An overview of basic and advanced statistic techniques for calibrating and comparing algorithms

Rubén Ruiz García

INSTITUTO TECNOLÓGICO DE INFORMÁTICA
APPLIED OPTIMIZATION SYSTEMS GROUP
DEPARTMENT OF APPLIED STATISTICS, OPERATIONS RESEARCH AND QUALITY
POLYTECHNIC UNIVERSTITY OF VALENCIA, SPAIN

UNIVERSIDAD POLITECNICA DE VALENCIA

EMAA WORKSHOP, ICELAND
2006

Grupo de Investigación Operativa
http://www.upv.es/gio

Outline

- Motivation
- Preliminaries
  - Parametric vs. non-parametric
- Experimental design
  - Example
  - Analysis of the results: ANOVA
  - Checking ANOVA assumptions
  - Interactions
- Decision trees
- Conclusions

Motivation

- After two decades of publications and efforts (McGeoch, 1986) we still find the same shortcomings in algorithm experimentation and evaluation as ever
- Often is difficult, if not impossible, to ascertain which algorithm is the best in a given domain from published results and comparisons
- Just some examples taken from INFORMS Journal on Computing:

INFORMS Journal on Computing
Vol. 16, No. 3, Summer 2004, pp. 284–290
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1603 | 0284

DOI 10.1287/ijoc.1030.0039

Searching for Good Multiple Recursive Random Number Generators via a Genetic Algorithm

In designing ideal multiple recursive random number (RN) generators (MRGs), the best set of multipliers, in terms of the lattice structure of the RNs produced, is sought. As the order of the MRG increases, the number of possible sets of multipliers to be examined grows exponentially. This paper proposes a genetic algorithm for designing good MRGs. The set of multipliers associated with the MRG is encoded as a binary string. Via the operations of reproduction, crossover, and mutation, new sets of multipliers are generated. The spectral values of the MRGs are calculated to guide the search process. As an illustration, the proposed algorithm is employed to find good sets of multipliers for MRGs of orders three and four. The results are better than those derived from other studies. To conclude, this paper not only finds better MRGs of orders three and four, but also develops an algorithm for designing MRGs of higher orders.

Key words: genetic algorithms; heuristics; spectral test
History: Accepted by Michel Gendreau; received October 2000; revised June 2002; accepted March 2003.

1. Introduction

Random numbers (RNs) are widely used in operations research, statistics, engineering, and many other fields (Knuth 1997). In the literature, various kinds of RN generators have been discussed (Knuth 1997, L'Ecuyer 1996, Niederreiter 1992, Tang 2002). Among them the multiple recursive generator (MRG) is probably the most popular one due to its long period, sound statistical properties, high computational efficiency, and easy implementation (Knuth 1997, L'Ecuyer 1999a). A kth order MRG has the following form:

$$R_n = a_1 R_{n-1} + a_2 R_{n-2} + \cdots + a_k R_{n-k} \pmod{m}, \quad (1)$$

where the a_i's are the constant multipliers, m is the prime modulus, and $R_0, \ldots, R_{k-1}$ are constant seeds in $\{0, 1, \ldots, m-1\}$ but not all zero. Obviously, as the number of terms k increases, the computational burden increases accordingly. To overcome this difficulty, a two-term MRG has been studied (L'Ecuyer et al. 1993):

$$R_n = a_j R_{n-j} + a_k R_{n-k} \pmod{m}, \quad (2)$$

where $1 \leq j \leq k-1$. This two-term formula reduces considerably the computational effort. However, a tradeoff is the deterioration of the lattice structure of the RNs generated (Kao and Tang 1997a, L'Ecuyer 1997, Tang and Kao 2002).

In designing good RN generators, sets of $(a_1, \ldots, a_k)$ multipliers with the ability of generating RNs of long period and sound statistical properties are sought. For a moderate value of the prime modulus $m = 2^{31} - 1$, there are m^2 combinations of the (a_j, a_k) multipliers for the two-term MRG to be investigated. An exhaustive analysis would take years for computation. When more terms are included, a typical problem of exponential explosion occurs. Several articles have addressed this issue and two major approaches are proposed. One is random search (L'Ecuyer 1999a, L'Ecuyer et al. 1993, L'Ecuyer and Couture 1997) and the other is forward/backward systematic search (Kao and Tang 1997b, 1998). Searching for good sets of $(a_1, \ldots, a_k)$ multipliers is a combinatorial optimization problem. Several metaheuristic approaches including simulated annealing, tabu search, and genetic algorithms (GAs) have been developed to solve this type of problem. Unlike simulated annealing and tabu search, which explore the solution space sequentially, GA works with populations of solutions. It is intuitively more suitable for this RN generation problem due to its nonsequential nature. The work of Entacher et al. (2001) is probably the only study investigating the applications of GA to RN generation. The type of RN generators studied is the prime modulus linear congruential generator. Their results indicate that there is still room for further studies.

INFORMS, Journal on Computing

2004

No word about how parameters and operators have been selected

No statistical testing whatsoever

INFORMS Journal on Computing 16(3), pp. 284–290, © 2004 INFORMS

Table 1 Results of the Second-Order MRG

	Algorithm	N, G	p_c, p_m	(a_1, a_2)	Spectral	Error (%)
A1	RW	50, 200	0.65, 0.03	(2,129, −1,485,120)	0.73153	5.66
A2	SUS	100, 100	0.70, 0.02	(1,075,354, −1,333,840)	0.77544	0
B1	RW, MAX	50, 200	0.60, 0.02	(1,471,887, −36,328)	0.74459	3.98
B2	RW, AVG	100, 100	1.00, 0.02	(1,907,179, −49,347)	0.74308	4.17
B3	SUS, MAX	100, 100	0.70, 0.02	(1,075,354, −1,333,840)	0.77544	0
B4	SUS, AVG	100, 100	0.70, 0.02	(1,075,354, −1,333,840)	0.77544	0
C1	RW, MAX	100, 100	1.00, 0.04	(18,643, −1,631,826)	0.74255	4.24
C2	RW, AVG	100, 100	1.00, 0.01	(91,473,907, 790,678)	0.73422	5.32
C3	SUS, MAX	50, 200	0.60, 0.05	(922,062, −1,546,064)	0.76949	0.77
C4	SUS, AVG	50, 200	0.90, 0.02	(6,230, −1,630,587)	0.73417	5.32
Exhaustive				(1,075,354, −1,333,840)	0.77544	0
Forward/backward				(−1,538,312, −45,991)	0.68237	12.0
Random				(−37,520, −1,597,830)	0.75967	2.03

Table 2 Results of the Third-Order MRG

	Algorithm	N, G	p_c, p_m	(a_1, a_2, a_3)	Spectral	Error (%)
A1	RW	50, 300	0.70, 0.03	(−9,177,280, 12,659, −49,812)	0.75402	24.60
A2	SUS	100, 150	0.85, 0.03	(−7,615,190, −884,466, −6,260,885)	0.74133	25.87
B1	RW, MAX	50, 300	0.60, 0.04	(−85,085, −9,586,980, −29,271)	0.75857	24.14
B2	RW, AVG	50, 300	0.70, 0.03	(−9,177,280, 12,659, −49,812)	0.75402	24.60
B3	SUS, MAX	50, 300	0.95, 0.05	(−9,181, −4,772,185, −8,873,899)	0.75937	24.06
B4	SUS, AVG	100, 150	1.00, 0.03	(105,730, 2,418,337, 8,589,934)	0.75228	24.77
C1	RW, MAX	100, 150	0.85, 0.01	(24,091, −11,997,115, 47,933)	0.77492	22.51
C2	RW, AVG	50, 300	1.00, 0.01	(29,195, −14,128,181, 53,816)	0.74867	25.13
C3	SUS, MAX	50, 300	1.00, 0.01	(−818,400, −61,550, −9,256,395)	0.75389	24.61
C4	SUS, AVG	100, 150	0.90, 0.02	(7,134,497, 14,030, −6,906,092)	0.74660	25.34
Forward/backward				(45,991, −1,274,471, −8,765,239)	0.63350	36.65
Random				(−154,706, 90,222, 13,015,052)	0.73550	26.45

Table 3 Results of the Fourth-Order MRG

	Algorithm	N, G	p_c, p_m	(a_1, a_2, a_3, a_4)	Spectral	Error (%)
A1	RW	50, 300	0.60, 0.06	(−2,193, 1,302,294, 4,578, −29,020,049)	0.76210	23.79
A2	SUS	100, 150	1.00, 0.06	(40,028, −24,403,223, 10,895, 30,246,248)	0.78088	21.91
B1	RW, MAX	100, 150	0.90, 0.01	(364,165, 28,256,363, 21,710, −90)	0.76676	23.32
B2	RW, AVG	100, 150	0.80, 0.02	(−118,743, 71,995, 33,038,209, −7,724,761)	0.77578	22.42
B3	SUS, MAX	100, 150	1.00, 0.06	(40,028, −24,403,223, 10,895, 30,246,248)	0.78088	21.91
B4	SUS, AVG	50, 300	0.90, 0.02	(−8,407, 509,486, 17,162, 32,051,994)	0.79565	20.44
C1	RW, MAX	100, 150	0.80, 0.06	(−1,226,432, 42,136, 17,344, −28,633,115)	0.76095	23.91
C2	RW, AVG	50, 300	0.95, 0.04	(−42,942, 27,889,398, 35,961, −471,559)	0.76412	23.59
C3	SUS, MAX	100, 150	1.00, 0.06	(40,028, −24,403,223, 10,895, 30,246,248)	0.78088	21.91
C4	SUS, AVG	50, 300	0.85, 0.02	(57,183, 394,033, 185,447, −32,537,631)	0.76329	23.67
Random				(27,889,398, −212,938, −44,510, 1,129,066)	0.69864	30.14

