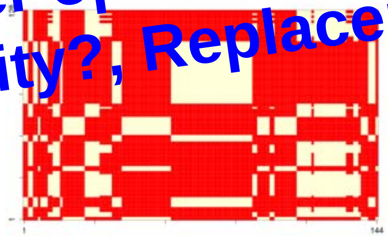
Submitted Soft Computing

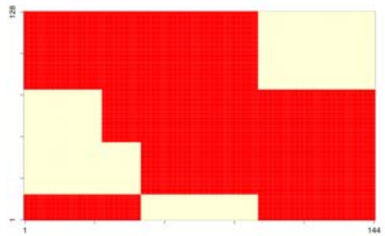
Parameter Control of Genetic Algorithms by Learning and Simulation of Bayesian Networks

C. Bielza,*, J.A. Fernández del Pozo, P. Larrañaga

Universidad Politécnica de Madrid, Departamento de Inteligencia Artificial, 28660 Boadilla del Monte, Madrid, Spain

Initial Table





Initial population?, Crossover operator?/probability?, Mutation operator?/probability?, Replacement policy?, Stopping criterion?

Genetic Algorithm

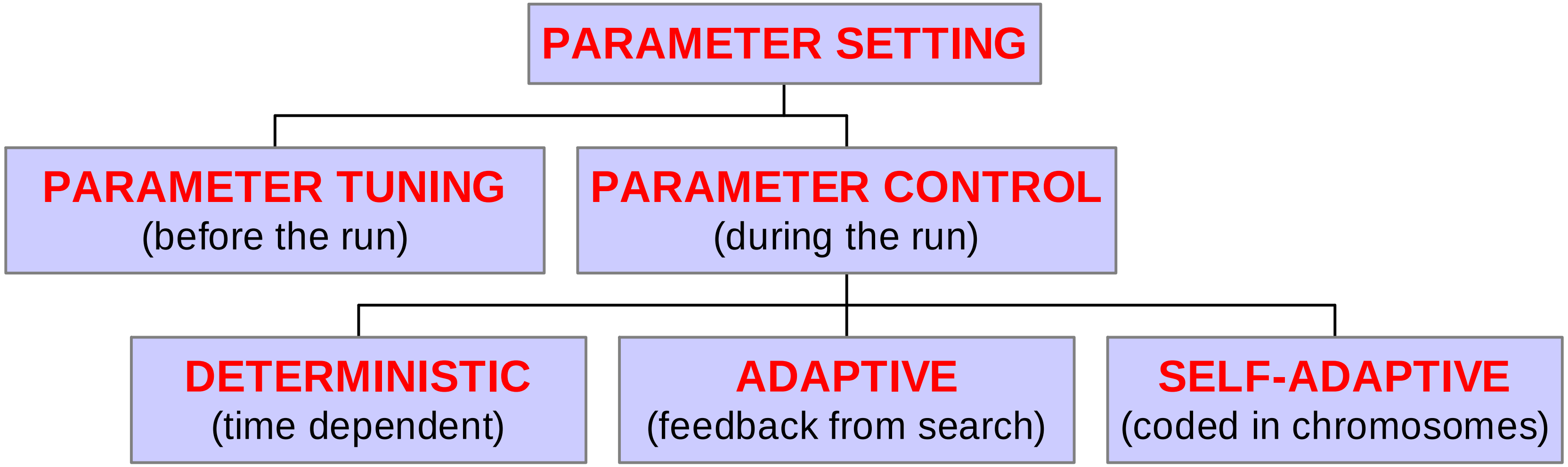
Optimal Table

Outline

Parameter setting for evolutionary algorithms is still an important issue in evolutionary computation.

There are two main approaches to parameter setting:

- Parameter tuning and
- Parameter control.



A.E. Eiben and J.E Smith

In this paper, we introduce self-adaptive parameter control of a genetic algorithm based on Bayesian network learning and simulation

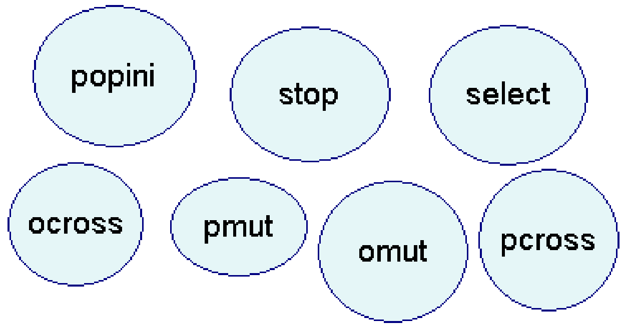
~ meta-GA (Grefenstette, 1986; Friesleben and Hartfelder, 1993; Bäck, 1996) and the nested evolution strategy (Rechenberg, 1994)

