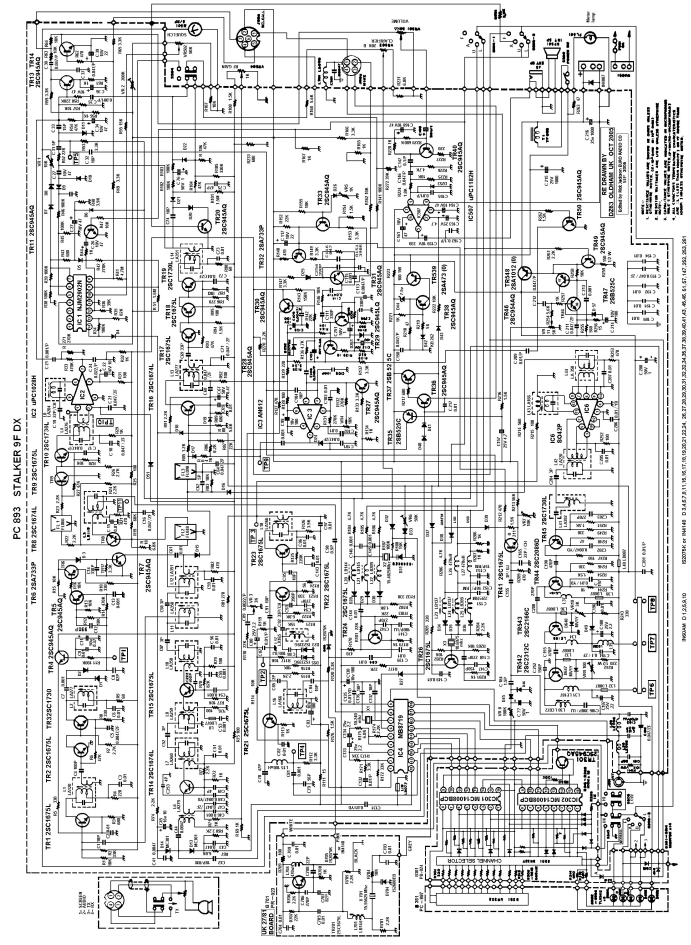
Estimation of Distribution Algorithms

Ata Kaban
School of Computer Science
The University of Birmingham

Black-box Optimisation

$$\min_{x \in \{0,1\}^n} \operatorname{temp}(x)$$

Black-box Optimisation and Learning



$$\min_{x \in \{0,1\}^n} \mathrm{temp}(x)$$

What is EDA

- EDA is a relatively new branch of evolutionary algorithms

  M Pelikan, D.E Goldberg & E Cantu-paz (**2000**): “Linkage Problem, Distribution Estimation and Bayesian Networks”. Evolutionary Computation 8(3): 311-340.

- Replaces search operators with the estimation of the distribution of selected individuals + sampling from this distribution
- The aim is to avoid the use of arbitrary operators (mutation, crossover) in favour of explicitly modelling and exploiting the distribution of promising individuals

The EDA Algorithm

Step 1: Generate an initial population P_0 of M individuals uniformly at random in the search space

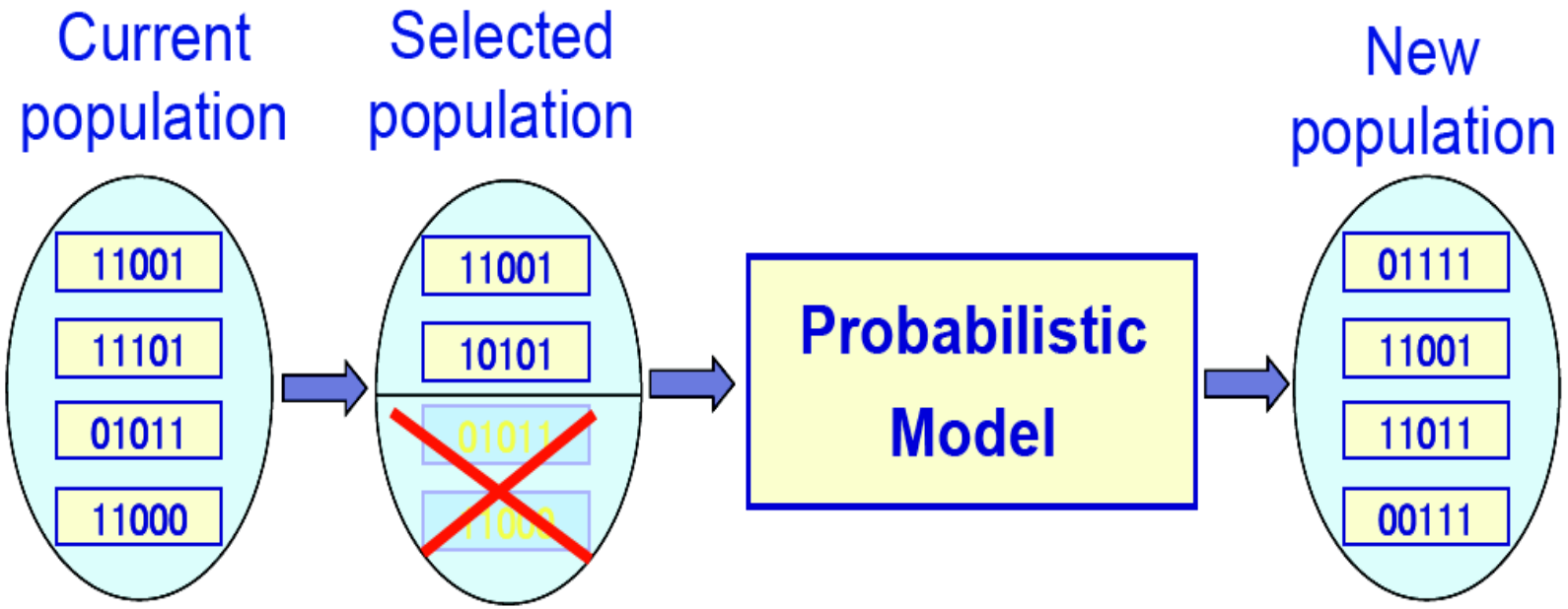
- Step 2: Repeat steps 3-5 for generations l=1, 2, … until some stopping criteria met
- Step 3: Select N<=M individuals from P_{l-1} according to a selection method
- Step 4: Estimate the probability distribution $p_l(x)$ of an individual being among the selected individuals
- Step 5: Sample M individuals (the new population) from $p_l(x)$

Typical situation

Have: Population of individuals + their fitness

#	Solution	Evaluation
1	00100	1
2	11011	4
3	01101	0
4	10111	3

Want: What solution to generate next?



...replace crossover+mutation with learning and sampling probabilistic model

Probabilistic Model

Different EDA approaches according to different ways to construct the probabilistic model.