Table 4 Results of the Two-Term Third-Order MRG

	Algorithm	N, G	p_c, p_m	(a_1, a_2, a_3)	Spectral	Error (%)
A1	RW	100, 100	0.60, 0.04	(48,287, 0, 1,898,747)	0.15533	1.37
A2	SUS	50, 200	0.80, 0.02	(0, 75,387, −1,833,888)	0.15472	1.76
B1	RW, MAX	50, 200	1.00, 0.04	(0, 922,458, −1,614,649)	0.15600	0.95
B2	RW, AVG	100, 100	0.85, 0.02	(0, −928,039, 1,624,420)	0.15556	1.23
B3	SUS, MAX	100, 100	0.70, 0.04	(0, −928,039, 1,624,420)	0.15556	1.23
B4	SUS, AVG	50, 200	0.80, 0.03	(950,634, 0, 1,606,195)	0.15550	1.26
C1	RW, MAX	50, 200	0.65, 0.03	(0, 922,458, −1,614,649)	0.15600	0.95
C2	RW, AVG	100, 100	0.80, 0.01	(0, 1,668,596, −2,095,105)	0.15479	1.71
C3	SUS, MAX	50, 200	0.80, 0.01	(0, −220,843, −1,833,888)	0.15564	1.17
C4	SUS, AVG	50, 200	0.60, 0.04	(1,821,444, 0, 533,934)	0.15473	1.75
Forward/backward				(1,774,779, 0, 45,991)	0.14811	5.96
Random				(0, 1,518,729, 23,875)	0.12806	18.69

Barrage of tables with average values

A New Genetic Algorithm for the Quadratic Assignment Problem

In this paper we propose several variants of a new genetic algorithm for the solution of the quadratic assignment problem. We designed a special merging rule for creating an offspring that exploits the special structure of the problem. We also designed a new type of a tabu search, which we term a *concentric tabu search*. This tabu search is applied on the offspring before consideration for inclusion in the population. The algorithm provided excellent results for a set of 29 test problems having between 30 and 100 facilities.
(*Quadratic Assignment; Heuristics; Genetic Algorithm; Memetic Algorithm; Tabu Search*)

1. Introduction

The *quadratic assignment problem* is considered one of the most difficult optimization problems to solve optimally. A rich body of literature exists on heuristic approaches for its solution. The problem is defined as follows.

A set of n possible sites are given and n facilities are to be located on these sites, one facility at each site. Let c_{ij} be the cost of moving items for one unit of distance from facility i to facility j and d_{ij} be the distance from site i to site j. The cost f to be minimized over all possible permutations, calculated for an assignment of facility i to site $p(i)$ for $i = 1, \dots, n$, is

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{p(i)p(j)}. \tag{1}$$

The first heuristic algorithm proposed for this problem was CRAFT (Armour and Buffa 1963), which is a descent heuristic. More recent algorithms use metaheuristics such as tabu search (Battiti and Tecchiolli 1994, Skorin-Kapov 1990, Taillard 1991), simulated annealing (Burkard and Rendl 1984, Wilhelm and Ward 1987), genetic algorithms (Ahuja et al. 2000, Fleurent and Ferland 1994, Tate and Smith 1995), ant-colonies search (Gambardella et al. 1999), or specially designed heuristics (Drezner 2002, Li et al. 1994). For a complete discussion and list of references see (Burkard 1990, Cela 1998, and Taillard 1995).

In this paper we first describe genetic algorithms in general, present the two merging processes used in the proposed genetic algorithms, and p three different procedures to be applied to off before consideration for inclusion in the popu (which we term a *post-merging procedure*, or Such algorithms are sometimes referred in the ture as *Memetic algorithms* (Radcliffe 1994). In S 3 we present extensive computational compa between all proposed variants. We summari results and propose future research in Section 4

2. Genetic Algorithms

Genetic algorithms have proven to be quite cessful for the solution of combinatorial prol For reviews see Goldberg (1989) and Salhi (Proposed genetic algorithms for the solution quadratic assignment problem are Ahuja et al. (Fleurent and Ferland (1994), and Tate and (1995).

INFORMS Journal on Computing © 2003 INFORMS
Vol. 15, No. 3, Summer 2003, pp. 320–330

0899-1499/03/1503/032
1526-5528 electron

INFORMS, Journal on Computing

2003

Improper experimentation for

fixing parameters and operators

No statistical testing at all

After many experiments with moderately sized problems ($30 \le n \le 64$) we selected a population size of 100. The number of generations for the concentric tabu was set to $\max\{20n, 1000\}$. The number of generations for the descent and the simple tabu was set to double these values. We noticed an improvement in the results of the algorithms when the population size is increased (and the number of generations is increased proportionally). However, in order to stay within reasonable run times, we opted to experiment with a fixed population size of 100.

Table 1 Comparison Between Different Merging Procedures

Problem	Best Known	No Genetic			TS/FF			Cohesive			Scrambled		
		†	‡	*	†	‡	*	†	‡	*	†	‡	*
Kra30a	88900	20	0	0.45	20	0	0.43	20	0	0.33	20	0	0.32
Kra30b	91420	20	0	0.44	20	0	0.43	20	0	0.33	20	0	0.31
Nug30	6124	20	0	0.49	20	0	0.47	20	0	0.37	20	0	0.33
Tho30	149936	20	0	0.49	20	0	0.46	20	0	0.35	20	0	0.33
Esc32a	130	20	0	0.52	20	0	0.51	20	0	0.35	20	0	0.37
Esc32b	168	20	0	0.43	20	0	0.42	20	0	0.30	20	0	0.30
Esc32c	642	20	0	0.29	20	0	0.28	20	0	0.27	20	0	0.27
Esc32d	200	20	0	0.34	20	0	0.33	20	0	0.28	20	0	0.28
Esc32h	438	20	0	0.34	20	0	0.34	20	0	0.29	20	0	0.29
Ste36a	9526	6	0.114	0.95	8	0.063	0.91	19	0.005	0.55	16	0.021	0.65
Ste36b	15852	20	0	0.91	20	0	0.86	20	0	0.61	20	0	0.68
Ste36c	8239.11	1	0.107	0.95	14	0.024	0.89	14	0.039	0.59	18	0.010	0.66
Tho40	240516	3	0.034	1.34	3	0.025	1.32	5	0.010	0.98	5	0.015	0.91
Sko42	15812	20	0	1.66	20	0	1.60	20	0	1.15	20	0	1.20
Sko49	23386	10	0.032	2.89	14	0.022	2.81	17	0.009	2.13	18	0.007	2.16
Wil50	48816	3	0.024	3.08	7	0.014	2.94	18	0.002	1.99	18	0.002	1.85
Sko56	34458	0	0.041	5.01	0	0.046	4.83	19	0.001	3.24	17	0.002	3.29
Sko64	48498	3	0.043	9.09	1	0.035	8.96	20	0	5.85	19	0.000	6.01
Esc64a	116	20	0	3.21	20	0	3.19	20	0	3.05	20	0	3.10
Sko72	66256	0	0.120	15.76	0	0.115	15.50	10	0.014	8.36	7	0.013	7.74
Sko81	90998	0	0.124	25.52	0	0.112	25.43	5	0.014	13.30	4	0.019	12.78
Sko90	115534	0	0.139	41.81	0	0.126	41.63	4	0.011	22.35	2	0.019	19.52

† Number of times out of 20 that best-known solution obtained,
‡ Percentage of average solution over the best-known solution,
* Time in minutes per run,

Some key parameters set after running a handful of instances and comparing averages

Table 3 **Comparison Between Genetic Algorithms Using Different PMPs**

Problem	Best Known	Descent			Simple Tabu			Concentric Tabu		
		†	‡	Time*	†	‡	Time*	†	‡	Time*
# of Runs:		200			100			20		
Kra30a	88900	162	0.253	0.06	93	0.089	0.09	20	0	0.33
Kra30b	91420	124	0.037	0.06	79	0.019	0.09	20	0	0.33
Nug30	6124	160	0.013	0.06	99	0.001	0.10	20	0	0.37
Tho30	149936	192	0.009	0.06	100	0	0.10	20	0	0.35
Esc32a	130	144	0.569	0.06	100	0	0.07	20	0	0.35
Esc32b	168	200	0	0.05	100	0	0.08	20	0	0.30
Esc32c	642	200	0	0.05	100	0	0.06	20	0	0.27
Esc32d	200	200	0	0.05	100	0	0.06	20	0	0.28
Esc32h	438	200	0	0.05	100	0	0.06	20	0	0.29
Ste36a	9526	49	0.246	0.08	37	0.149	0.12	19	0.005	0.55
Ste36b	15852	195	0.015	0.08	100	0	0.14	20	0	0.61
Ste36c	8239.11	73	0.142	0.08	59	0.066	0.12	14	0.039	0.59
Tho40	240516	4	0.069	0.13	4	0.042	0.23	5	0.010	0.98
Sko42	15812	173	0.014	0.16	96	0.001	0.30	20	0	1.15
Sko49	23386	2	0.107	0.28	12	0.062	0.48	17	0.009	2.13
Wil50	48816	26	0.038	0.25	42	0.011	0.47	18	0.002	1.99
Sko56	34458	63	0.054	0.42	59	0.007	0.72	19	0.001	3.24
Sko64	48498	69	0.051	0.73	65	0.019	1.23	20	0	5.85
Esc64a	116	200	0	0.40	100	0	0.49	20	0	3.05
Sko72	66256	1	0.112	0.93	9	0.056	1.45	10	0.014	8.36
Sko81	90998	0	0.087	1.44	0	0.058	2.18	5	0.014	13.30
Sko90	115534	3	0.139	2.31	4	0.073	3.51	4	0.011	22.35
Sko100a	152002	7	0.114	3.42	3	0.070	5.11	5	0.018	33.55
Sko100b	153890	6	0.096	3.47	17	0.042	5.11	10	0.011	34.05
Sko100c	147862	2	0.075	3.22	11	0.045	4.69	5	0.003	33.80
Sko100d	149576	0	0.137	3.45	0	0.084	5.15	1	0.049	33.90
Sko100e	149150	4	0.071	3.31	17	0.028	4.70	18	0.002	30.67
Sko100f	149036	1	0.148	3.55	1	0.110	5.25	1	0.032	35.74
Wil100	273038	0	0.076	3.51	3	0.043	5.24	5	0.002	33.11

† Number of times out of the corresponding number of runs that best-known solutions obtained.
‡ Percentage of average solution over the best-known solution.
* Time in minutes per run.