Outline

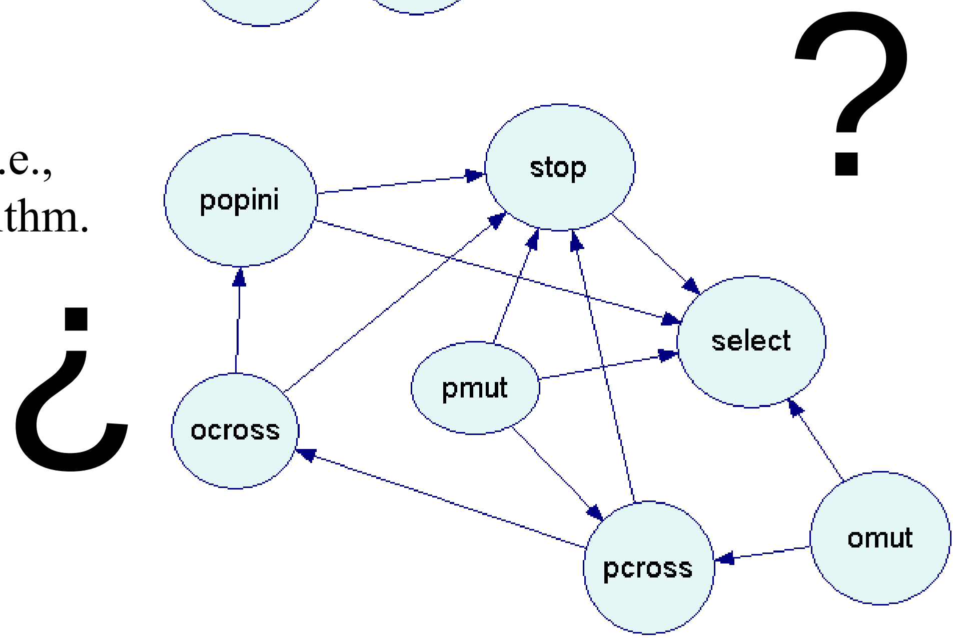
•The nodes of this Bayesian network are
the parameters of the genetic algorithm to be controlled.

SOLUTIONS <- GA(PROBLEM,popini,stop,select,
ocross, omut, pcross,pmut,…..)

•Its structure captures probabilistic conditional (in)dependence relationships between the parameters.



•This is learned from the best individuals, i.e.,
the best configurations of the genetic algorithm.



•To evaluate individuals,
the genetic algorithm has to be run with
the corresponding parameter configuration.

Outline

•Since all these runs are time-consuming tasks, each genetic algorithm is run with small population sizes and is stopped before convergence.

•This way there is a guarantee of not losing the promising individuals.

•Experiments with an optimal search problem for simultaneous row and column orderings yield the same optima as state-of-the-art methods but with a sharp computational time reduction.

•Moreover, our approach can cope with as yet unsolved high-dimensional problems.

Introduction

An EA has many strategy parameters, e.g.
- mutation operator and mutation rate
- crossover operator and crossover rate
- selection mechanism and selective pressure (e.g. tournament size)
- population size

Good parameter values facilitate good performance

Q1 How to find good parameter values (for one/many fitness landscape**)?**

A general-purpose universal optimization strategy is theoretically impossible,
Wolpert, D., Macready, W. (1997).
No free lunch theorems for optimization.
IEEE Transactions on Evolutionary Computation, 1, 67-82

- EA parameters are rigid (constant during a run)

BUT an EA is a dynamic, adaptive process
THUS optimal parameter values may vary during a run

Q2: How to vary parameter values?

Introduction

Parameter tuning: the traditional way of testing and comparing different values before the "real" run

Problems:

- users mistakes in settings can be sources of errors or sub-optimal performance
- costs much time
- parameters interact: exhaustive search is not practicable
- good values may become bad during the run

Parameter control: setting values on-line, during the actual run, e.g.

- predetermined time-varying schedule p = p(t)
- using feedback from the search process
- encoding parameters in chromosomes and rely on natural selection

Problems:

- finding optimal p is hard, finding optimal p(t) is harder
- still user-defined feedback mechanism, how to ``optimize''?
- when would natural selection work for strategy parameters?

Introduction

Three major types of parameter control:

deterministic:
some rule modifies strategy parameter without feedback from the search (based on some counter)

adaptive:
feedback rule based on some measure monitoring search progress

self-adaptative:
parameter values evolve along with solutions; encoded onto chromosomes they undergo variation and selection

Tarjet
Level
Trigger

Practically any EA component can be parameterized and thus controlled on-the-fly:

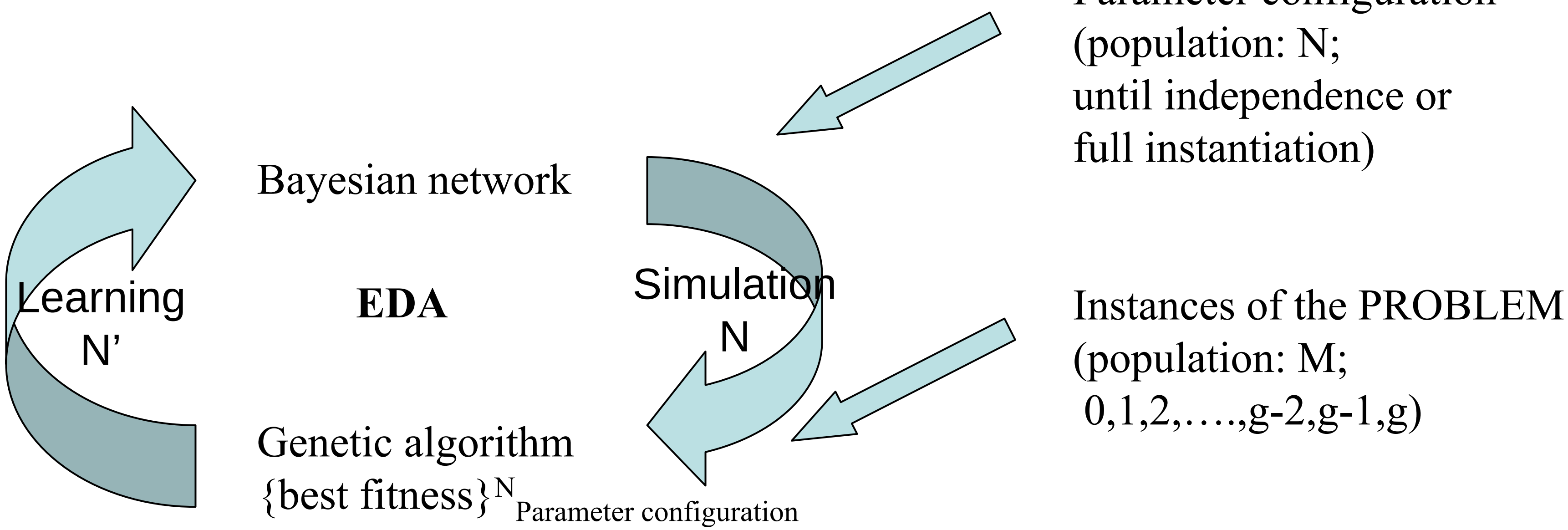
- representation
- evaluation function
- variation operators
- selection operator (parent or mating selection)
- replacement operator (survival or environmental selection)
- population (size, topology)

