Computational Economics **9:** 241–255, 1996.

Optimal Experimental Design for Combinatorial Problems

SELDEN B. CRARY
EECS Department, University of Michigan, Ann Arbor, Michigan, USA

COSIMO SPERA*
Dept. of Quantitative Methods, University of Siena, P.zza S. Francesco

(Accepted 8 September 1995)

Abstract. We discuss two experimental designs and show how to use them to evaluate difficult empirical combinatorial problems. We restrict our analysis here to the knapsack problem but comment more generally on the use of computational testing to analyze the performances of algorithms.

Key words: Computational, experimental design, Empirical evaluation, NP-hard problems, Knapsack problems

1. Introduction

In the past two decades researchers in operations research (OR) and computer science (CS) have begun to analyze algorithms through computational tests [17], [8], [22], [14]. While in disciplines like physics, medicine and economics empirical works have similar importance to theoretical works, this has not been so in OR and CS. Many reasons may be found for this, but we list three of the most evident here.

- Computational testing does not rely on deductive mathematics, so many do not consider it to be work of high quality;
- *Good journals* rarely accept empirical papers, often because editors are unable to judge the work; and
- Computational testing is not considered to be a *science*, because it is felt to lack a formal framework.

The advantages of computational tests in OR and CS are, however, manifold. They do not require the equipment needed by other disciplines, such as physics and medicine, and the costs to carry out the computational tests are low. In many cases, all that is needed is a workstation and a "graduate student". Once the test cases are established and hardware requirements are known (number of floating point operations, memory dimension, and the like), the experiments can be easily duplicated in other laboratories. The Internet can be used to make an instances data-base stored in a particular location available world wide to the entire scientific

* Corresponding author. Part of this work was carried out while the author was visiting the IOE Department at University of Michigan and the CS department at Columbia University.

community. Researchers may also contribute by providing instances of a specific problem to update the data-base.

Frequently these instances play an important role in the implementation of algorithms for solving particular problems. But this can cause problems. Implementors often employ tricks to speed up the examination of given data sets, leading to a computational testing speed race lacking a *scientific* approach. To make things worse, some experimentalists only add instances to the provided data set that work well for their methods, thereby adding little truly new information. All should realize that often negative results are as good as positive. But negative results are frequently omitted from the literature.

The last observation we raise is that, while computational testing may be cheap in cost, it can be very time consuming. So we like to obtain the needed information from a minimum number of test cases. This is particularly true when evaluating problems known to be difficult to solve and whose run time is not possible to anticipate for a given instance.

This paper show how computational testing can be conducted by following a *scientific* approach. Section two introduces the knapsack problem and the solution methods we are evaluating. Section three describes the theory for the experimental design, focusing on two particular designs: D-optimal and I-optimal. Section four shows the design layouts obtained. Conclusions and future expansions complete the paper.

2. The knapsack problem

A simple introduction to the knapsack problem is as follows. Suppose that a thief breaks into a jewelry store, bringing along a knapsack. Once in the store, he wishes to solve the following decision problem: which items (jewels) should be taken to maximize the value of the contents of the knapsack? We assume that the thief has good estimates of the values of the n jewels in the store and that these items may be considered one dimensional. This is seen to be a simple $0-1$ decision problem whose mathematical formulation is

$$\sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{subject to } \sum_{i=1}^{n} w_i x_i \leq w_0 \tag{2}$$

$$x_i = 0, 1 \quad i = 1, \ldots, n \tag{3}$$

where p_i, w_i $i = 1, \ldots, n$ are, respectively, the profit and the dimension values for the items, w_0 is the dimension of the knapsack, and the x_i are the binary decision variables. (1) represents the function to maximize, and (2) and (3) are the constraints. We refer to this problem as the single $0-1$ knapsack problem (SKP),

which is a very simple problem to state, but not to solve. In fact, as a decision problem, it has been proved to be NP-complete [10]. There are two approaches used to obtain the optimal solution to the SKP: dynamic programming (DP) and branch & bound (B&B) algorithms. The first approach gives a pseudo polynomial time complexity, $O(nw_0)$. Difficult knapsack instances for DP are generated by Chavtal [4]. These instances are easy to describe: p_i is set equal to w_i (SKP becomes the subset sum problem), and its value is given by an integer in the interval [1, $10^{n/2}$]. The capacity value w_0 is set equal to $\sum_{i=1}^{n} p_i/2$. For n sufficient large, Chavtal proves that "the running time of every recursive algorithm is bounded from below by $2^{2/10}$ for an overwhelming majority of such problems".