A very simple idea

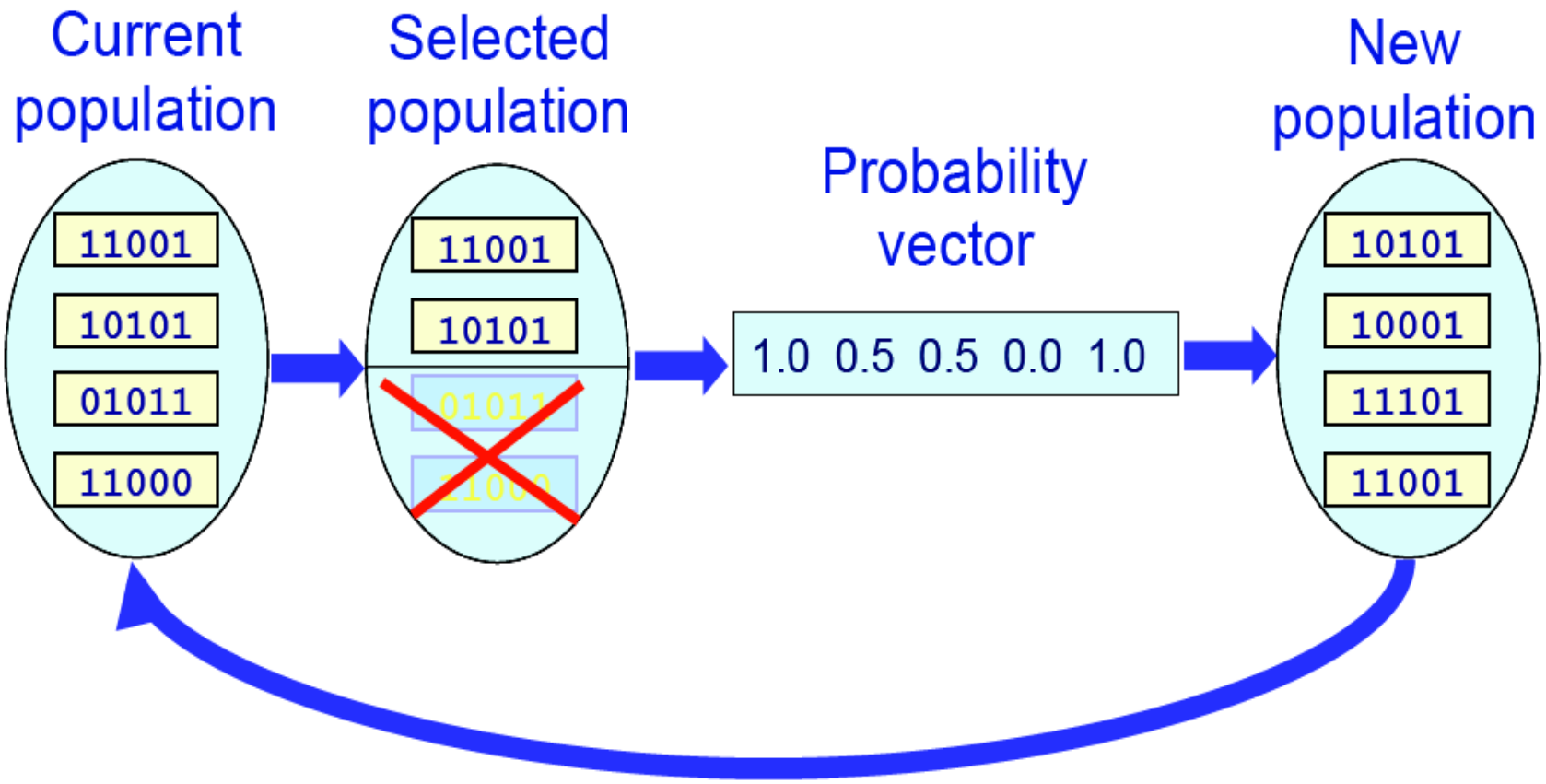
- for n-bit binary strings

- model: a probability vector $p=(p_1,...,p_n)$

  p_i = probability of 1 in position i

  Learn p: compute the proportion of 1 in each position

  Sample p: Sample 1 in position i with probability p_i.



Note: The variables (bits, genes) are treated independently here.
'Univariate Marginal Distribution Algorithm' (UMDA)

How does it work?

Bits that perform better get more copies

And get combined in new ways

But context of each bit is ignored.

Example problem 1: Onemax

$$f(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i$$

Univariate Marginal Distribution Algorithm

1: Initialise the vector $p_0 := (1/2, \ldots, 1/2)$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Sample λ bitstrings $y^1, \ldots, y^\lambda$ according to the distribution

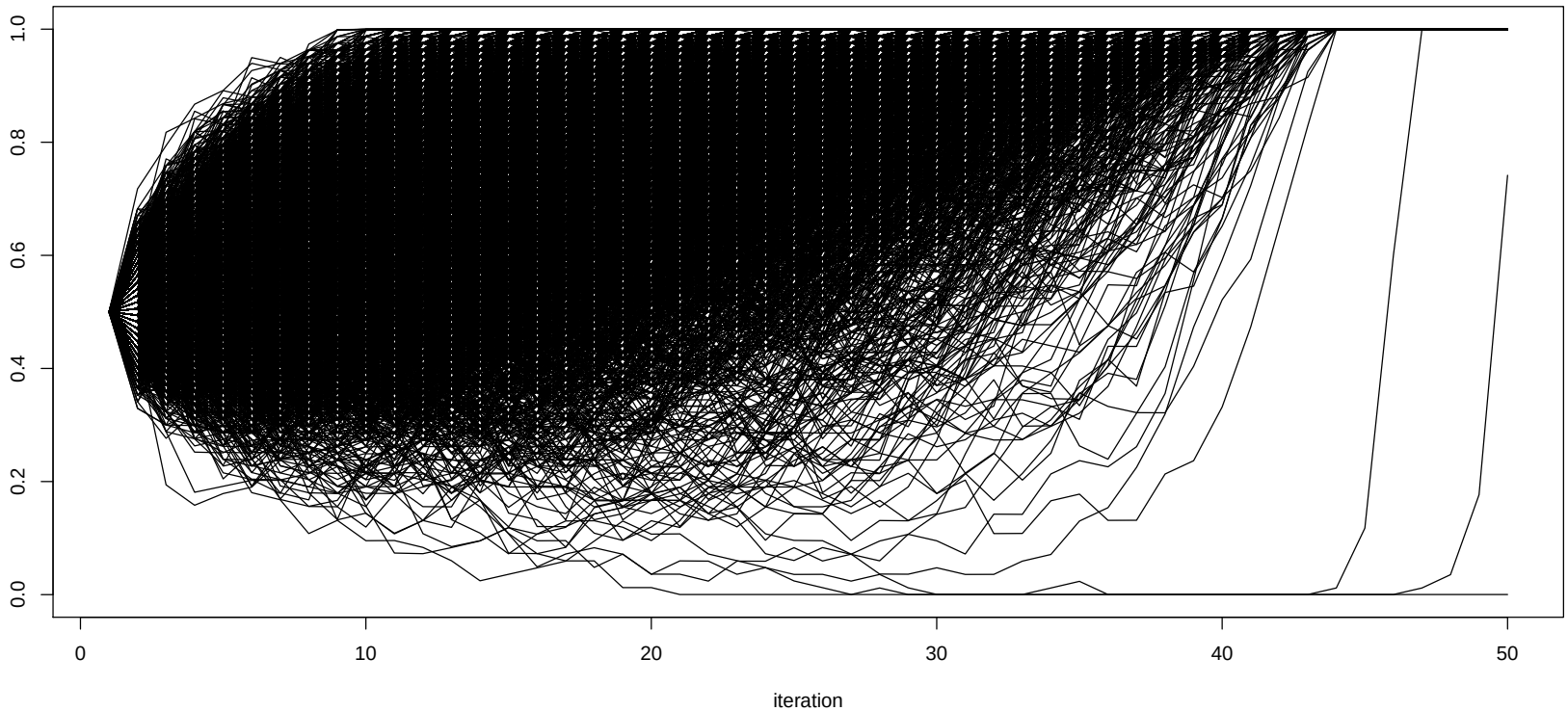
$$p_t(x) = \prod_{i=1}^{n} p_t(i)^{x_i} (1 - p_t(i))^{1-x_i}$$

4: Let $y^{(1)}, \ldots, y^{(\lambda)}$ be the bitstrings sorted by fitness func. f
5: Compute the next vector p_{t+1} according to

$$p_{t+1}(i) := \begin{cases} \frac{1}{n} & \text{if } X_i = 0 \\ \frac{X_i}{\mu} & \text{if } 1 \leq X_i \leq \mu - 1 \\ 1 - \frac{1}{n} & \text{if } X_i = \mu, \end{cases}$$

6: where $X_i := \sum_{j=1}^{\mu} y_i^{(j)}$.