Self-adaptive genetic algorithm parameter control guided by an estimation of distribution algorithm

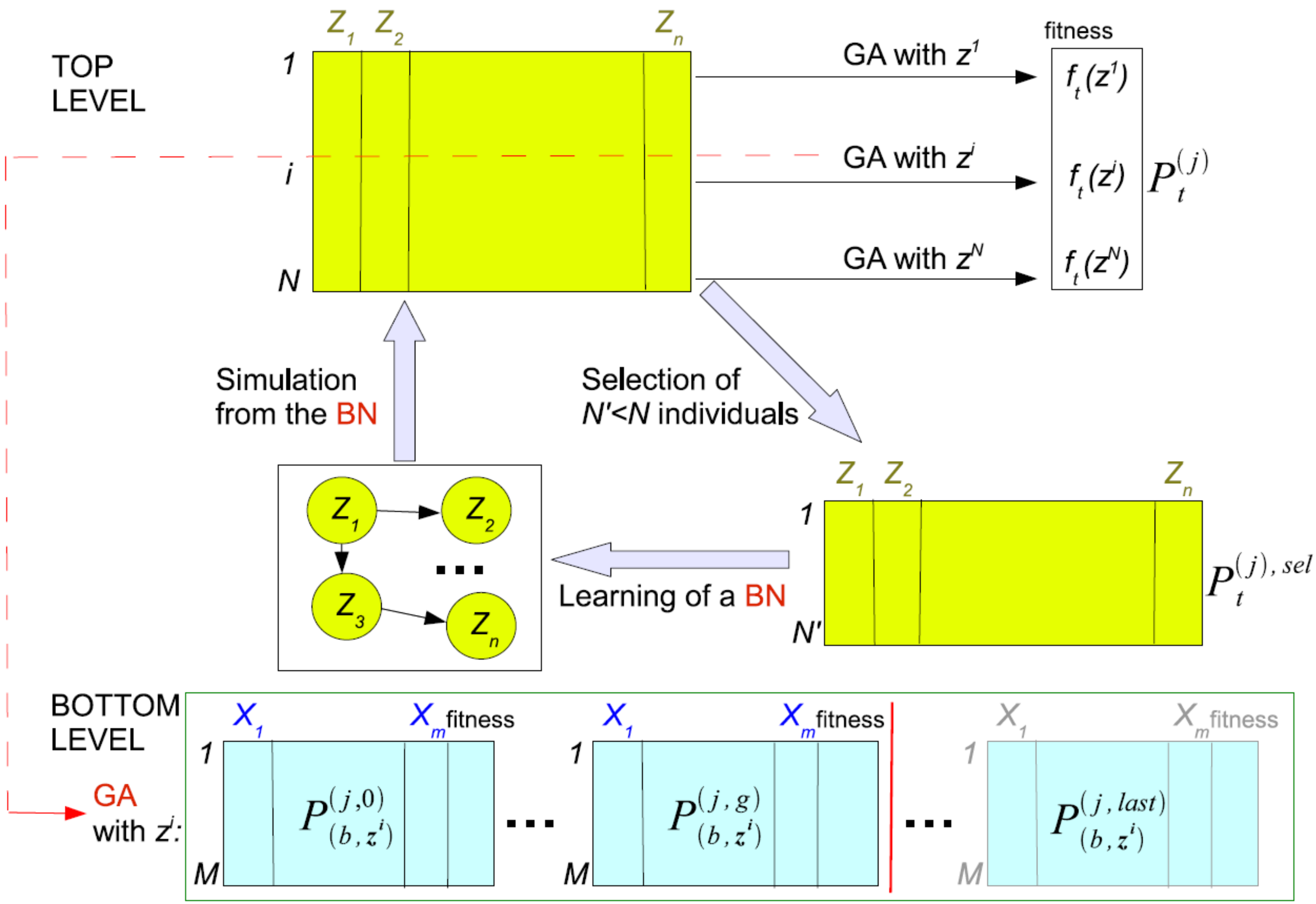
Parameter setting in evolutionary computation has two technical drawbacks:
(a)it is time consuming, and
(b) it rarely considers the dependence among parameters

- At the top level, a Bayesian network guides an intelligent search in the parameter space.

