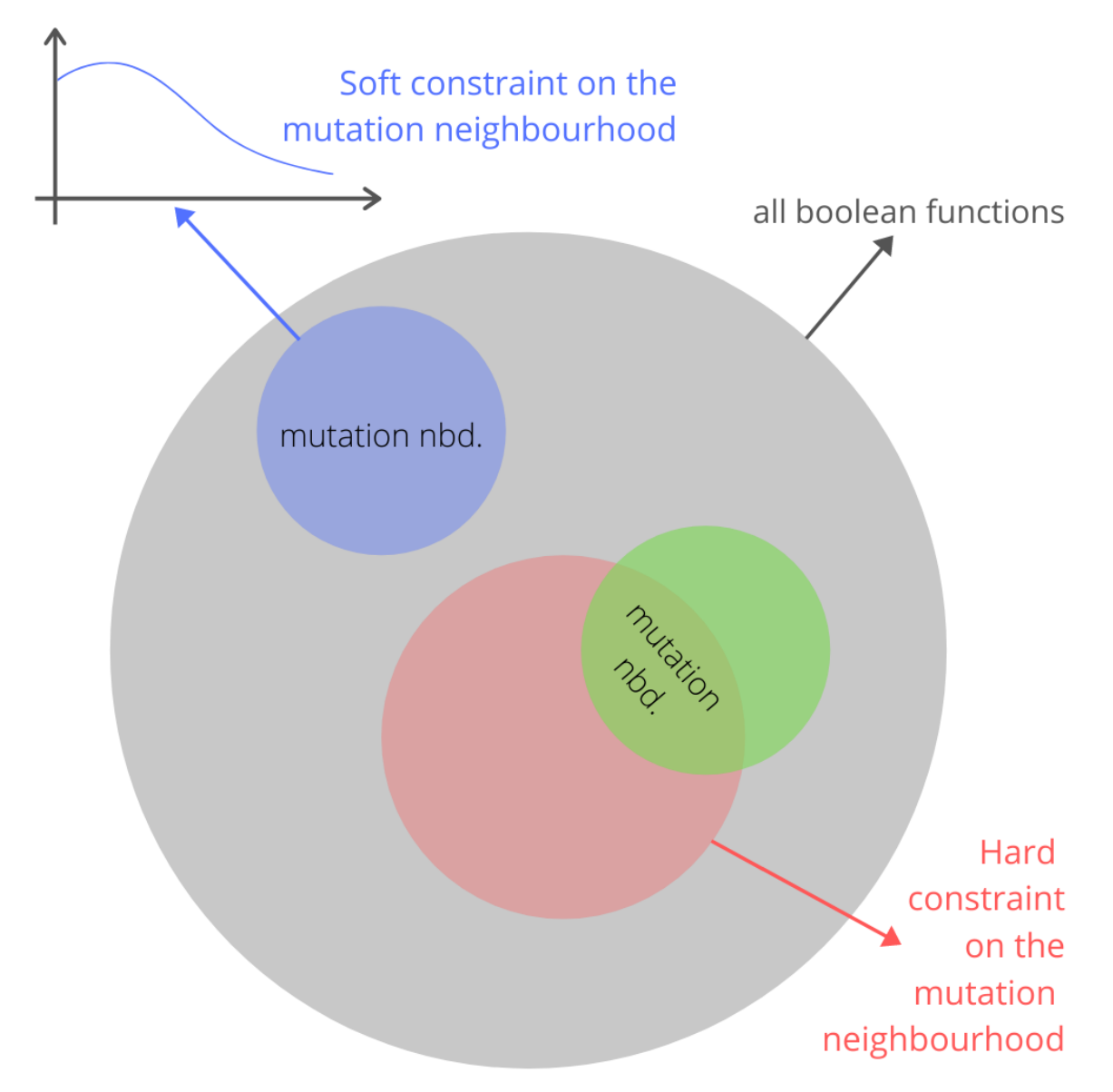
Soft constraint on the mutation neighbourhood
all boolean functions
mutation nbd.
mutation nbd.
Hard constraint on the mutation neighbourhood

Figure 3. Relating the 2 frameworks by defining a soft and hard constraint

Method

The distribution favouring low-complexity mutations in the AIT setting, called the *universal distribution*, could be interpreted as a 'soft' constraint on the mutation space, in contrast to the 'hard' constraint of explicitly restricting mutations to a chosen function class.