UMDA on ONEMAX ($n = 5000$)

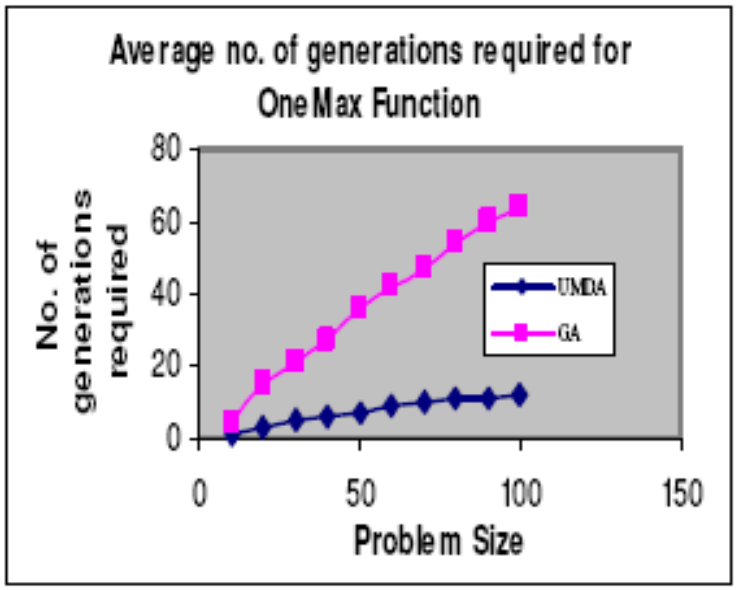


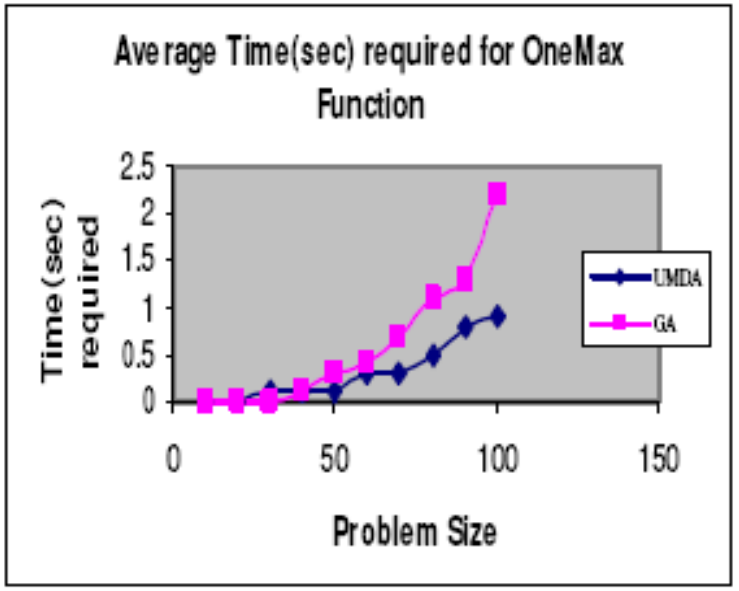
Different EDA approaches

- Independent variables (compute only a probability vector)
  - ‘Univariate Marginal Distribution Algorithm (UMDA)’
  - ‘Population Based Incremental Learning (PBIL)’
  - ‘Compact Genetic Algorithm (CGA)’
- Bivariate Dependencies
  - …
- Multiple Dependencies
  - …

+ Discrete vs Continuous versions

Experimental Results (OneMax Function)





Compared with GA with one point crossover & mutation

Example problem 2: Subset Sum

Given a set of integers and a weight, find a subset of the set so that the sum of its elements equals the weight.

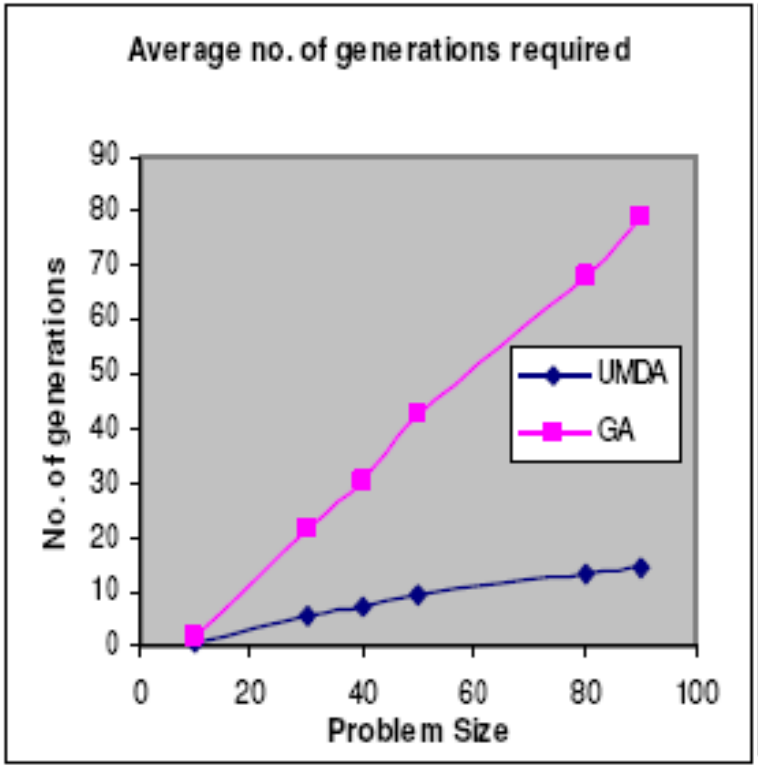
E.g. given {1,3,5,6,8,10}, W=14

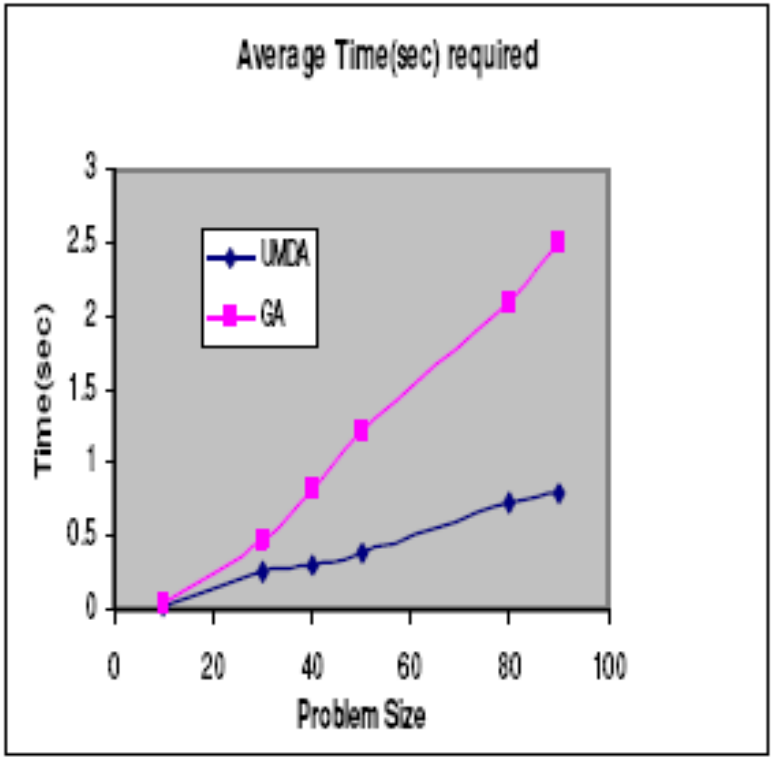
solutions: {1,3,10},{3,5,6},{6,8},{1,5,8}

UMDA and GA for Subset Sum Problem

- Fitness: Absolute difference between sum of variables in an individual and expected weight
- Termination: Sum of variables in an individual equal to expected weight
- GA: one point crossover and mutation

Experimental Results (Subset Sum Problem)-I





Some theoretical results

Theorem 7 *The UMDA (with margins) with parent population size* $\mu \geq c \log n$ *for a sufficiently large constant* $c > 0$*, and offspring population size* $\lambda \geq (1+\delta)e\mu$ *for any constant* $\delta > 0$*, has expected optimisation time* $\mathcal{O}(n\lambda \log \lambda + n^2)$ *on* LEADINGONES *and* BINVAL.