- At the bottom level, the GA is run on a fixed case (configuration of the parameters)



Self-adaptive genetic algorithm parameter control guided by an estimation of distribution algorithm



Self-adaptive genetic algorithm parameter control guided by an estimation of distribution algorithm

1. We fix the value of N' based on a result given by Friedman, N., Yakhini, Z. (1996).
On the sample complexity of learning Bayesian networks. In: *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI 96*), pp 274-282,Morgan Kaufmann

2. The Bayesian network structure learning algorithm must be based on theMDL principle
Lam, W., Bacchus, F. (1994). Learning Bayesian belief networks:
An approach based on the MDL principle. *Computational Intelligence*, 10, 269-293

3. Individuals can be simulated from the Bayesian network using
a standard procedure, like probabilistic logic sampling (PLS) (Henrion, M. (1988).
Propagating uncertainty in Bayesian networks by probabilistic logic sampling.
In J.F. Lemmer, L.N. Kanal (eds.), *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence*)

4. The stopping criterion is based on the absence of arcs in the Bayesian network

The underlying assumption is that there exist a number of generations at the bottom level, *g*, and with a population size, *M*, such that the following two pools of $N' < N$ individuals are the best ranked

Is this hypothesis true?...............we have performed an approximate test

Self-adaptive genetic algorithm parameter control guided by an estimation of distribution algorithm

algorithm

Input: Parameters: N, N' (top level) and M, r (bottom level)

$j = 0$ [*Generation counter for the top level*]

1. [*Initial sampling*]

$\mathcal{P}_t^{(j)} \leftarrow$ Sample N individuals $\mathbf{z}$ at random

For each $\mathbf{z}$ in $\mathcal{P}_t^{(j)}$

do [*Bottom level*]

Run a genetic algorithm for g generations under the $\mathbf{z}$ parameter configuration to output g populations of size M with individuals $\mathbf{x}$: $\mathcal{P}_{(b,\mathbf{z})}^{(j,0)}, \ldots, \mathcal{P}_{(b,\mathbf{z})}^{(j,g)}$

Evaluate $\mathbf{z}$ as $f_t(\mathbf{z}) = f_b(\mathbf{x}_\mathbf{z})$

do [*Top level*]

2. [*Selection*]

$\mathcal{P}_t^{(j),sel} \leftarrow$ Select $N' < N$ individuals from $\mathcal{P}_t^{(j)}$

3. [*Learning*]

$BN^{(j)} \leftarrow$ Learn a Bayesian network from $\mathcal{P}_t^{(j),sel}$

4. [*Sampling*]

$\mathcal{P}_t^{(j+1)} \leftarrow$ Sample N individuals from $BN^{(j)}$

For each $\mathbf{z}$ in $\mathcal{P}_t^{(j+1)}$

do [*bottom level*]

Run a genetic algorithm for g generations under the $\mathbf{z}$ parameter configuration to output g populations of size M with individuals $\mathbf{x}$: $\mathcal{P}_{(b,\mathbf{z})}^{(j+1,0)}, \ldots, \mathcal{P}_{(b,\mathbf{z})}^{(j+1,g)}$

Evaluate $\mathbf{z}$ as $f_t(\mathbf{z}) = f_b(\mathbf{x}_\mathbf{z})$

Repeat steps 2-4 **until** a stopping criterion is met

Output: $\mathcal{P}_t^{(j+1)}$, and $\hat{\mathbf{x}}^*$ obtained after running the GA until it converges for each $\mathbf{z} \in \mathcal{P}_t^{(j+1)}$

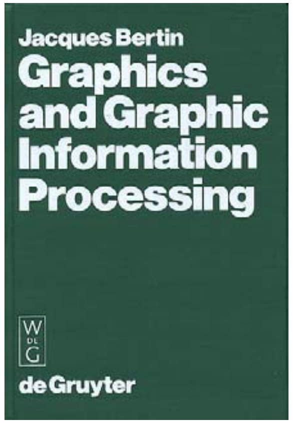
Experiments

•Optimal ordering of tables

Experiments based on a problem that consists of finding the optimal ordering of table rows and columns to make it easier to read and interpret

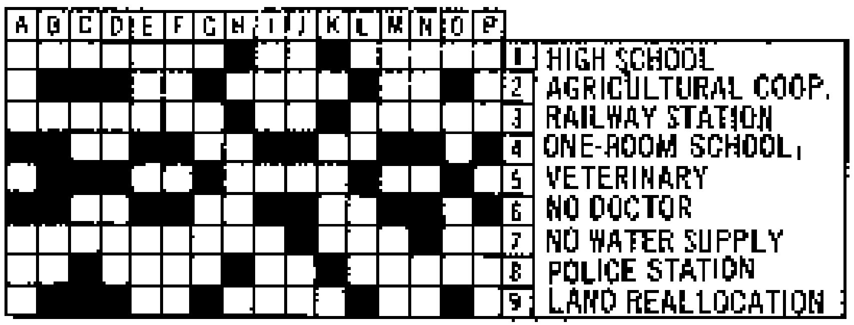
Characteristics	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
High School	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
Agricult. Coop.	0	1	1	1	0	0	1	0	0	0	0	1	0	0	1	0
Railway Station	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
One-Room-School	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1
Veterinary	0	1	1	1	0	0	1	0	0	0	0	1	0	0	1	0
No Doctor	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1
No Water Supply	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0
Police Station	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
Land Reallocation	0	1	1	1	0	0	1	0	0	0	0	1	0	0	1	0