Comparison among algorithms done similarly !!!

Motivation

- Recent examples such as these can be found in many other OR journals where new algorithms and/or techniques are shown
- Some areas, like for example routing and scheduling are even worse as statistical techniques (even simple paired tests) are scarcely used

Motivation

- The same old questions:
  - Which design options should I use?
  - Why some options work better than others?
  - Is the performance similar for all types of instances?
  - Am I correctly calibrating my algorithm?
  - Is my algorithm better than competitors?
- …are still answered incorrectly in most published work
- …some of them are not even raised or dealt with at all

Motivation

- The result of this is well known (Hooker, 1994, 1995, among many others):
  - Questionable findings, questionable contribution
  - Results almost impossible to reproduce
  - Hardly any possible generalization
  - Vague reports on results
  - No insight on *why* the proposed methods work
  - No insight on how instance characteristics affect performance
  - No quantification of what parts of the proposed method are actually helping
  - No indication of interactions…

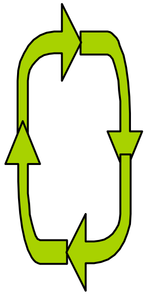
Motivation

- Clearly, we already know enough to put an end to all this
- There is plenty of published papers and reports where all these problems are addressed and where tools are given to avoid them (McGeoch, 1992; Barr et al., 1995; McGeoch, 1996; Rardin and Uzsoy, 2001, Bartz-Beielstein, 2003...)

Motivation

- In this talk I will try to overview the basics of correct and sound statistical experimentation
- It will not be by any means comprehensive…
- …but it will be really applied with hands-on examples
- We will skip some important issues
- I will stress the usage of parametric statistics whenever possible
- Towards the end I will briefly introduce some advanced statistical techniques

Preliminaries

- What we usually want:
  
  - To know is this or that feature of the algorithm we are building is worthwhile (**design**)
  - To comprehend why something works and why doesn't, specially when using different instances (**analysis**)
  - To convince everybody with sound results that our algorithm is better (**comparison**)
- This triad of questions can be answered with the same tools in a sound statistical way

Preliminaries

- We will work with samples (instances)
- But we want sound conclusions: generalization over a given population (all possible instances)
- Thus we need STATISTICAL INFERENCE
- Very important:
  - Descriptive statistics are nice but one should never infer from a median, average or percentile
  - Sadly, and as we have seen, this is exactly what we find in the literature: "*the proposed algorithm is better than algorithm X because it gives better average results on some instances (out of a benchmark of 20)*"

Preliminaries
Parametric vs. non-parametric

- As we know:
  - Parametric inferential tests do have some assumptions and requirements on your data
  - This is necessary so that the theoretical statistical models we adopt are appropriate for making inferences
  - Non-parametric tests are "distribution-free"

- Then, Why don't we just use non-parametric tests?

Preliminaries
Parametric vs. non-parametric

- There are very, very few “completely assumption free” statistical tests
- Non-parametric tests can be too over conservative
  - The differences in the means have to be strong in order to find statistically significant differences

- This might not sound too bad… but digging a little bit more…

Preliminaries
Parametric vs. non-parametric

- We will be contrasting the following hypothesis:
  - H_0 = There are no differences in the response variable
- Truth table:

	Hypothesis testing over H_0	
Nature of H_0	No reject	Reject
True	☺	Error Type I —
False	Error Type II —	☺ (POWER)

Preliminaries
Parametric vs. non-parametric

- Power of a test: 1- _
  - Probability of rejecting H_0 when it's false
  - The power increases, among other things with the sample size
- _ it's very difficult to estimate a priori
- It is desired to have a low _, a low _ and a high power

Preliminaries
Parametric vs. non-parametric

- With all this in mind:
  - If the differences in the means are not strong enough the non-parametric tests have very little power
  - This means that we will be having high _:
    - The non-parametric tests tend to not accept H_0 when it's false
    - You will be wrongly answering negatively to the triad of questions!!

Preliminaries
Parametric vs. non-parametric

- Parametric testing:
  - Robust: you really have to depart from the assumptions in order to find trouble
  - If sample is large enough (>100) CLT takes care of many things
  - If the sample is large, using non-parametric makes very little sense…
  - …but interestingly, many significance tests in non-parametric statistics are based on asymptotic (large samples) theory

Preliminaries
Parametric vs. non-parametric

- You really need large data samples…
  - If you really find that your algorithm is a mere 3% better than all other algorithms with very few samples then you have done something wrong or you cannot really generalize
  - Or if you have an algorithm that is a 300% better than all others in a small sample probably you do not need statistics
- … therefore, after all this the question now is reversed:
- **“Why use non-parametric tests?”**

Experimental design

- Among the basic techniques, experimental design can help us answer all the triad of questions
- All other basic questions can also be adequately answered
- Easy to understand, easy to use:

DESIGN OF EXPERIMENTS (DOE)

Experimental design

- The experimental design is just a few guidelines to carry out the experiments so to obtain results as clearly and as efficiently as possible
- There are many types of experiments and many associated techniques
- In my opinion, one does not really need to go far in DOE before reaching our goals
- Computer experimentation is a very easy environment as far as DOE goes (Bartz-Beielstein, 2003)

Experimental design

- Some special characteristics of computer experiments as far as DOE goes:
  - Reproducibility to the bit (re-using the random seed)
  - Malleable environment in most cases (input can be controlled)
  - A priori knowledge present most times
  - “Cheap” and fast data collection
  - Systematic errors in experimentation are unlikely to occur and easy to avoid

Experimental design

- Response variable: The aim of the experiment; characteristic that we want to study: percentage deviation from optima, time needed to a given solution/quality…
- Controlled Factor: variables, options, parameters that we CAN control and that might affect the response variable
  - Quantitative: Probability of crossover (levels)
  - Qualitative: Type of crossover (variants)

Experimental design

- Treatment: a given combination of the levels/variants of the different controlled factors
- Experience: the execution of a treatment and the associated resulting value of the response variable
- Replicate: when a given treatment is executed more than once
- Non controlled factor: All other factors (known or not) that we can NOT control

Experimental design

- The easiest design is called FULL FACTORIAL
  - All the combinations of levels of all factors are experimented
  - Powerful design
  - Easy analysis of the results
  - Exponential growth on the number of experiences as the number of factors and/or levels grows
  - The results are usually presented in a table

Experimental design

Treatment	Factors			Replicates		
	F1	F2	F3	Y_1	Y_2	Y_3
1	1	1	1	Y_{1111}	Y_{1112}	Y_{1113}
2	2	1	1	Y_{2111}	Y_{2112}	Y_{2113}
3	1	2	1	Y_{1211}	Y_{1212}	Y_{1213}
4	2	2	1	Y_{2211}	Y_{2212}	Y_{2213}
5	1	1	2	Y_{1121}	Y_{1122}	Y_{1123}
6	2	1	2	Y_{2121}	Y_{2122}	Y_{2123}
7	1	2	2	Y_{1221}	Y_{1222}	Y_{1223}
8	2	2	2	Y_{2221}	Y_{2222}	Y_{2223}

Experimental design

- The order in which the treatments (experiences) are carried out should be RANDOMIZED
- Probably this is not needed in computer algorithms but memory leaks and in general degradation of computer resources represent a very dangerous lurking variable
- Lurking variables: non-controlled factors that affect controlled factors in a systematic and consistent way
- This generates a non controlled structure in the data, which kills the experimentation

Experimental design Example

- Example of a screening experiment
  - Design and calibration of an Iterated Greedy metaheuristic. Application to the permutation flowshop problem (Stützle, Pranzo and Ruiz, in preparation):

S0=Construct_Initial_Secuence(); **How to construct it?**

S1=Local_Search(S0); **Do we need local search?**

While NOT(TerminationCriterion()) do

  S2=Partially_Destruct(S1); **How to destruct? How much to destruct?**

  S3=Construct_Secuence(S2); **How to reconstruct?**

  S4=Local_Search(S3); **Do we need local search?**

  If Acceptance_Criterion(S4,S1) then S1=S4 **How to accept?**

Experimental design Example

- Response variable:
  - Minimization of the percentage deviation over the best solution known for a set of HARD instances
- Controlled factors:
  - Type of initialization (2 variants): heuristic and random
  - Type of destruction (2 variants): random and blocked

Experimental design Example

- Controlled factors (cont):
  - Type of reconstruction (2 variants): greedy and random
  - Application of local search (2 variants): no, yes
  - Acceptance criterion (2 variants): SA, descent
  - Iterations for acceptance (2 levels): 1, 5
  - Number of jobs to destruct (2 levels): 4, 6
- 7 factors at two levels: full factorial of 128 tests

Experimental design
Example

- In this case is better to run a half fraction: 27-1=64 treatments: Fractional factorial experiment
  - Resolution VII: allows us to study interactions of three factors with ease
- Very important to consider:
  - 3 groups of instances, 10 instances each= 30 instances
  - All instances have 20 machines and differ in the number of jobs (50, 100 and 200)
  - 5 replicates per treatment
- 64 treatments · 30 instances · 5 replicates = 9600 data
- RANDOMIZE + USE VRT!!