Theorem 8 *For some constant* $a > 0$ *and any constant* $c \in (0, 1)$*, the UMDA (with margins) with parent population size* $a \ln(n) \leq \mu \leq \sqrt{n(1-c)}$*, and offspring population size* $\lambda \geq (13e/(1-c))\mu$*, has expected optimisation time* $\mathcal{O}(n\lambda)$ *on* ONEMAX.

Theorem 9 *For sufficiently large constants* $a > 1$ *and* $c > 0$*, the UMDA (with margins) with offspring population size* $\lambda \geq a\mu$*, and parent population size* $\mu \geq c\sqrt{n} \log n$*, has expected optimisation time* $\mathcal{O}\left(\lambda\sqrt{n}\right)$ *on* ONEMAX.

Algorithmica (2019) 81:668–702
https://doi.org/10.1007/s00453-018-0507-5

Level-Based Analysis of the Univariate Marginal Distribution Algorithm

Duc-Cuong Dang[1] · Per Kristian Lehre[2] · Phan Trung Hai Nguyen[2]

Does the simple probability vector idea always work?

When does it fail?

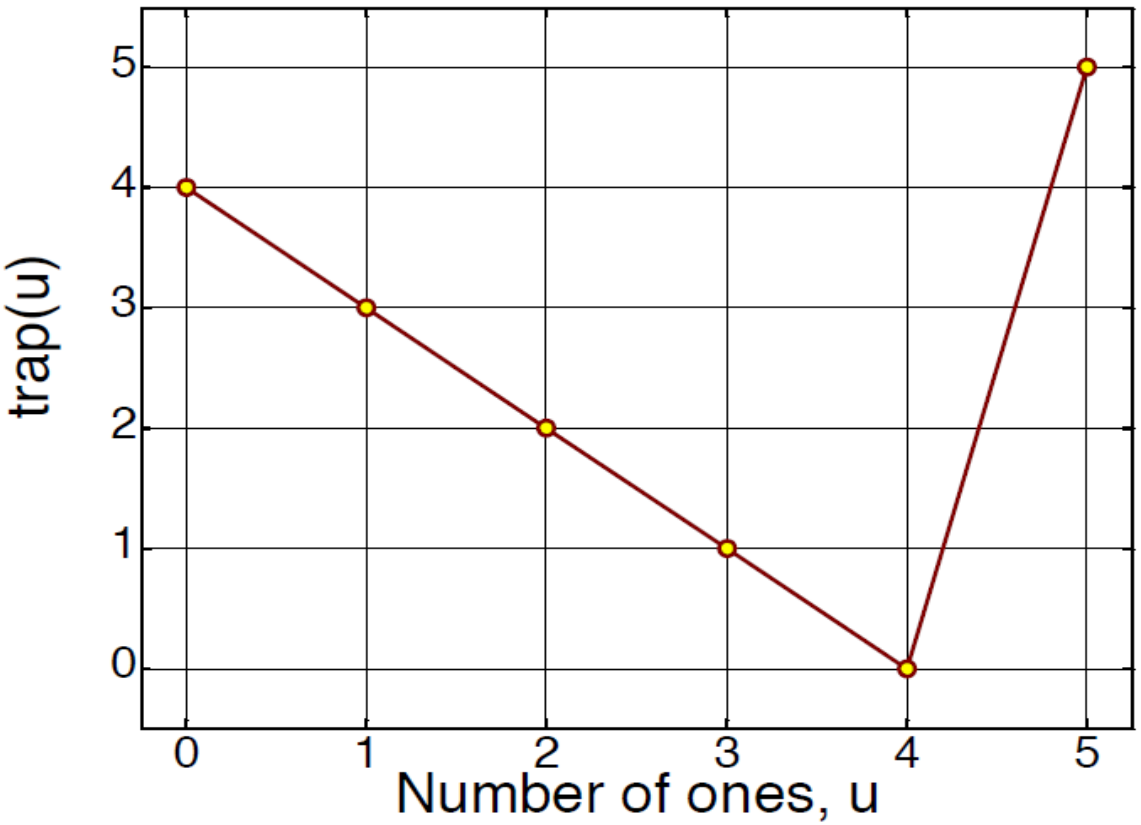
Example problem 3: Concatenated traps

- string consists of groups of 5 bits
- each group contributes to the fitness via trap(ones=nr of ones):

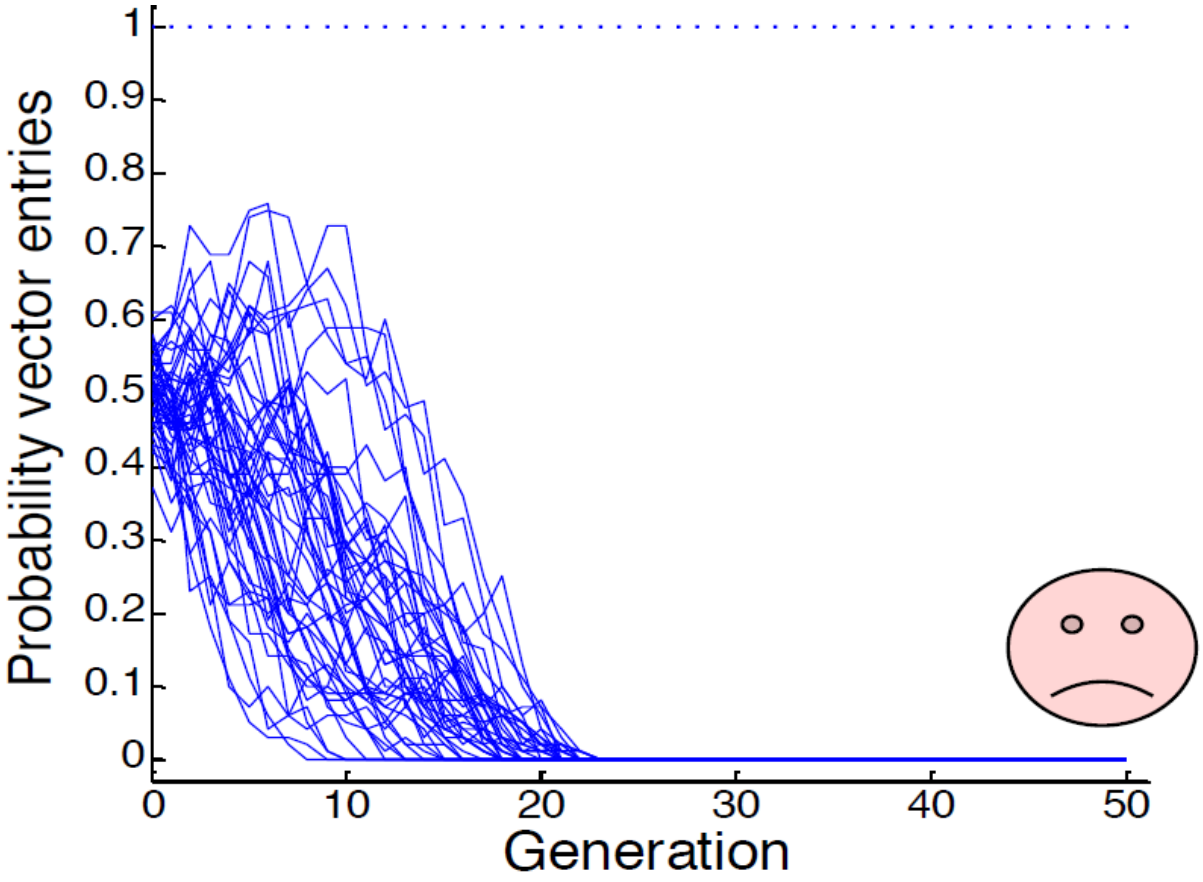
$$trap(ones) = \begin{cases} 5 & \text{if } ones = 5 \\ 4 - ones & \text{otherwise} \end{cases}$$

-fitness = sum of single trap functions

Global optimum is still 111...1



Will the simple idea of a probability vector still work on this?