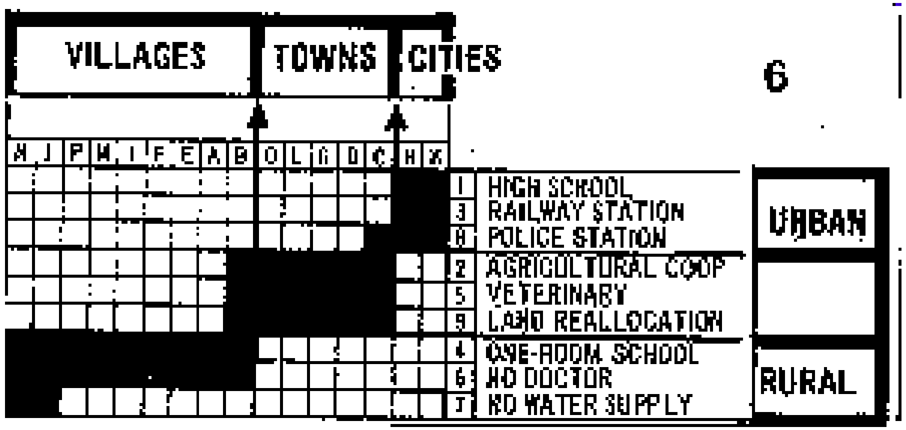
The Bertin-type tables



$$s(\pi^r(i), \pi^c(j)) = \sum_{l=\max(1,\pi^r(i-1))}^{\min(r,\pi^r(i+1))} \sum_{m=\max(1,\pi^c(j-1))}^{\min(c,\pi^c(j+1))} (e_{\pi^r(i),\pi^c(j)} - e_{\pi^r(l),\pi^c(m)})^2$$

$$f_b(\mathbf{x}) = f_b(\pi^r, \pi^c) = \sum_{i=1}^{r} \sum_{j=1}^{c} s(\pi^r(i), \pi^c(j))$$





Experiments

- Parameterization of the GA

Parameter	Possible values	Choice in Niermann (2005)
Initial population	random, heuristic	random
Crossover operator	rxc, pmx, cx, ox1, ox2, ap, vr	rxc
Crossover probability	0.50, 0.85, 0.90, 0.95	0.50
Mutation operator	2opt, dm, em, ism, tim, ivm, sm	2opt
Mutation probability	0.50, 0.10, 0.05, 0.01	0.50
Replacement policy	ets, exs, fsb	ets
Stopping criterion	fix, lock, var	fix

Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S. (1999).
Genetic algorithms for the travelling salesman problem:
A review of representations and operators. *Artificial Intelligence Review*, 13, 129-170.

Niermann (2005)
Optimizing the ordering of tables with evolutionary computation.
The American Statistician, 59, 41-46.
always used a single parameterization of the GA

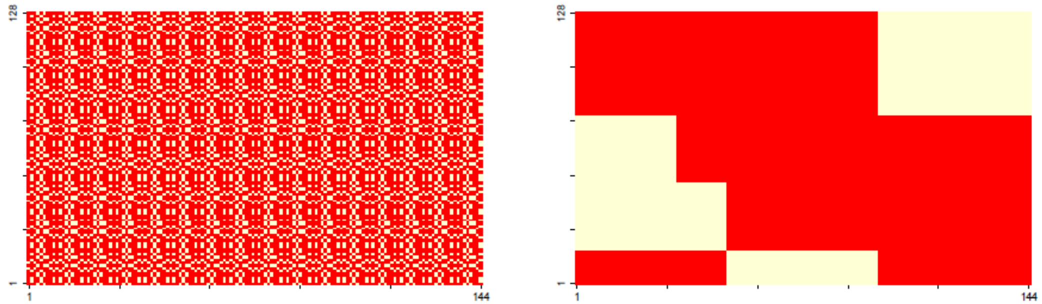
Bielza, C., Fernández del Pozo, J.A., Larrañaga, P., Bengoetxea, E. (2010).
Multidimensional statistical analysis of the parameterization of a genetic algorithmfor the optimal ordering of tables. *Expert Systems with Applications*, 37, 804-815
covered all the possible combinations of the seven parameters: 14,112 x 10 times

Experiments

•Dimensions of the set of tables

Table	No. of rows r	No. of columns c
Bertin8	32	36
Bertin32	64	72
Bertin128	128	144
Bertin256	256	288
Bertin512	512	576

Bertin128



(a) (b)

•Setting up the algorithm

Two nested levels, the top level governed by an **EDA** and a bottom level governed by a **GA**

N, is fixed at 92 individuals, N' is 46 individuals, $M = [\ln(r \cdot c)] + 1$, g?

g as the minimum number of generations when this pool of N’ individuals are equal for three consecutive $g-2$, $g-1$ and g generations: **approximate test**