Experimental design Example

- Crucial: Termination criteria set at a maximum elapsed CPU time that depends on the instance (n·m·30 ms)

IG TEST															
Algorithm Parameters															
Alg	Initialization	Destruction_T	Reconstruction_T	LS	Acceptance_C	Iterations_Acc	Destruct	Instance	n	m	replicate	Objective	Time (micros)	BOUNDS	RPD
44	1	0	1	0	1	5	6	Ta103	200	20	5	11980	120000000	11281	6,1962592
53	1	1	0	1	0	1	4	Ta110	200	20	1	11427	120000000	11288	1,23139617
24	0	1	0	1	1	5	6	Ta105	200	20	3	11379	120000000	11259	1,06581402
25	0	1	1	0	0	1	6	Ta087	100	20	4	6574	60000000	6268	4,88194001
13	0	0	1	1	0	1	6	Ta054	50	20	2	3769	30000000	3723	1,23556272
24	0	1	0	1	1	5	6	Ta104	200	20	5	11459	120000000	11275	1,63192905
5	0	0	0	1	0	1	4	Ta052	50	20	4	3721	30000000	3704	0,45896328
37	1	0	0	1	0	1	6	Ta105	200	20	4	11327	120000000	11259	0,60396128
64	1	1	1	1	1	5	6	Ta110	200	20	4	11478	120000000	11288	1,6832034
23	0	1	0	1	1	1	4	Ta051	50	20	4	3898	30000000	3850	1,24675325
29	0	1	1	1	0	1	4	Ta102	200	20	3	11405	120000000	11203	1,80308846
23	0	1	0	1	1	1	4	Ta105	200	20	4	11318	120000000	11259	0,52402522
64	1	1	1	1	1	5	6	Ta101	200	20	1	11400	120000000	11195	1,83117463
35	1	0	0	0	1	1	6	Ta085	100	20	5	6428	60000000	6314	1,80551156
64	1	1	1	1	1	5	6	Ta060	50	20	1	3823	30000000	3756	1,78381257
36	1	0	0	0	1	5	4	Ta060	50	20	2	3831	30000000	3756	1,99680511
62	1	1	1	1	0	5	4	Ta085	100	20	4	6435	60000000	6314	1,91637631
37	1	0	0	1	0	1	6	Ta108	200	20	4	11487	120000000	11334	1,34992059
64	1	1	1	1	1	5	6	Ta090	100	20	3	6547	60000000	6434	1,75629468
14	0	0	1	1	0	5	4	Ta086	100	20	4	6487	60000000	6364	1,9327467
43	1	0	1	0	1	1	4	Ta086	100	20	4	6622	60000000	6364	4,05405405
8	0	0	0	1	1	5	4	Ta088	100	20	3	6508	60000000	6401	1,67161381
52	1	1	0	0	1	5	6	Ta086	100	20	4	6526	60000000	6364	2,54556882
29	0	1	1	1	0	1	4	Ta056	50	20	1	3716	30000000	3681	0,95082858

Experimental design
Analysis of the results: ANOVA

- Sir Roland Fisher, 1930
- The ANOVA (analysis of variance) is one the most powerful statistical tools available
- The term ANOVA is a source of confusion: detects differences on means by analyzing the variance!
- The ANOVA is a statistical model where the variation in the response variable is partitioned into components that correspond to the different sources of variation (factors)

Experimental design
Analysis of the results: ANOVA

- Let's study the results
- ANOVA TABLE

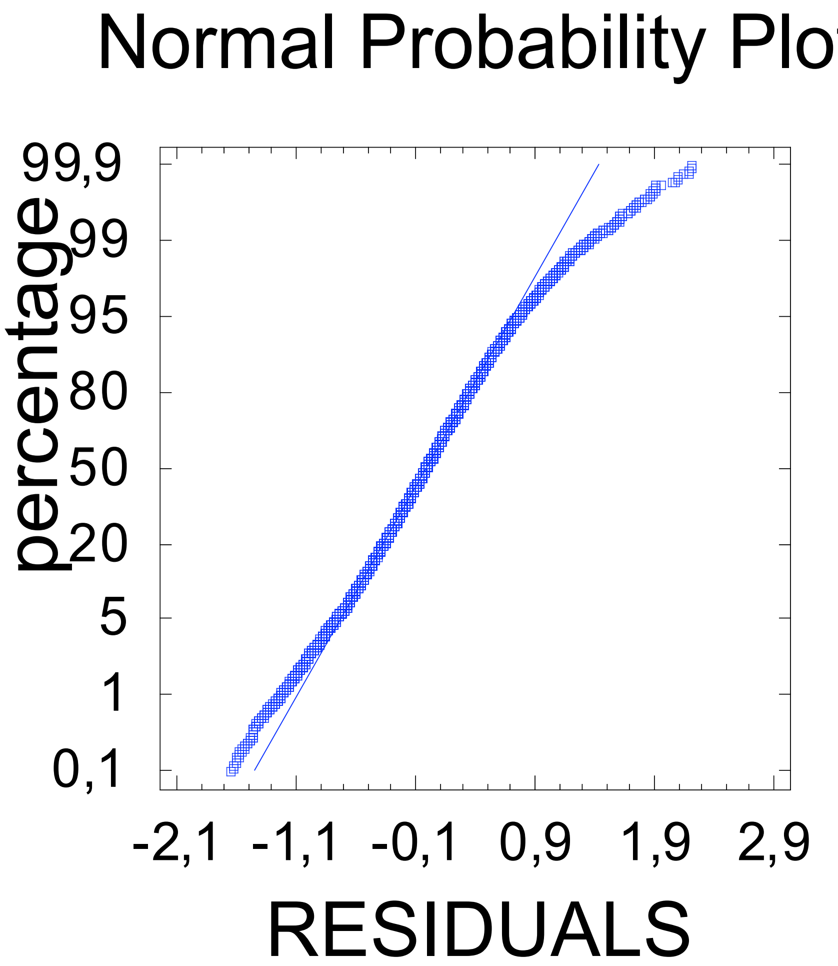
Analysis of Variance for RPD - Type III Sums of Squares

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
MAIN EFFECTS					
A:Acceptance_C	7,26506	1	7,26506	27,62	0,0000
B:Destruct	389,076	1	389,076	1479,08	0,0000
C:Destruction_T	50,0663	1	50,0663	190,33	0,0000
D:Initialization	60,7802	1	60,7802	231,06	0,0000
E:Iterations_Acc	393,743	1	393,743	1496,82	0,0000
F:LS	12444,9	1	12444,9	47309,62	0,0000
G:n	133,771	2	66,8856	254,27	0,0000
H:Reconstruction_T	4286,73	1	4286,73	16296,09	0,0000
I:replicate	0,402254	4	0,100563	0,38	0,8215

Experimental design
Checking ANOVA assumptions

- Before actually starting, we have to check the three main assumptions of the ANOVA: normality, homocedasticity and independence of the residuals
- Checking normality:
  - Outlier analysis
  - Distribution fitting of the data to a normal distribution, Normal Probability plots…
  - Numerical tests are very strict and normally they will reject the hypothesis that the data comes from a normal distribution

Experimental design
Checking ANOVA assumptions



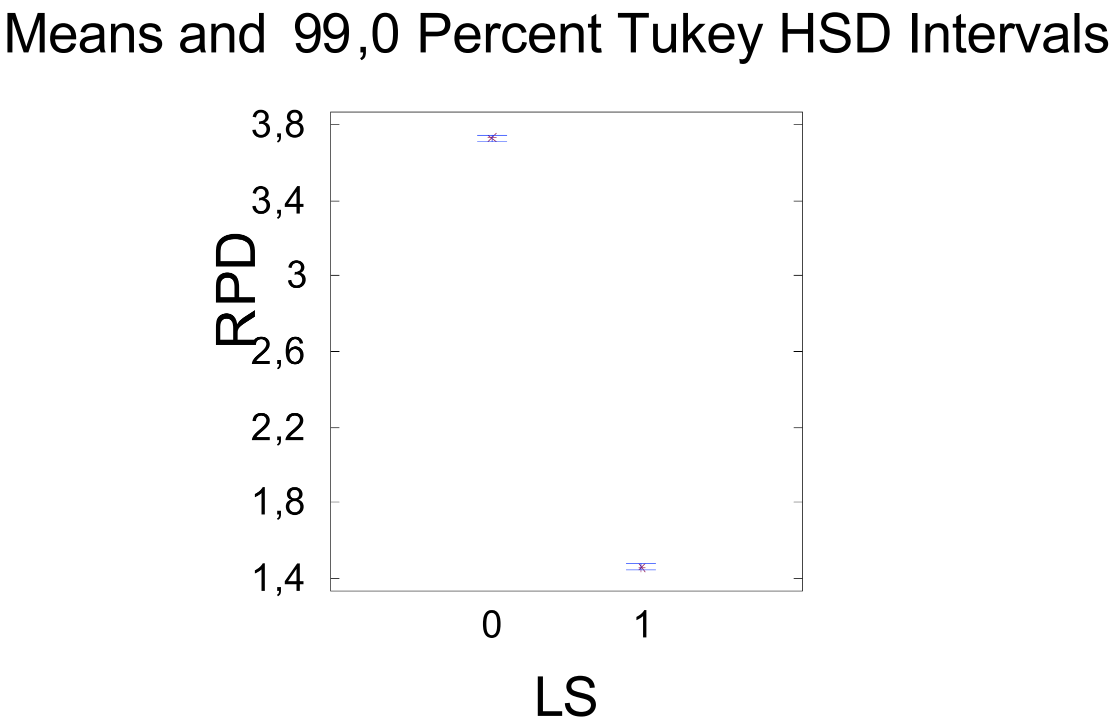
- Ooops!
  - Non normality
  - Studies support the fact that ANOVA is very robust wrt to normality
  - Still there is much that we can do