Probability vector entries
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
Generation

Why did it fail?

It failed because probabilities single bits are misleading in the Traps problem.

Will it always fail?

Yes.

Take the case with a single group. The global optimum is at 11111. But the average fitness of “0****” is 2, whereas the average fitness of “1****” is 1.375 (homework to verify!)

Single bits are misleading.

How to fix it?

If we would consider 5-bit statistics instead if 1-bit ones...

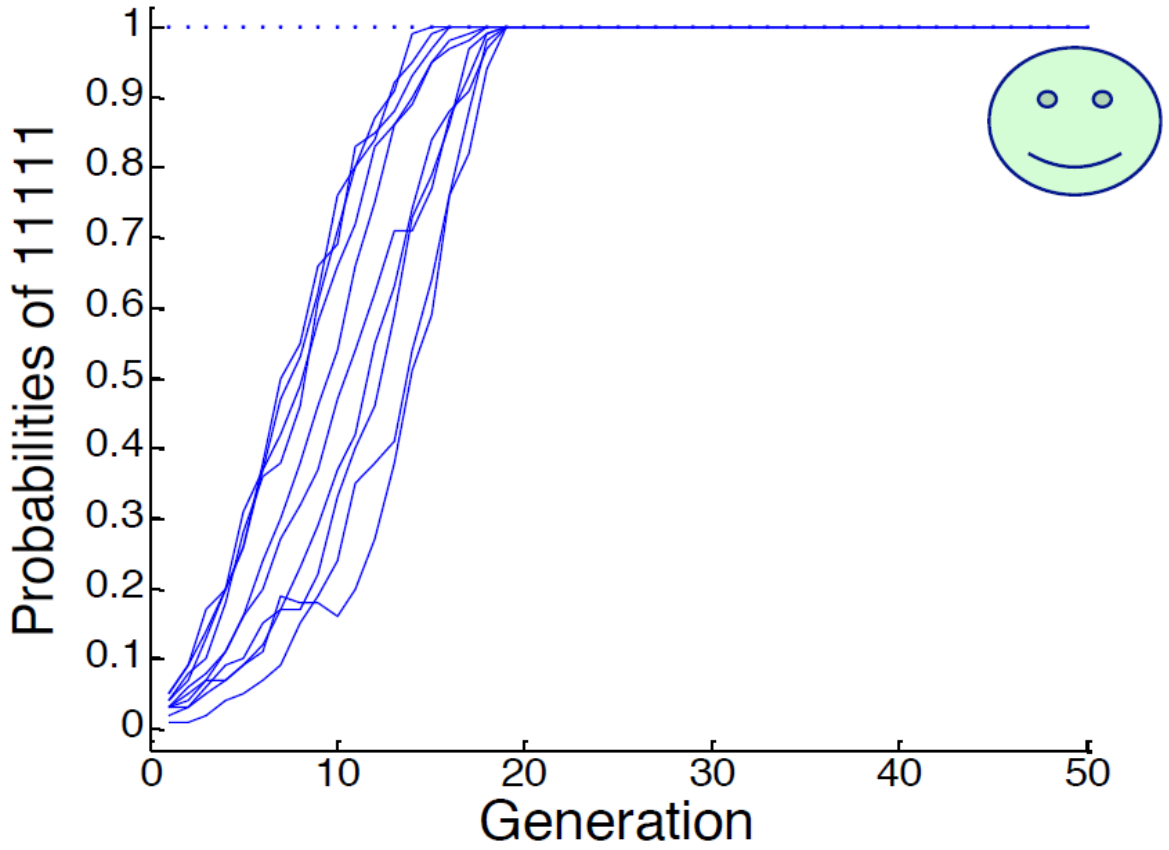
...then 11111 would outperform 00000.

Learn model:

Compute p(00000), p(00001), …, p(11111)

Sample model:

Generate 00000 with p(00000), etc.

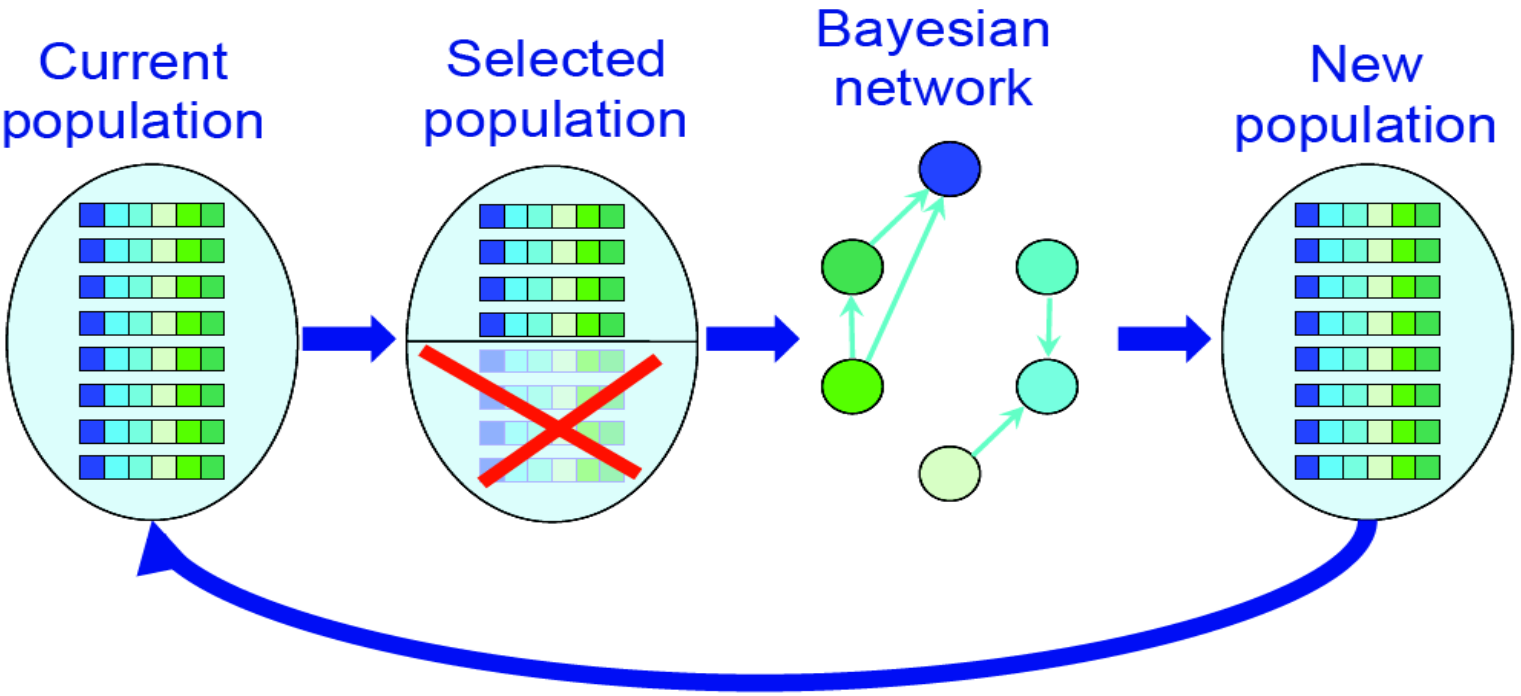


The good news: The correct model works!