Experiments

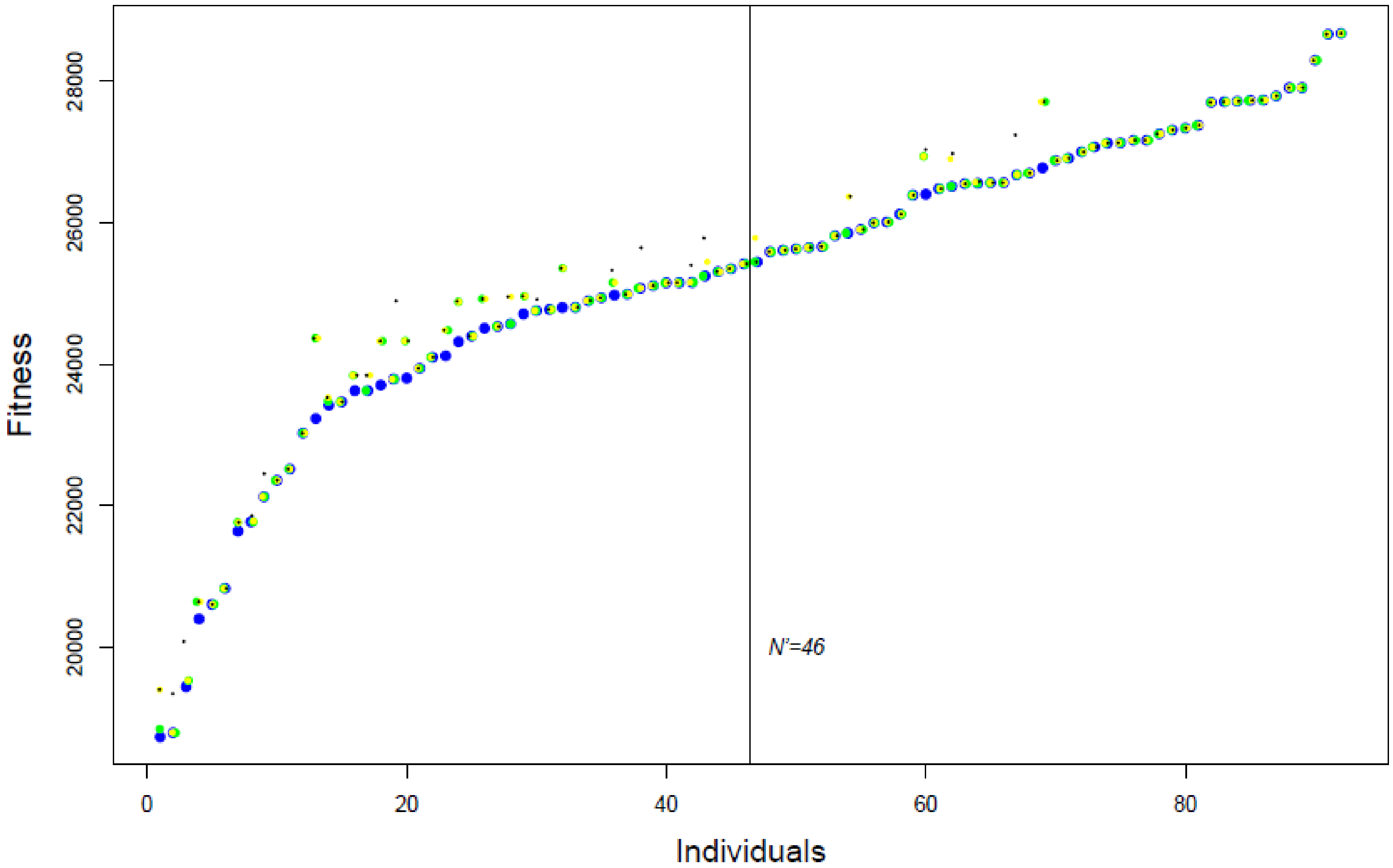
Table	Initial f_b	Best f_b in Niermann (2005)	Best f_b in Bielza *et al.* (2010)	Time in Niermann (2005)	Time in Bielza *et al.* (2010)
Bertin8	3,952	550	474	02m	104h 24m
Bertin32	16,096	2,010	978	06m	739h 32m
Bertin128	64,960	15,704	1,986	55m	28,259h 17m
Bertin256	260,992	124,232	NA	09h 51m	≈596,368h
Bertin512	1,046,272	855,428	NA	24h 06m	≈12,571,066h

Bertin128, Niermann, S. (2005).
reached a solution with a stress value of 15,704 in 55m,
whereas Bielza *et al.* (2010) obtained a stress of 1,986 in 28,259h 17m
(on a 180 eServer BladeCenter JS20 cluster, with 2,744 CPUs and main memory of 5,488 GB).