Experimental design
Checking ANOVA assumptions

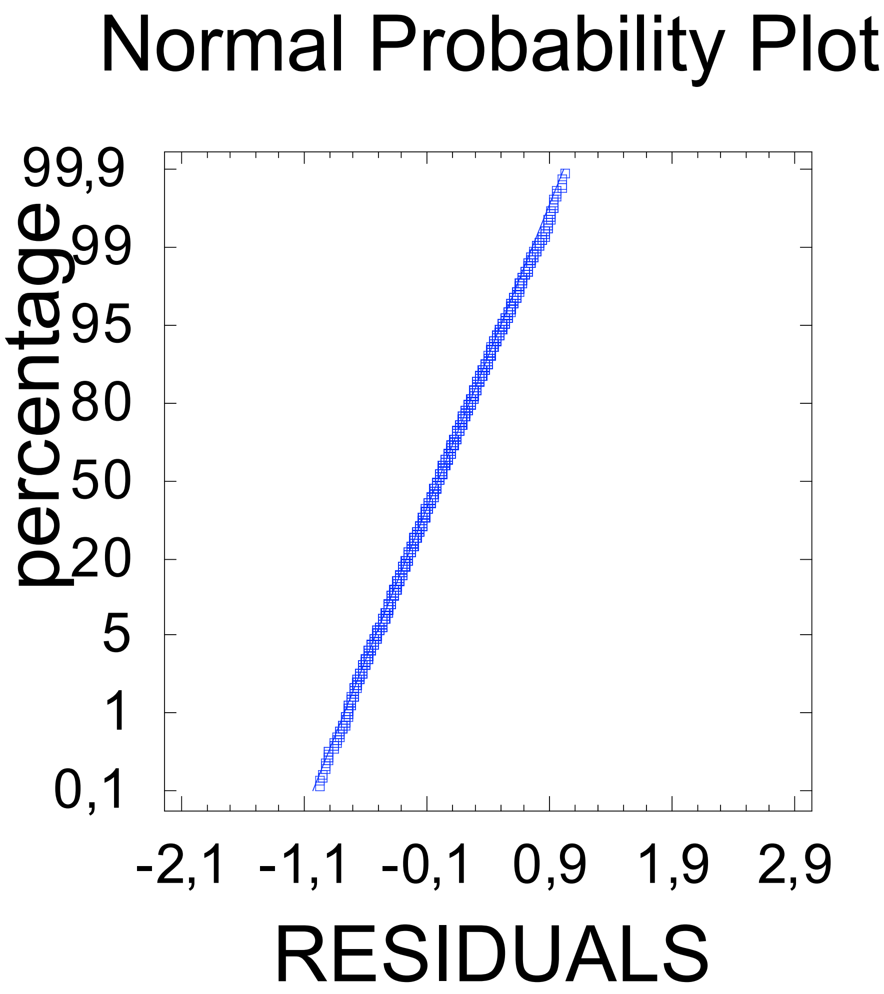
- Sources of trouble regarding normality:
  - Presence of severe outliers
    - Outliers should not be eliminated as the environment is controlled. Check for bugs or other potential problems in the code
  - Factors or levels with excessive effect
    - There is no need to test what is evident
  - “Clustering”
    - Totally different behavior on the results depending on some levels or factors

Experimental design
Checking ANOVA assumptions

- According to the ANOVA table, the factor LS has a very large effect
- Means plot: a simple plot, usually along with confidence intervals suitable for multiple comparisons:



Experimental design
Checking ANOVA assumptions

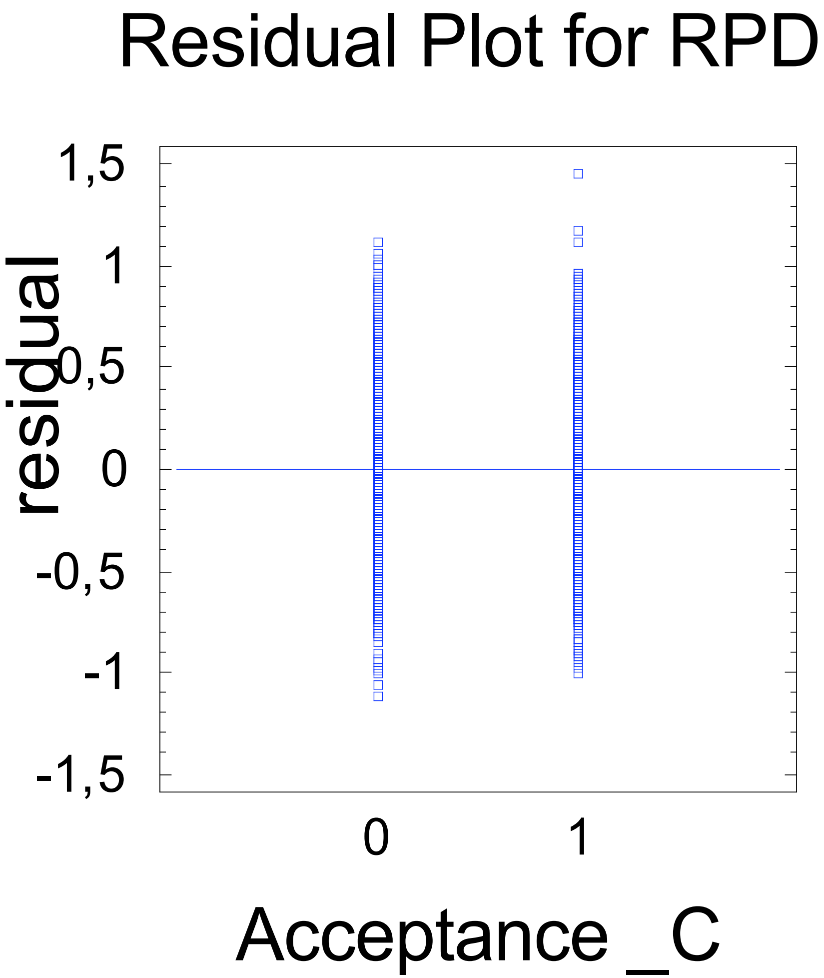


- Much better now
- Many transformations possible
- I would not worry unless aberrant plot

Experimental design
Checking ANOVA assumptions

- Checking homocedasticity:
  - Study the dispersion of the residuals with respect to the levels of all factors
    - Some levels or factors might result in higher or lower variance
  - Study the dispersion of the residuals with respect to the values of the response variable
    - Probably higher or lower values of the response variable might result in higher or lower variance

Experimental design
Checking ANOVA assumptions



- No problem
- It has to be repeated for every factor
- Also for the response variable

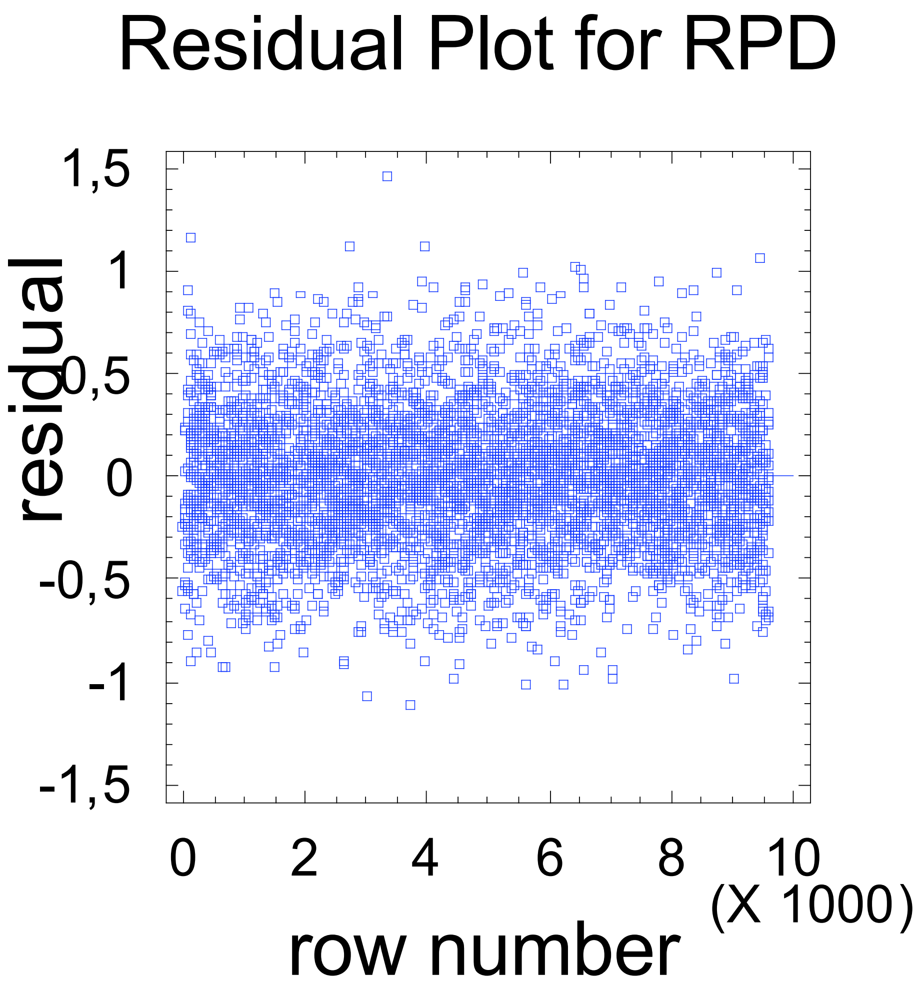
Experimental design
Checking ANOVA assumptions

- Sources of trouble regarding homocedasticity:
  - A level of a factor resulting in more variance
    - Disregard the level in the experiment
  - More variance in the "hard" instances
    - Separated ANOVAS, one for each group of instances
  - Increased variance as response variable increases (decreases)
    - Properly select the response variable!