But we used knowledge of the problem structure.

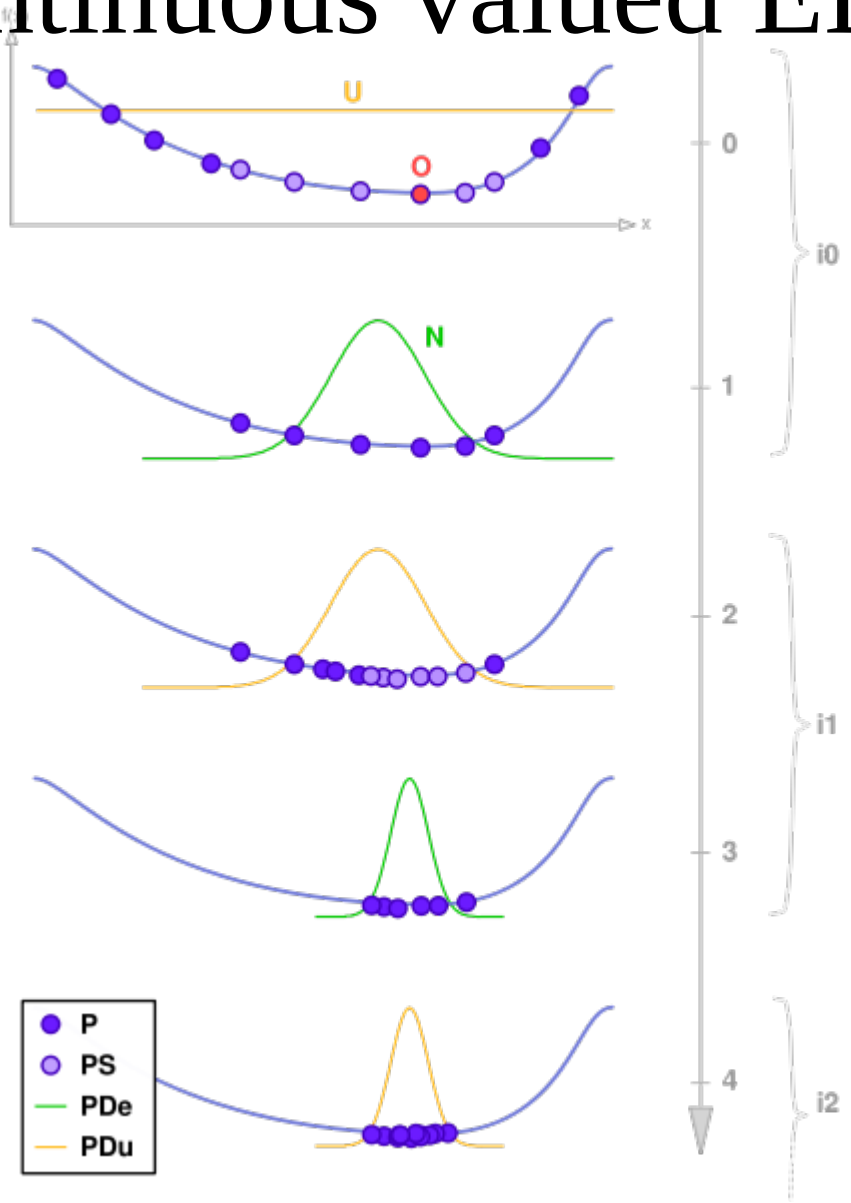
In practice the challenge is to estimate the correct model when the problem structure is unknown.

Bayesian optimisation algorithm (BOA)