g

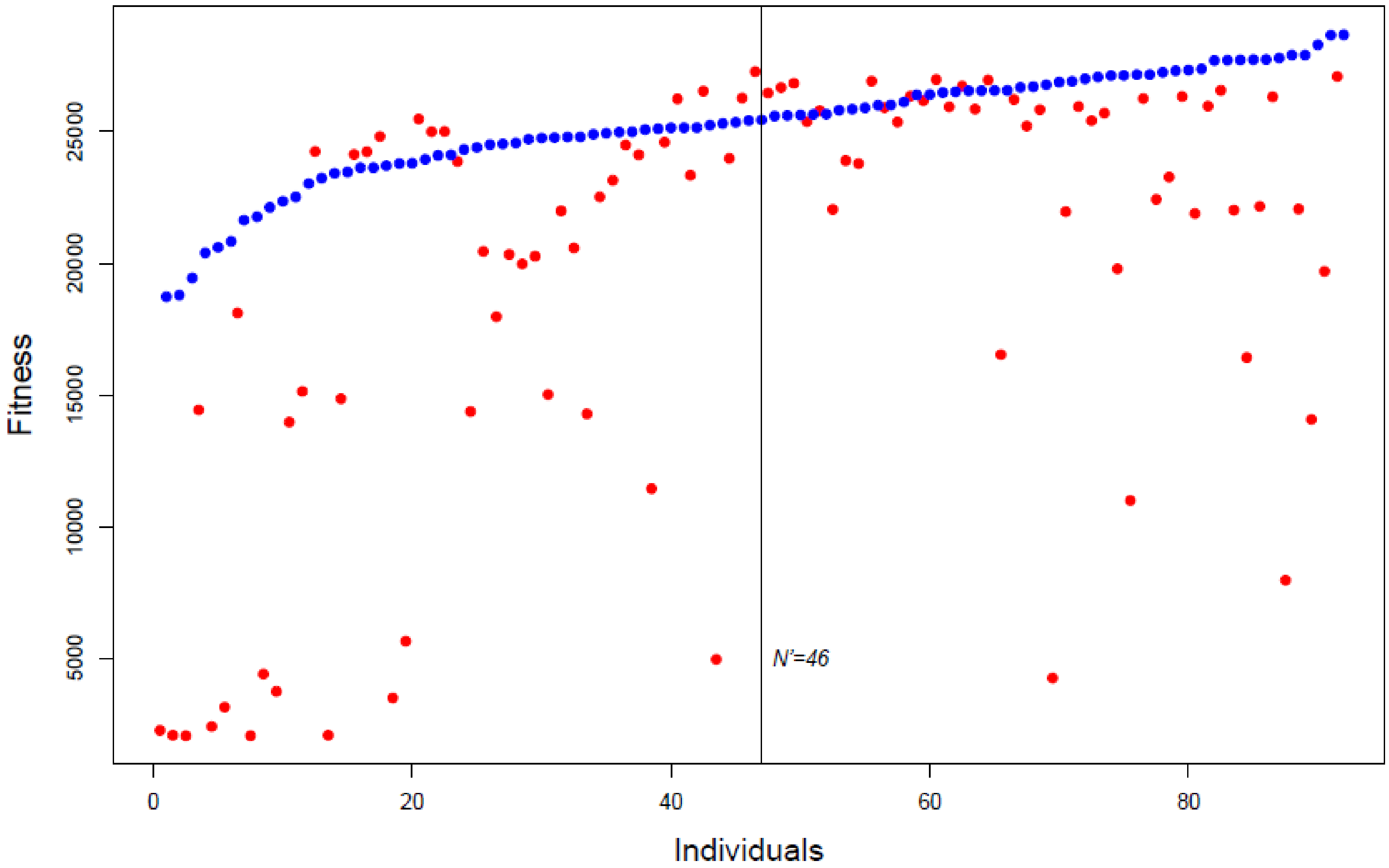
Table	EDA generations											
	1	2	3	4	5	6	7	8	9	10	11	12
Bertin8	15	39	44	41	21	35	12	19	5	3	-	-
Bertin32	10	20	48	23	16	28	8	7	7	7	7	-
Bertin128	20	11	25	48	29	16	44	15	26	13	6	6
Bertin256	18	22	35	29	23	24	7	44	13	6	-	-
Bertin512	16	23	33	29	38	7	48	39	22	14	-	-

Experiments



Fitness values of the 92 individuals when running the GA for 22 generations (**black**), 23 generations (yellow), 24 generations (**green**) and 25 generations (**blue**) for Bertin128 table, in the third generation of the EDA

Experiments

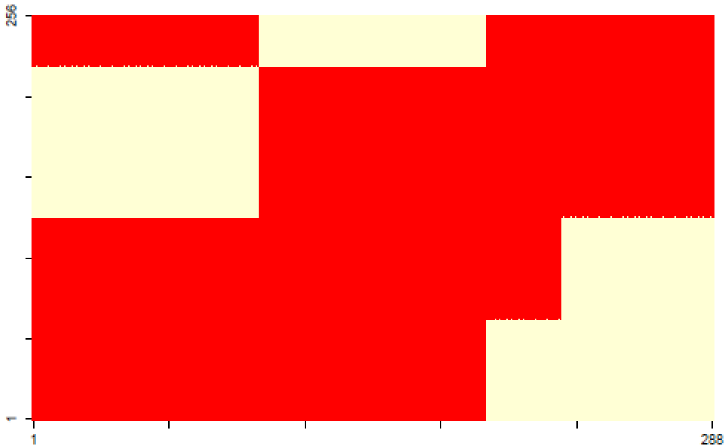


Fitness values of the 92 individuals when running the GA until convergence (**red**) and 25 generations (**blue**) for Bertin128 table

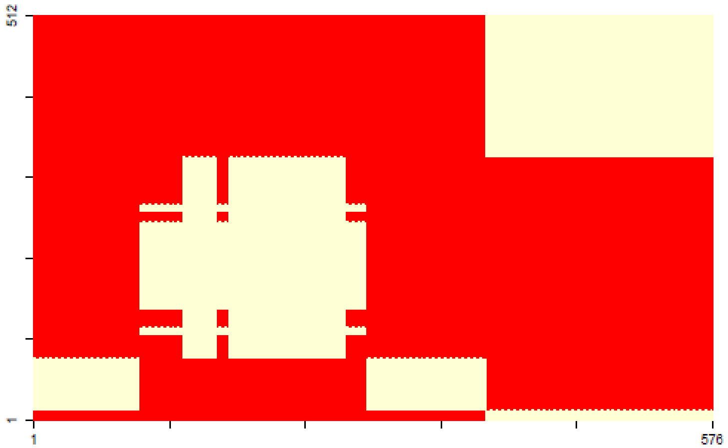
Experiments

- Results

Table	Best f_b in Bielza *et al.* (2010)	Time in Bielza *et al.* (2010)	Our best f_b	Our time
Bertin8	474	104h 24m	474	2h 00m
Bertin32	978	739h 32m	978	6h 48m
Bertin128	1,986	28,259h 17m	1,986	15h 50m
Bertin256	NA	≈596,368h	4,002	34h 52m
Bertin512	NA	≈12,571,066h	15,240	143h 11m