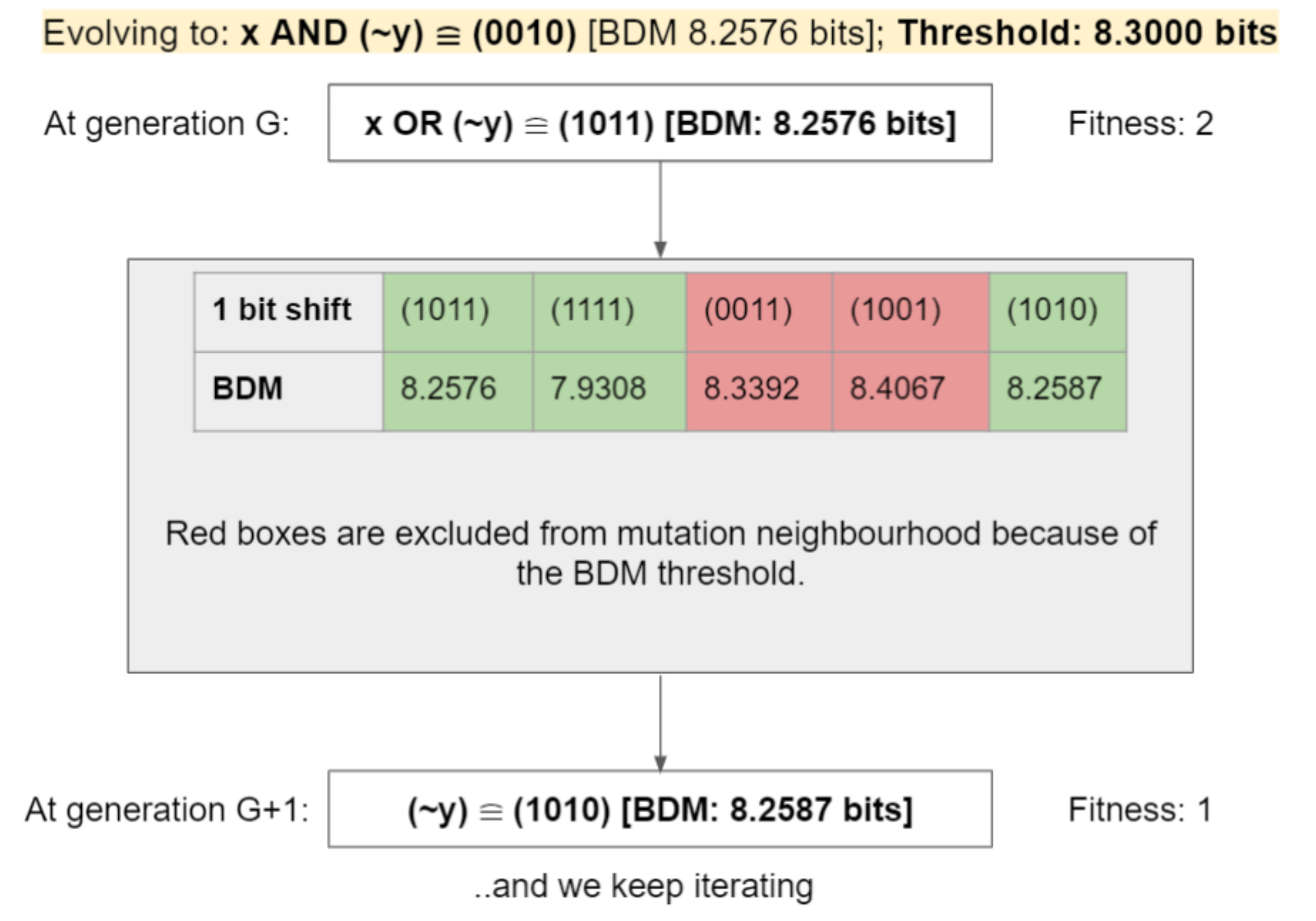
Evolving to: x AND (~y) ≘ (0010) [BDM 8.2576 bits]; Threshold: 8.3000 bits
At generation G:
x OR (~y) ≘ (1011) [BDM: 8.2576 bits]
Fitness: 2
1 bit shift | (1011) | (1111) | (0011) | (1001) | (1010)
BDM | 8.2576 | 7.9308 | 8.3392 | 8.4067 | 8.2587
Red boxes are excluded from mutation neighbourhood because of the BDM threshold.
At generation G+1:
(~y) ≘ (1010) [BDM: 8.2587 bits]
Fitness: 1
..and we keep iterating