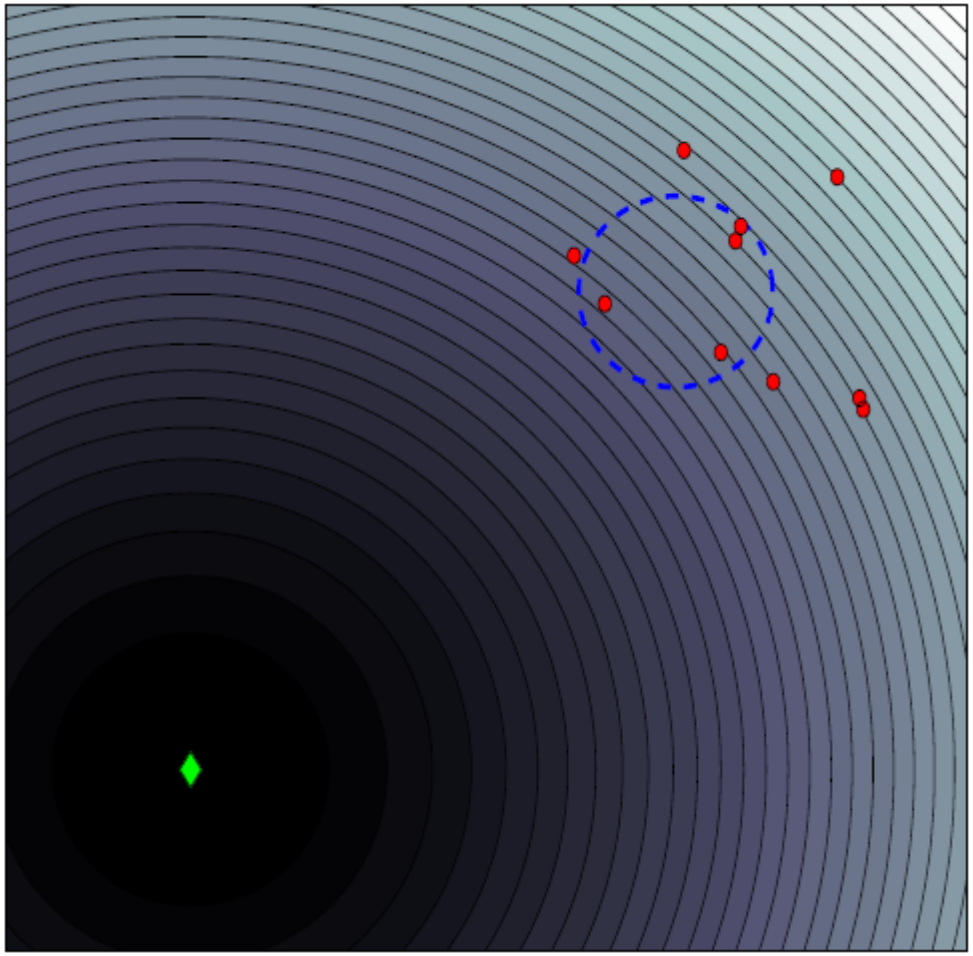


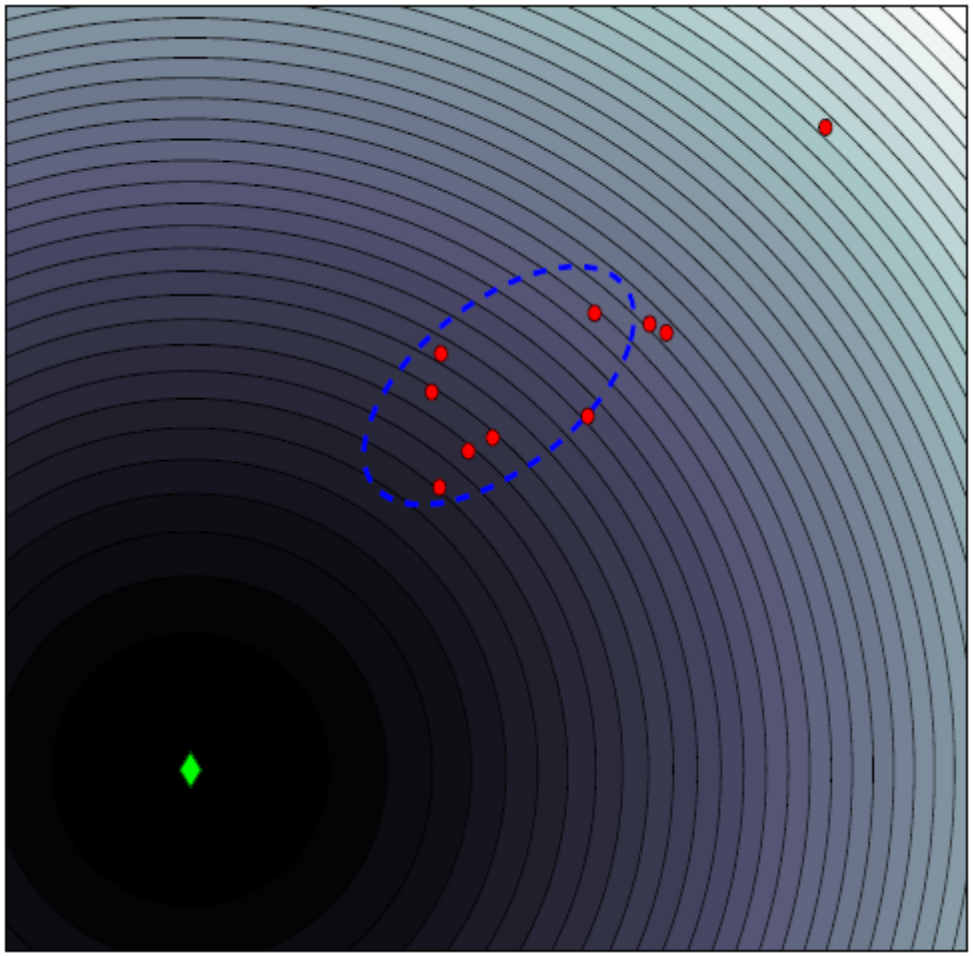
Many researchers spend their lives working out good ways to learn a dependency model, e.g. a Bayesian network.