Experimental design
Checking ANOVA assumptions

- ANOVA is very sensitive to lack of independence
- Checking independence of the residual:
  - Plot of the dispersion of residuals over run number or time
    - We should expect the residual to be independent from time
  - Analyze the residual looking for self correlation patterns
    - The residual should be “white noise”

Experimental design
Checking ANOVA assumptions

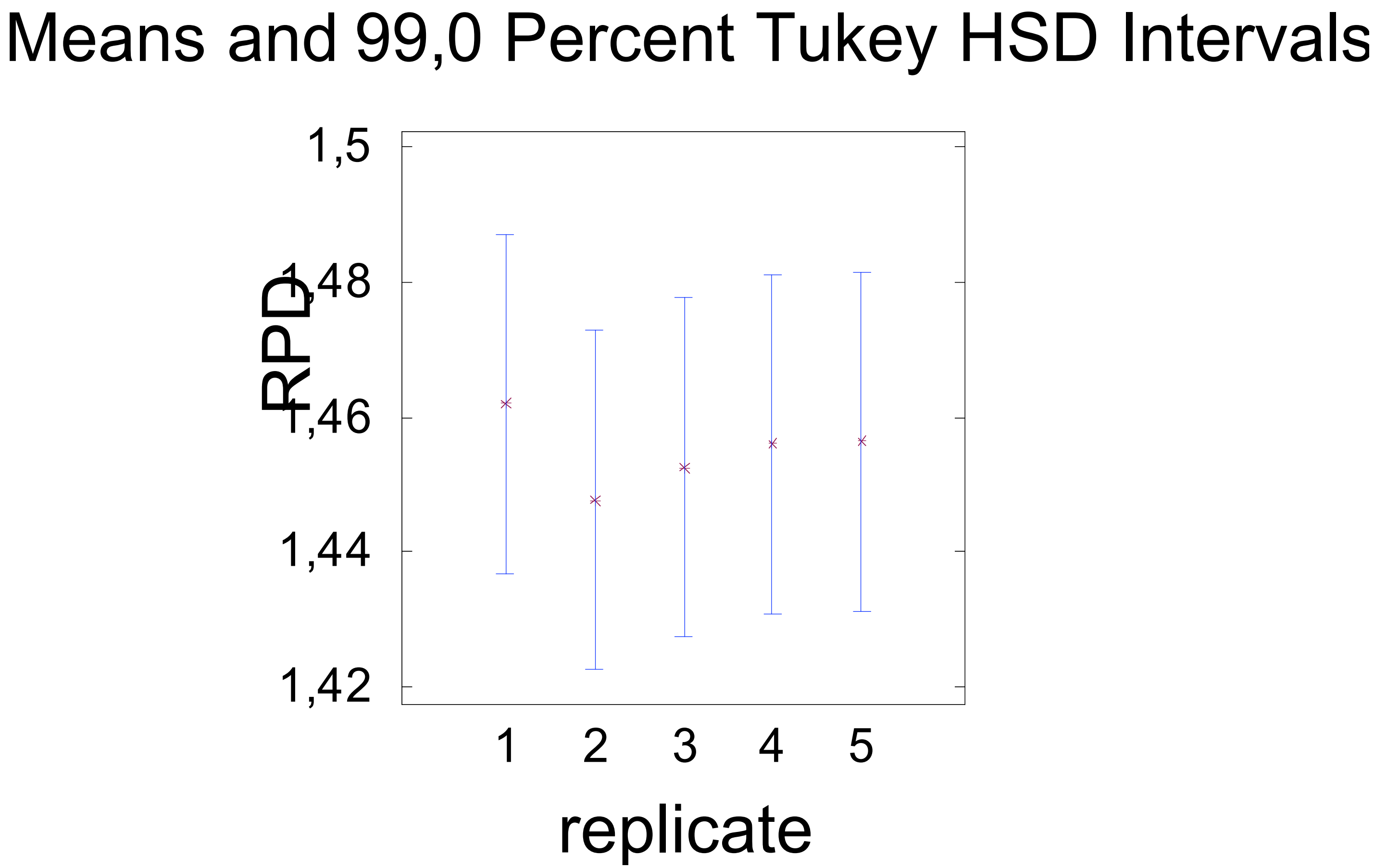


- No problem
- Controlled environment: no lurking variables

Experimental design
Checking ANOVA assumptions

- Sources of trouble regarding independence of the residual:
  - Residual bigger over time
    - Experiences run in batch mode, computer resources degrading over time
  - Structure in the residual
    - Randomization or “shuffling” of the experiences
    - ANOVA model NOT complete

Experimental design
Checking ANOVA assumptions



Experimental design
Checking ANOVA assumptions

- Checking assumptions:
  - If the experiment is carried out with care…
  - if there are sufficient samples…
  - and if the technique is applied correctly…
  - … there should not be any problem
- If everything else fails
  - Then use a non-parametric test and hope to obtain something!

Experimental design
Analysis of the results: ANOVA

- With large samples the p-value tends to be close to zero
  - If the sample size is large enough (infinity) any difference in the means of the factors, no matter how small, will be significant
- Real vs. Statistical significance (Montgomery and Runger, 2002)
  - Study factors until the improvement in the response variable is deemed small

Experimental design
Analysis of the results: ANOVA

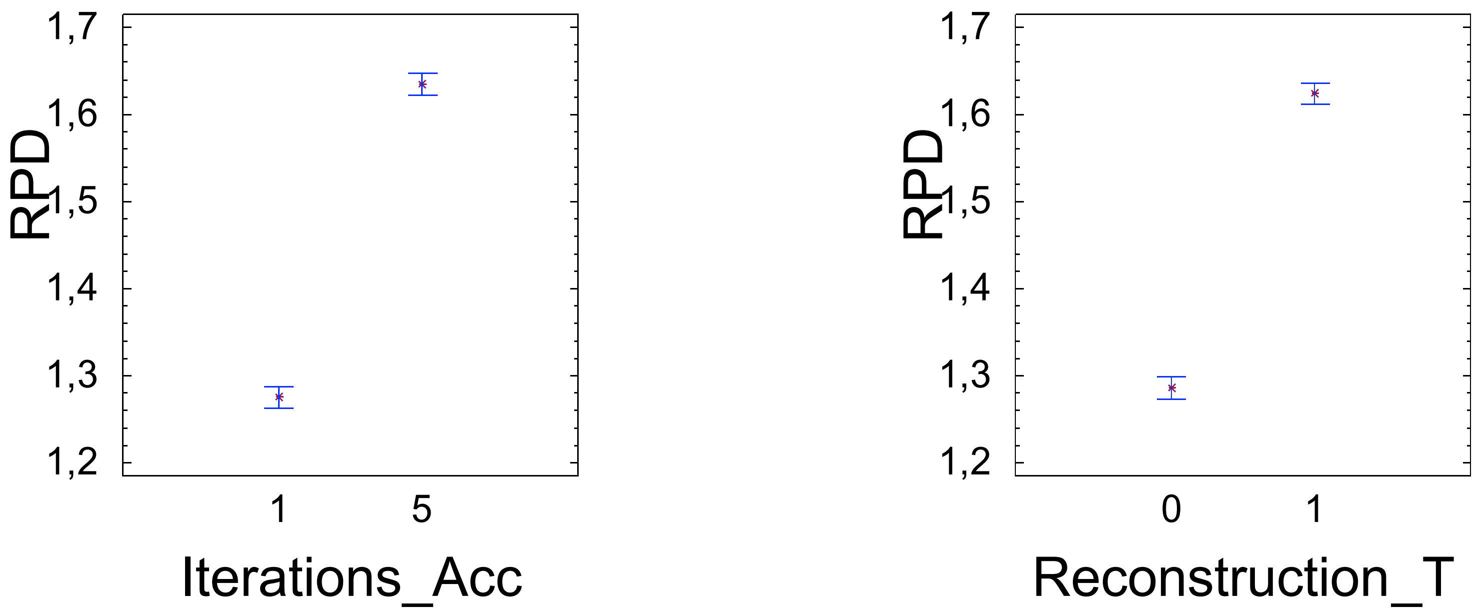
Analysis of Variance for RPD - Type III Sums of Squares

Source	Sum of Squares	Df	Mean Square	F-Ratio	P-Value
MAIN EFFECTS					
A:Acceptance_C	1,14754	1	1,14754	10,23	0,0014
B:Destruct	33,0077	1	33,0077	294,17	0,0000
C:Destruction_T	0,264526	1	0,264526	2,36	0,1247
D:Initialization	0,0288163	1	0,0288163	0,26	0,6123
E:Iterations_Acc	155,4	1	155,4	1384,96	0,0000
F:n	120,115	2	60,0573	535,25	0,0000
G:Reconstruction_T	137,248	1	137,248	1223,20	0,0000

- Examine the factors by F-Ratio value:
  - Iterations_Acc, Reconstruction_T, n, Destruct, Acceptance_C

Experimental design
Analysis of the results: ANOVA

Means and 99,0 Percent Tukey HSD Intervals



Experimental design
Interactions

- A very interesting feature of the ANOVA is that one can study interactions
- For algorithm design, the most interesting interactions are those between the design options and the characteristics of the instances
- “Short experiments”, “One factor at a time” or even modern racing algorithms (Birattari et al., 2002) do not allow the study of interactions