Figure 4. Toy-example of our G.A. where hard constraint is a BDM threshold

We represent boolean functions via matrices of their corresponding minterms to study their evolution in the AIT setting. In addition to the soft constraint, we also restrict the mutation space in a hard way to a subset of all Boolean functions which are meant to be simple in terms of algorithmic structure. This is done via a BDM-value threshold where the mutation space is $M(r) = \{x : Hamming(r, x) \leq 1 \;\&\; BDM(x) \leq threshold\}$ (Figure 4).

How do BDM values relate to different function classes?

Conjunctions are spread over a much smaller range than parity functions, and both are clearly separated from randomly sampled Boolean functions- 'simpler' classes appear to have lower BDM values so the class restriction can be emulated by a BDM threshold.

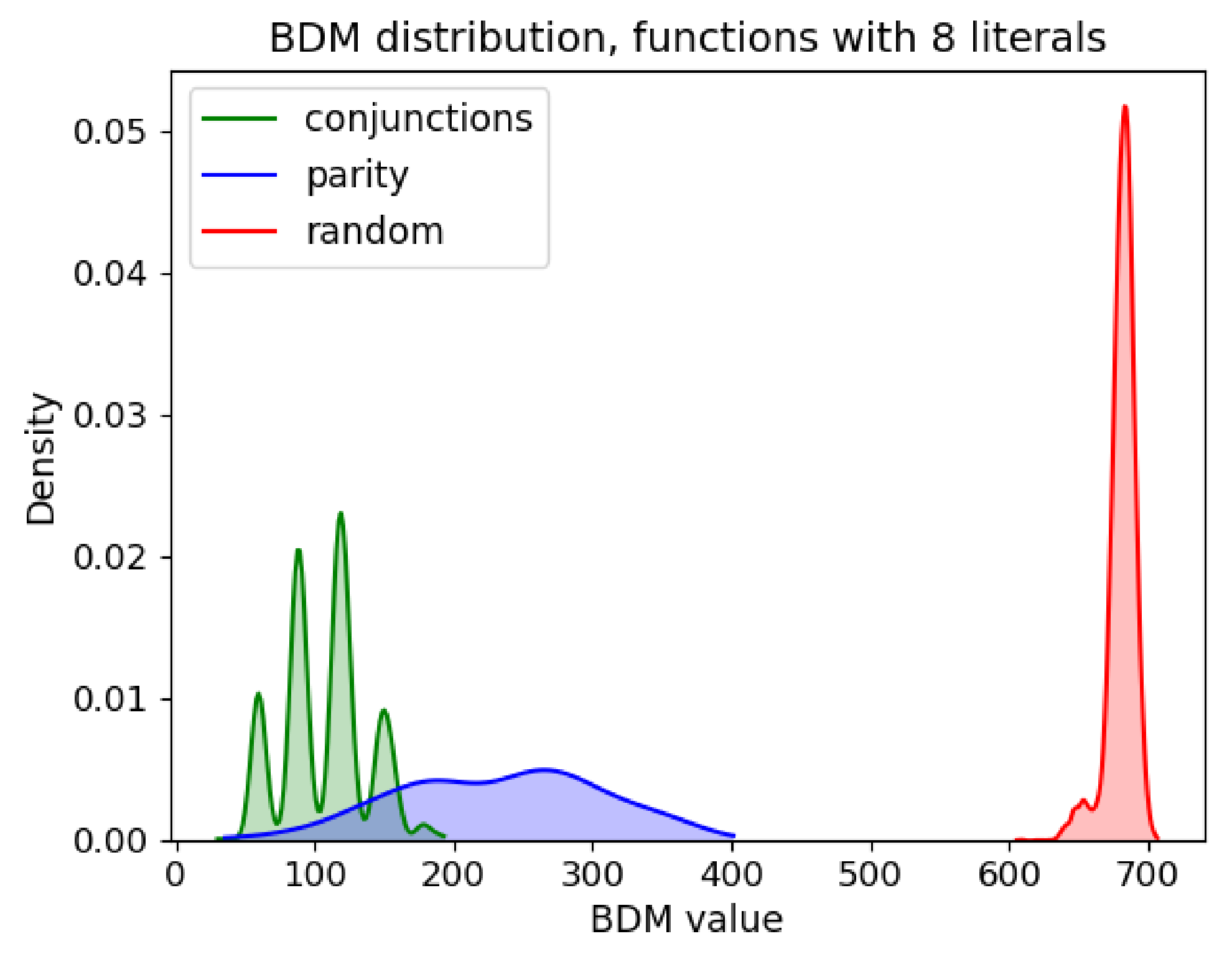

Figure 5. A hard threshold on the BDM value can approximate the selection of mutations from a chosen function class

Do thresholded settings lead to an improvement in convergence?

Mutations as per the full universal distribution are the strongest form of inductive bias, and lead to both fastest convergence and selection of mutations with the lowest BDM values (Figure 6, Table 1), while the hard threshold also sees some lowering compared to the baseline.

BDM Thr.	None	126.26 bits	174.96 bits	Full BDM distr.
Targets: 6-variable conjunctions (conjunctions)				
Gens.	9.37 ± 5.12	8.74 ± 4.82	9.01 ± 5.05	5.04 ± 3.08
Exts.	0 ± 0.0	1.84 ± 5.18	0 ± 0.0	0 ± 0.0
Targets: 6-variable odd parity functions				
Gens.	32.76 ± 7.45	29.67 ± 9.37	32.04 ± 7.11	17.15 ± 9.78
Exts.	0 ± 0.0	41.8 ± 5.89	0.02 ± 0.13	0 ± 0.0

Table 1. Generations to converge and no. of extinctions for different priors. Pink/blue: BDM threshold set at Max. BDM for conjunctions and parity fns. respectively