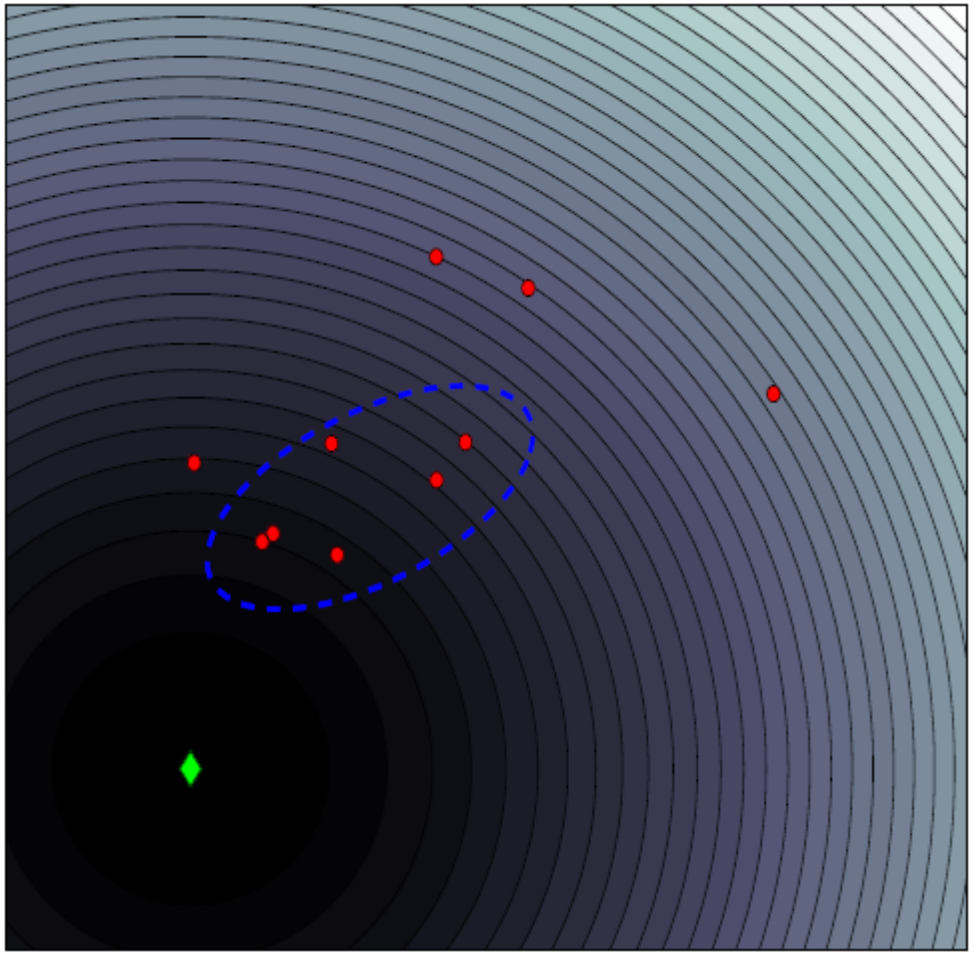
Continuous valued EDA

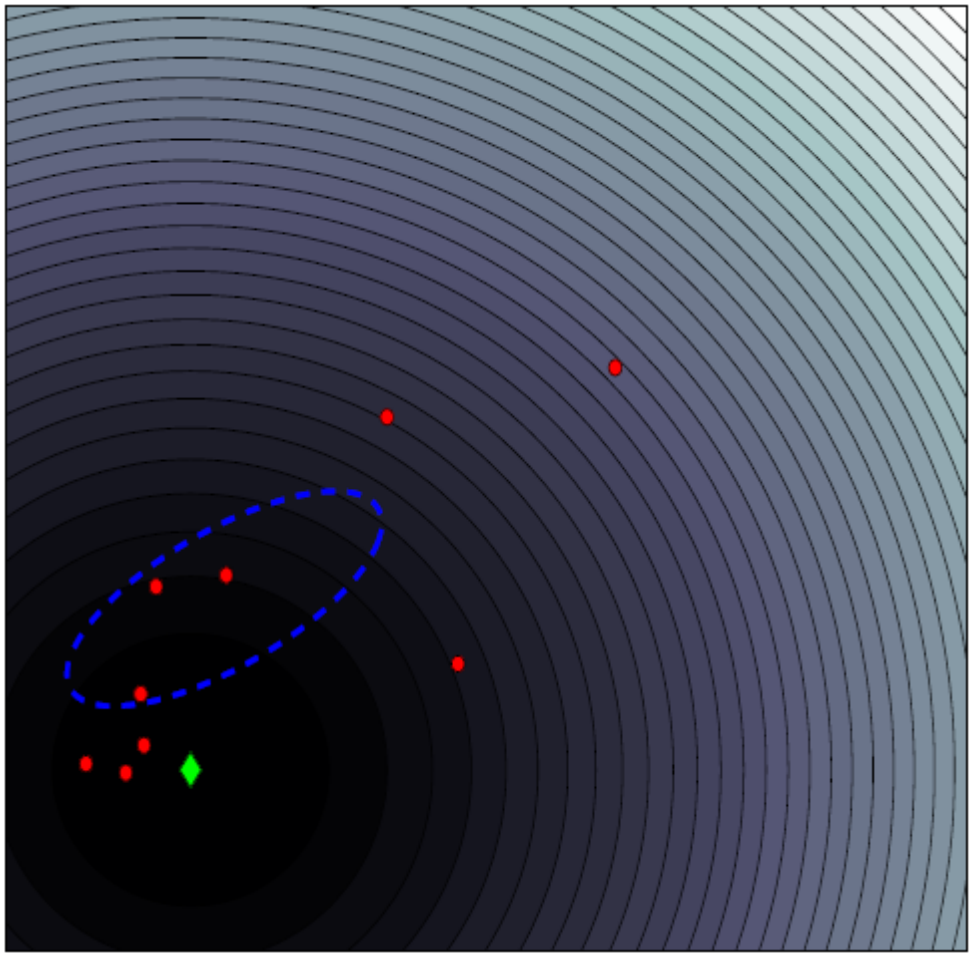


2D problem

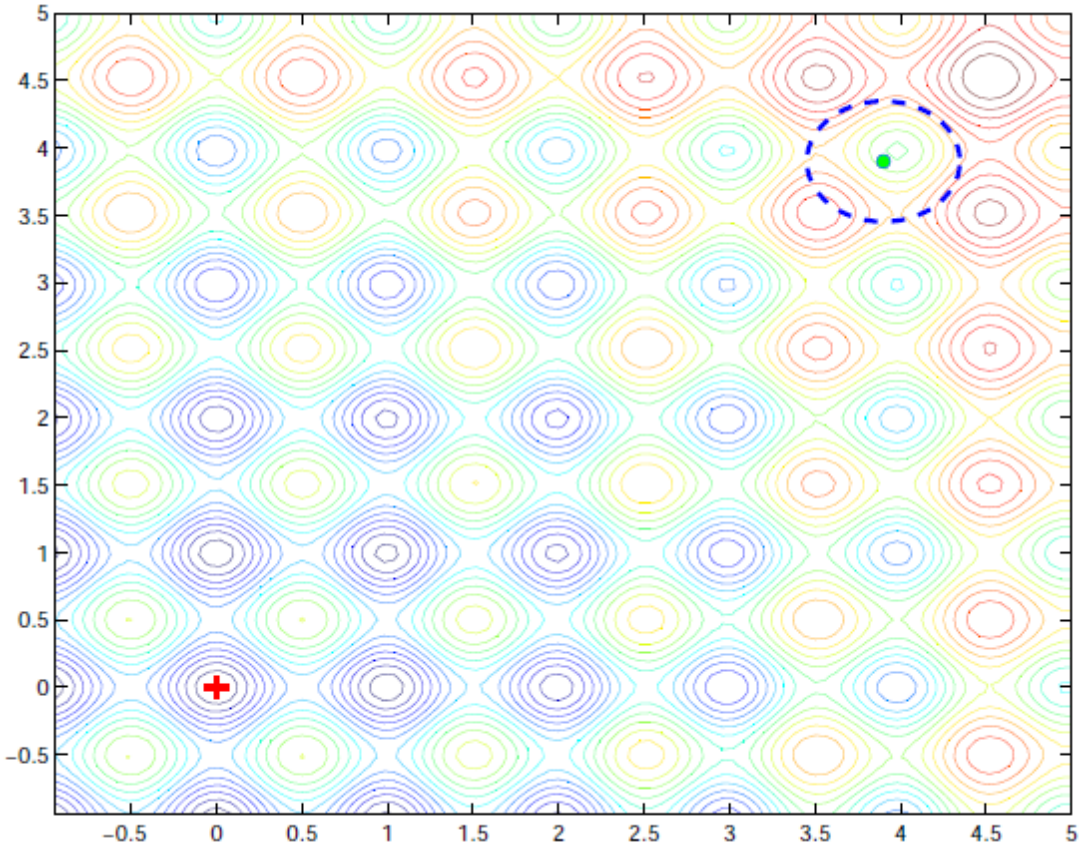


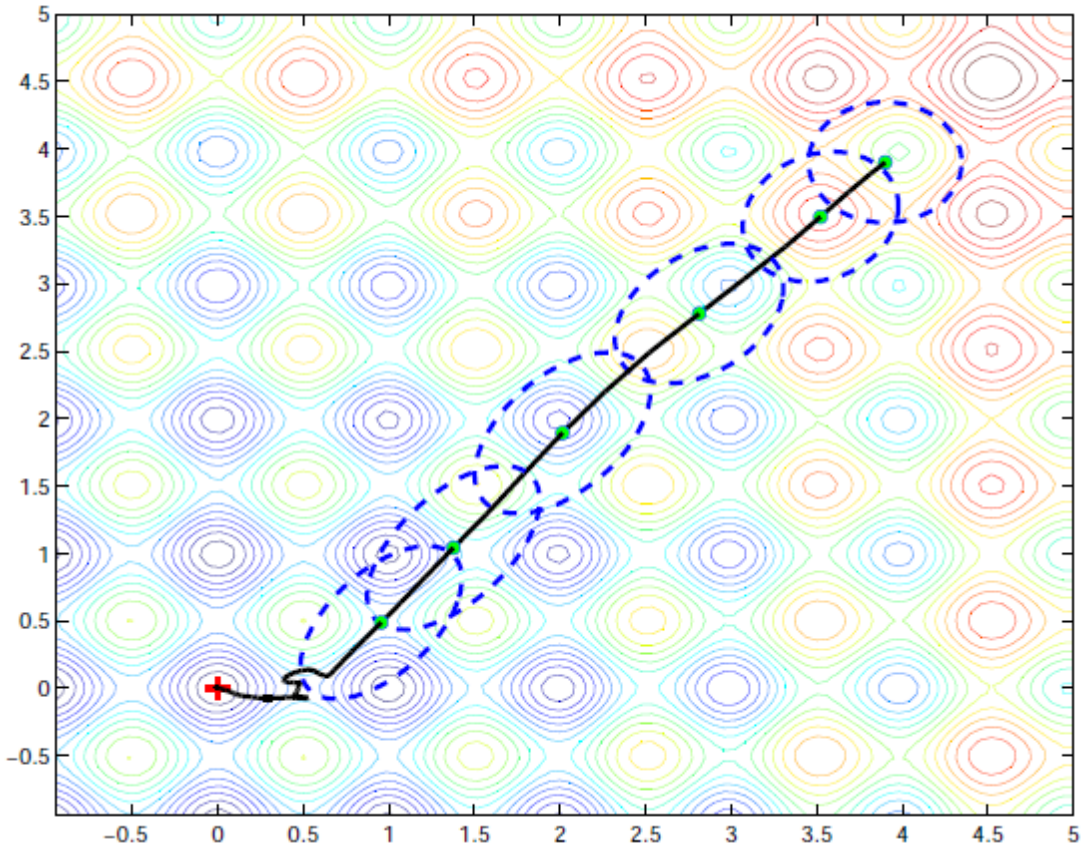






Multi-modal problem




5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
−0.5
−0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Conclusions

- EDA: a new class of EA that does not use conventional crossover and mutation operators; instead it estimates the distribution of the selected ‘parent population’ and uses a sampling step for offspring generation.
- With a good probabilistic model it can improve over conventional EA's.
- Additional advantage is the provision of a series of probabilistic models that reveal a lot of information about the problem

Resources

- EDA page complete with tutorial, sw & demo, by Topon Kumar Paul and Hitoshi Iba:

http://www.iba.t.u-tokyo.ac.jp/english/EDA.htm

- Martin Pelikan videolecture of Tutorial from GECCO'08

http://martinpelikan.net/presentations.html

(+includes references to key papers & software)

- MatLab Toolbox for many versions of EDA:

http://www.sc.ehu.es/ccwbayes/members/rsantana/software/matlab/MATEDA.html