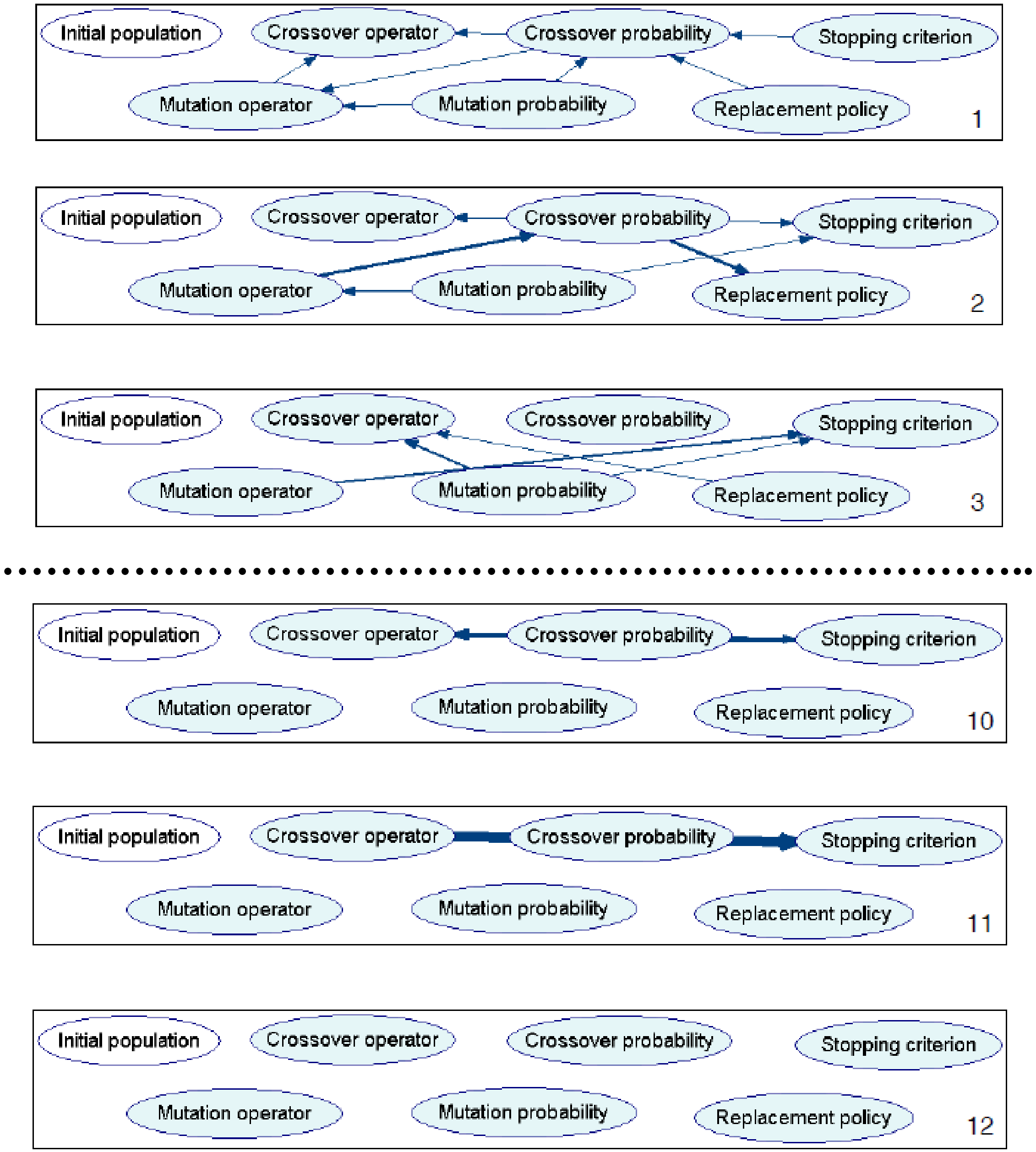
(a)



(b)

Bertin512

Experiments



- Bayesian networks learned at each EDA generation
- The initial networks are denser
- As long as the EDA evolves and finds better configurations, the networks are sparser

Conclusions

•A new approach for self-adaptive parameter control of a GA

•At a top level

Bayesian network is learned from the pool of the best parameter configurations and then new individuals are sampled according to the joint distribution function encoded by this network

•At the bottom level

GAs are run for a few generations and with small-sized populations to quickly get an estimation of the fitness function the EDA needs to further evolve

Conclusions

•Tested on a combinatorial optimization problem:

Bertin-type tables

•The capacity for processing such big problems by combining an **EDA** and a **short GA** guided by the EDA is the main value of the approach

•General enough to be applied to any other population-based evolutionary algorithm

•EDAs designed to handle continuous or even mixed variables