Experimental design
Interactions

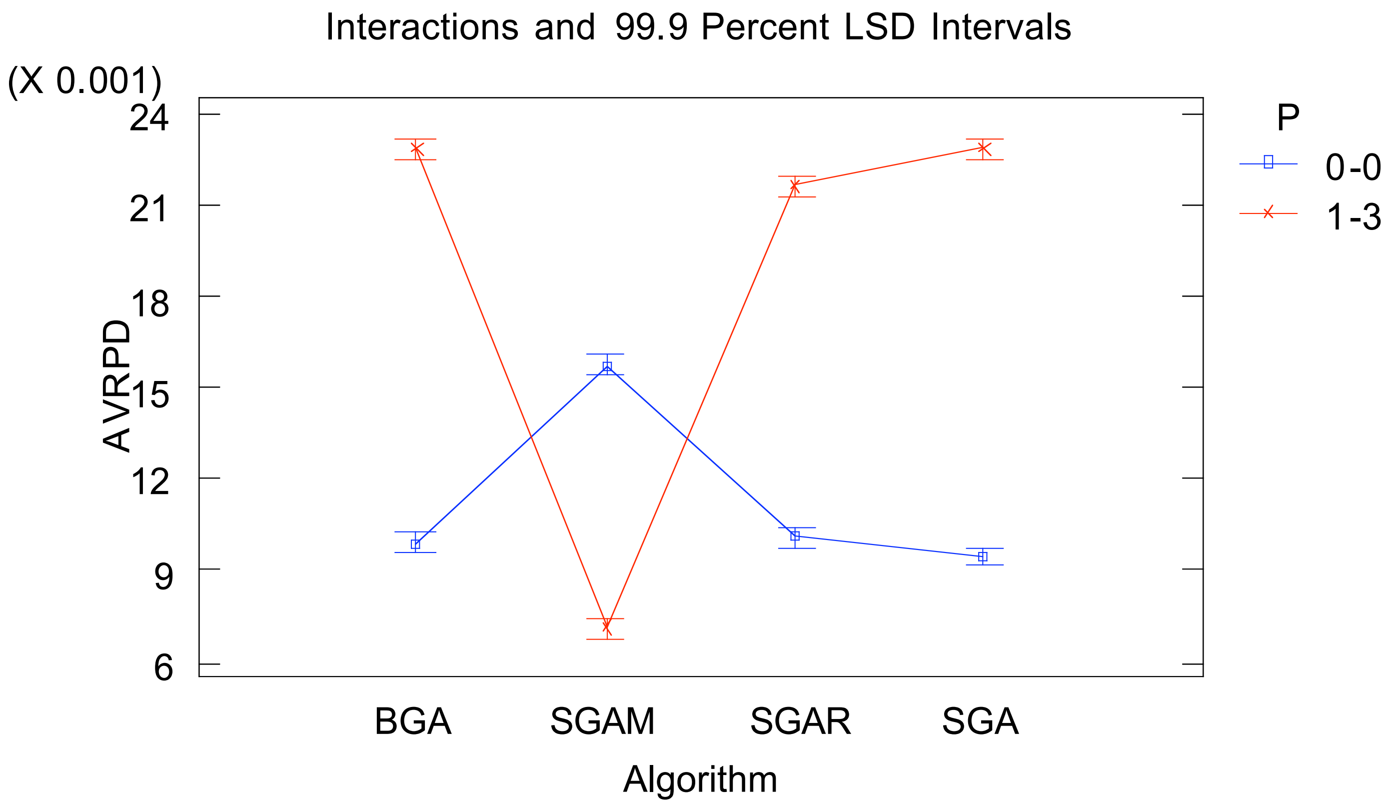
- Let us work with another example (Ruiz et al., in press at C&OR, Thijs and Ruiz, in preparation)
- Heuristics and genetic algorithms for realistic scheduling problems
- 10 controlled factors depicting different characteristics of the instances
- Very large datasets and comprehensive experiments: we want to know why algorithms work

Experimental design Interactions

Factor		Small (9,216)	Large (3,072)
Number of jobs	*n*	5,7,9,11,13,15	50,100
Number of stages	*m*	2, 3	4, 8
Number of unrelated parallel machines per stage	*mi*	1, 3	2, 4
Distribution of the release dates for the machines	*rm*	0, U[1,200]	0, U[1,200]
Probability for a job to skip a stage	*F*	0%, 50%	0%, 50%
Probability for a machine to be eligible	*E*	50%, 100%	50%, 100%
Distribution of the setup times as a percentage of the processing times	*S*	U[25,74], U[75,125]	U[25,74], U[75,125]
Probability for the setup time to be anticipatory %	*A*	U[0,50], U[50,100]	U[0,50], U[50,100]
Distribution of the lag times	*lag*	U[1,99], U[_99,99]	U[1,99], U[_99,99]
Number of preceding jobs	*P*	0, U[1,3]	0, U[1,5]

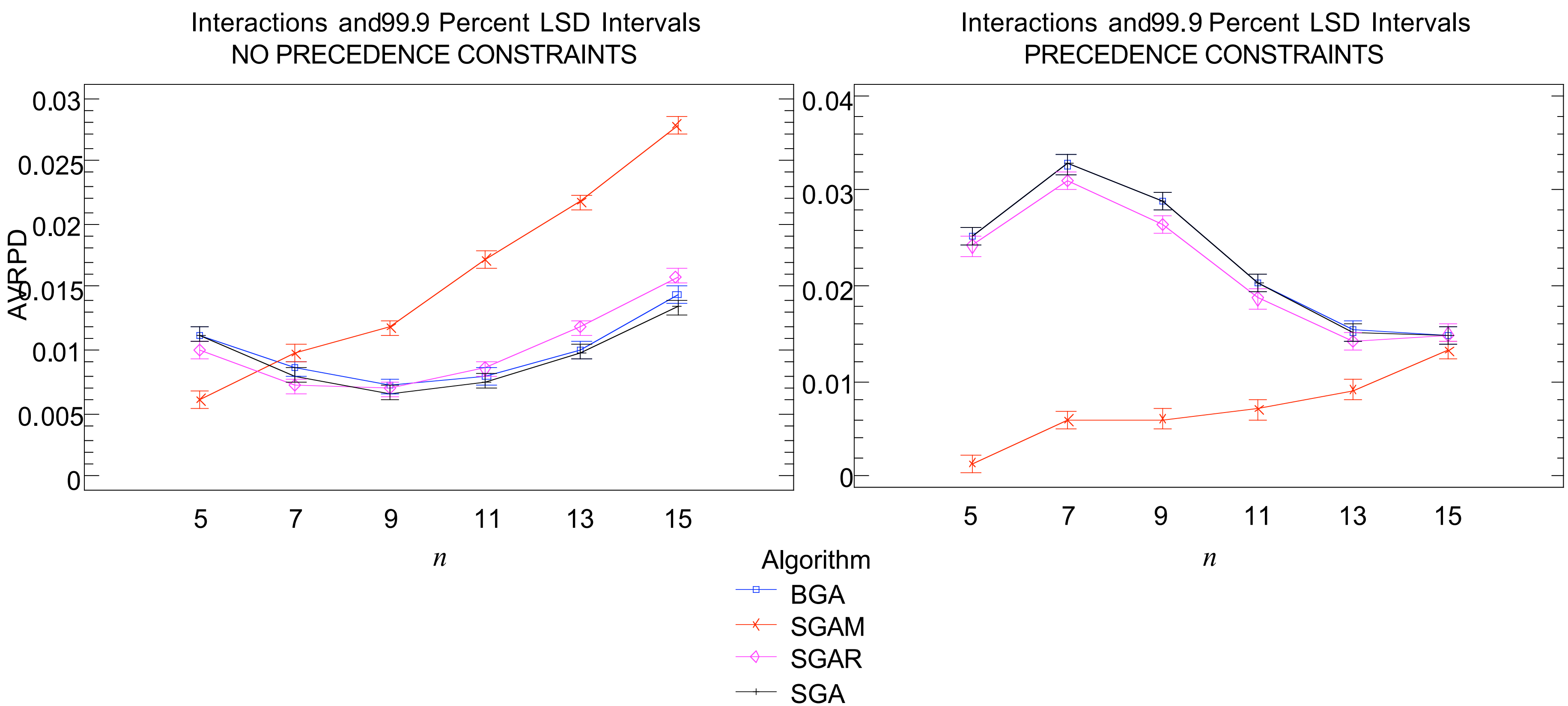
Experimental design Interactions

- Example of a very strong 2-factor interaction:



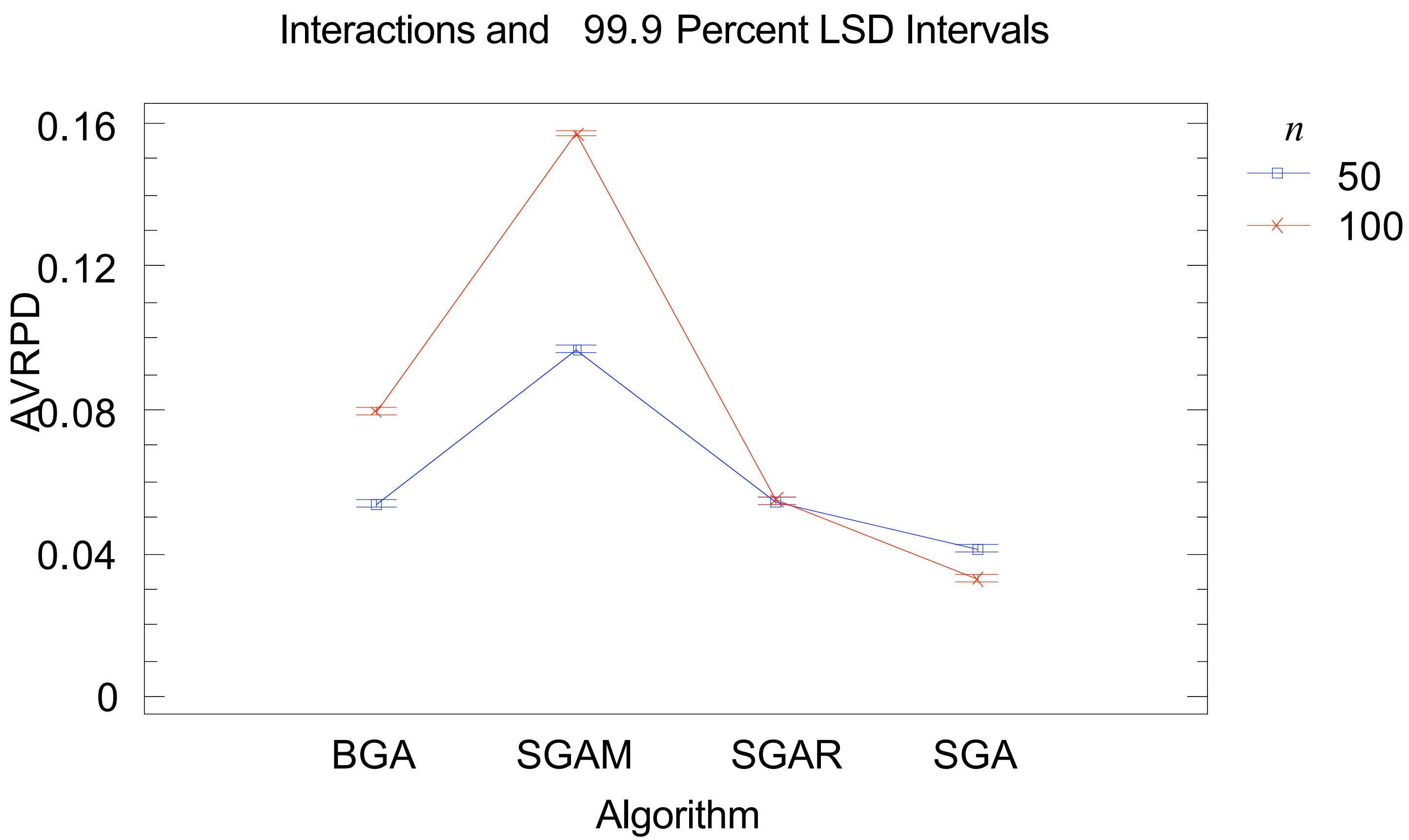
Experimental design
Interactions

- Example of a very strong 3-factor interaction:



Experimental design
Interactions

- Another example of 2-factor interaction



Decision trees

- In some cases, the nature of the data that we obtain does not allow for a parametric analysis no matter the number of samples
- A clear example comes from categorized response variables
- Non-parametric tests (Wilcoxon, Kruskal-Wallis) are very limited as regards the study of interactions
- Decision trees and Automatic Interaction Detection (AID) tools are non-parametric and at the same time perfect for interaction study

Decision trees

- AID (Morgan and Sonquist, 1963) recursively bisects experimental data according to one factor into mutually exclusive and exhaustive sets that describe the response variable in the best way. AID works on an interval scaled response variable and maximizes the sum of squares between groups by means of an F statistic
- We use an improved version called Exhaustive CHAID from Biggs et al. (1991) that allows multi-way splits and significance testing. The result is a decision tree

Decision trees

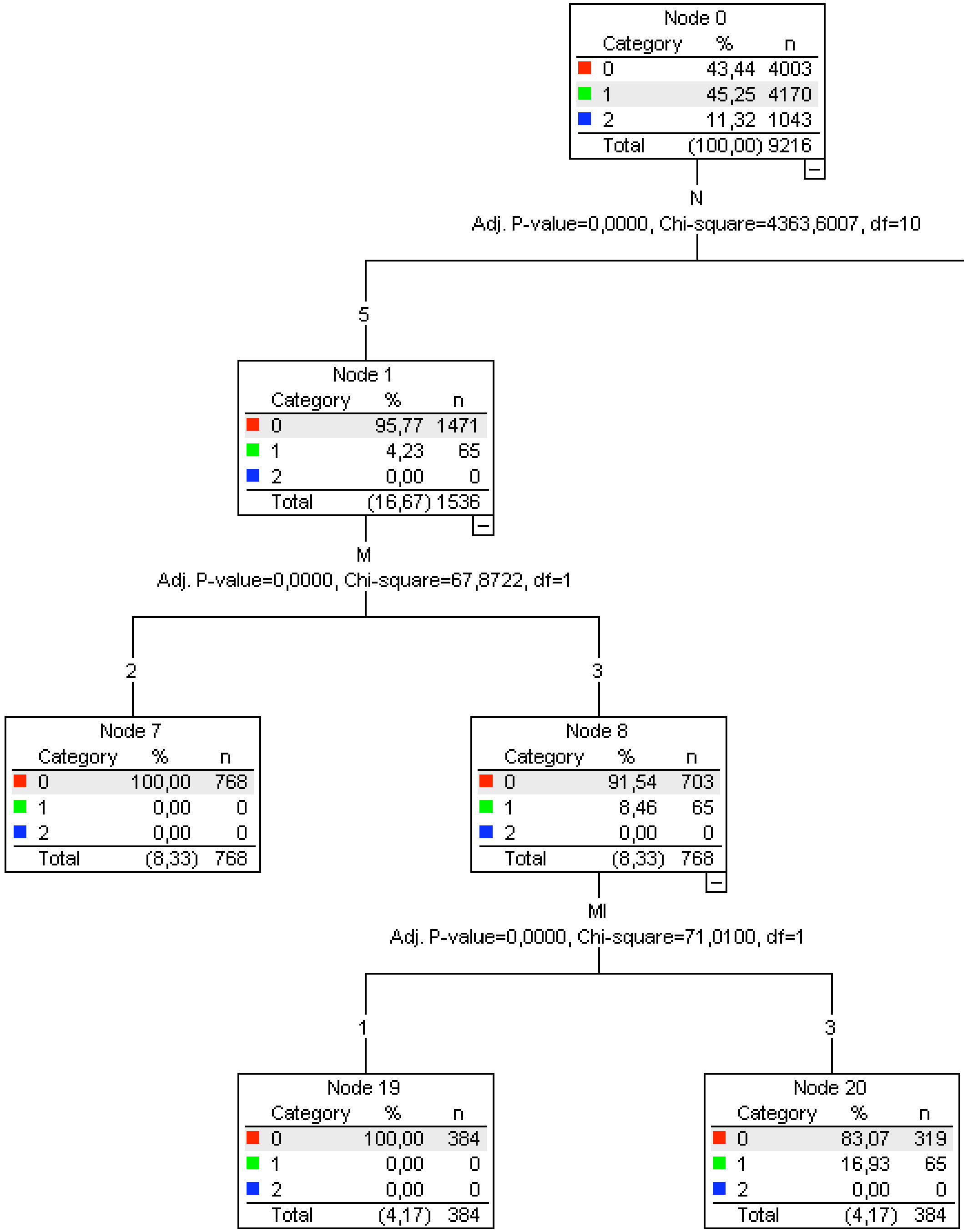
- Decision trees are very common in social and health sciences
- I have not seen them applied to algorithm design and calibration
- An example of categorical variable
  - Analysis of the performance of a MIP model on the previous dataset of 9,216 instances. Three different possible results:
  - 0: Optimum solution found within the time limit
  - 1: Time limit reached, solution found
  - 2: Time limit reached, no solution found

Decision trees

- First clumsy attempt: a table with averages

	m	2		3	
n	m_i	1	3	1	3
5	%Opt	100.00	100.00	100.00	90.36
	Av Time	0.32	2.06	10.47	73.14
	%Limit	0.00	0.00	0.00	9.64
7	%Opt	83.85	85.16	75.26	69.27
	Av Time	60.58	99.33	18.31	75.81
	%Limit	16.15	14.84	24.74	30.73
9	%Opt	60.16	65.36	48.44	41.41
	Av Time	124.30	89.95	51.38	65.79
	%Limit	39.84	34.64	38.54	58.33
11	%Opt	35.68	34.11	28.91	26.56
	Av Time	106.81	125.49	140.87	124.99
	%Limit	51.56	65.89	45.31	61.98
13	%Opt	14.06	20.31	8.85	16.93
	Av Time	254.17	146.95	230.03	209.46
	%Limit	61.98	73.44	63.54	61.46
15	%Opt	1.82	12.24	1.56	5.21
	Av Time	492.76	176.77	246.60	261.60
	%Limit	71.61	72.40	67.45	69.79

- Decision trees are much more informative



Decision trees

- CHAID needs large data samples and many replicates in order to be usable
- It looses power when there are many categories and results are difficult to analyze
- Not a common option in software. SPSS Decision Tree
- Interesting alternative for rank valued results in algorithm comparison

Decision trees

- After analyzing the tree many conclusions on the performance of the model can be obtained
  - This allowed us to detect weak spots that required further modeling
  - We gained a deep understanding of the model and how it could be improved
  - All the conclusions drawn are supported by a sound statistical procedure

Conclusions

- Even today we find inadequate analysis and testing of algorithms
- Parametric statistics pose an interesting and powerful alternative to non-parametric methods
- The DOE procedure and the posterior analysis by means of ANOVA techniques represent a very powerful approach that can be used for comparing performance of different algorithms and to calibrate methods

Conclusions

- Of particular interest is the study of interactions
- Insight on why algorithms work and how different features are affected by the input
- CHAID and decision trees: powerful non-parametric alternative for categorical response variables
- Sound statistical experimentation is a MUST

An overview of basic and advanced statistic techniques for calibrating and comparing algorithms

Rubén Ruiz García

INSTITUTO TECNOLÓGICO DE INFORMÁTICA
APPLIED OPTIMIZATION SYSTEMS GROUP
DEPARTMENT OF APPLIED STATISTICS, OPERATIONS RESEARCH AND QUALITY
POLYTECHNIC UNIVERSTITY OF VALENCIA, SPAIN

UNIVERSIDAD POLITECNICA DE VALENCIA

EMAA WORKSHOP, ICELAND
2006

Grupo de Investigación Operativa

http://www.upv.es/gio