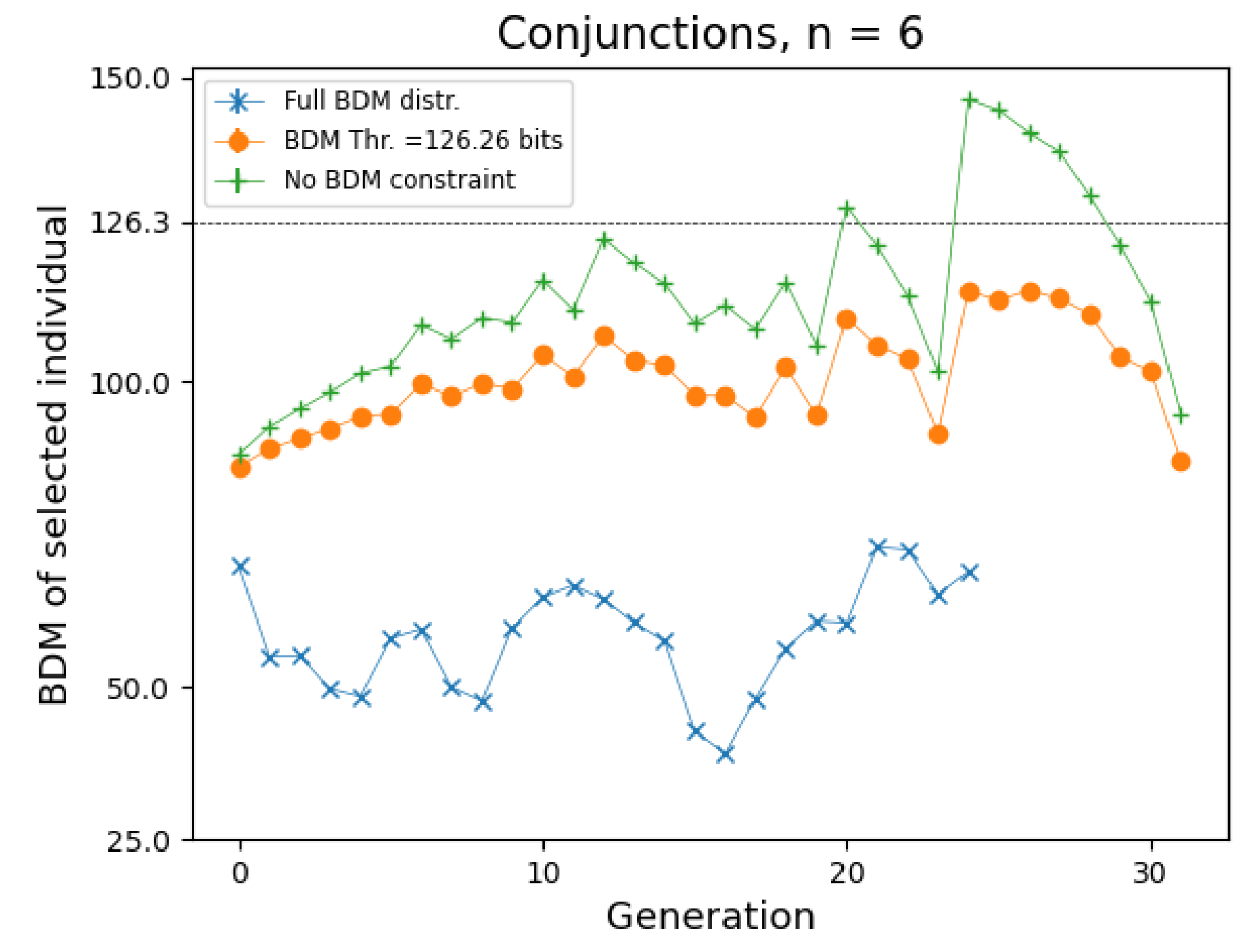

Figure 6. Generation-wise BDM values by bias setting.

Thus, bounding algorithmic complexity when evolving Boolean functions approximates the restriction of the search space to 'simple' function classes. A 'hard' constraint, the BDM threshold, is a weaker prior than the 'soft' constraint, the full universal distribution; it leads to a smaller but still notable speed-up in the evolution of functions which are simpler in the algorithmic sense.

Conclusion

We think that a restriction on realisable functional mechanisms (such as conjunctions) may be a more plausible constraint on the search space than assuming knowledge of the distribution over an exponentially large space. But the facilitation of evolution in such restricted search spaces might in turn be seen as the result of an implicit favouring of low-complexity mutations.

Acknowledgements

Grateful to the ACM-W scholarship program and Google for supporting Hetvi's attendance.

References

[1] Gregory J. Chaitin. Evolution of mutating software. *Bull. EATCS*, 97:157–164, 2009.

[2] Santiago Hernández-Orozco, Narsis A. Kiani, and Hector Zenil. Algorithmically probable mutations reproduce aspects of evolution, such as convergence rate, genetic memory and modularity. *Royal Society Open Science*, 5(8):180399, August 2018.

[3] Leslie G. Valiant. Evolvability. *Journal of the ACM*, 56(1):1–21, January 2009.