It is generally believed that B&B implementations perform better than DP implementations. In [20] Martello and Toth present a large number of computational results obtained running randomly generated test cases for different B&B algorithms. The generated instances are said to be uncorrelated if w_i and p_i are uniform random numbers from the interval [1,1000], weakly correlated if w_i is as above and p_i is a uniform random number from $[w_i - 100, w_i + 100]$, and strongly correlated if w_i is as above and p_i is a uniform random number from $[w_i - 1, w_i + 1]$. All the algorithms are based on the following key principles:

- *1. Compute an Upper Bound for the solution.*
  An upper bound for the optimal solution is computed and updated after a forward move is taken. The performance of the algorithm depends directly on the accuracy of this value.
- *2. A Forward Move.*
  A forward move consists of inserting as many consecutive items selected from the sorted list in decreasing order $d_i = p_i/w_i$ into the current solution.
- *3. A Backtrack Move.* A backtrack move consists of removing the last item inserted from the current solution.

Among the algorithms presented by Martello and Toth in [20], we have selected the ones they have already coded so that programming skill does not influence the computational results. The three algorithms are indicated as MT1, MT2, NKP01. Their detailed descriptions are in [21], [19]. Here we sketch the main differences.

MT2 and NKP01 use a *reduction* procedure, [1], but different upper bound values for the optimal solution. These values are reported in [20]. It is important to stress that these values are not analytically comparable, see [20]. In the OR literature these bounds take the name U6 (for MT2) and U2 (for NKP01 and MT1). These bounds influence the performance of the algorithms because a branch of the search tree is pruned when the current solution gives a value better than the upper bound value associated with this branch. The reduction procedure seems to have a great effect for large-scale problems. It fixes a priori the variables that will not be included in the optimal solution, (i.e. those that are zero at the optimum) and those that are included (i.e. those with value one at the optimum). MT1 and MT2 use different upper bound values; however, MT1 does not use the reduction procedure. MT1 and NKP01 use the same upper bound value and differ from each other in

that NKP01 uses the reduction procedure. A previous computational study shows how to construct a factorial experiment to determine the significant factors for this problem [18]. To determine if these algorithms differ meaningfully, we need to identify the nuisance factors that are not of main concern but which affect the response. In this case, the response is the CPU time needed to obtain the optimal solution. A machine independent response would be the number of nodes visited in the search tree. This choice reduces the residual measurement error to zero. Nuisance factors and their sources of variation can be controlled through three possible methods:

- Fix the nuisance factor to a constant.
- Define the levels of a given factor and assign units of experiments to each of these levels so that sources of variation can be distributed over the entire experiment.
- Include the nuisance factors in the experiment.

The third method uses a blocking procedure to isolate the variation attributable to the nuisance factors. The procedure defines n blocks of p homogeneous experimental units, where p is the number of levels for the nuisance factors. A previous study reporting on computational experiments on integer linear programming by Lin and Rardin [17] describes several possible design layouts for this approach. In our study we consider four factors: algorithms, number of total items, capacity value, length of the interval $[a, b]$ from which the values p_i and w_i are extracted. The last three factors are the nuisance factors. Their levels are defined as follows:

- **Number of variables**: we consider four different values for n: 100, 200 to take into account small size problems, 500, 1000 for medium size.
- **Capacity**: we consider five different values for the capacity w_0. These are computed as follows: determine $w_s = \sum_{j=1}^{n} w_j/m$, for $m = 2, 4, 8, 16, 32$ and then set $w_0 =$ closest prime to w_s.
- **Interval length**: let a be a value in the set $A = \{500, 1000, 1250\}$, and b be a value in the set $B = \{750, 1000, 1500\}$, under the condition that $a < b$ there are 5 possible intervals: [500–750]; [500–1000]; [500–1500]; [1000–1500]; [1250–1500]. One interval has length 1000, two intervals have length 500 and the remaining two intervals have length 250. Intervals with equal length differ for the number of digits in the data.

The factorial experimental design layout is given in Table I.

The computational results for the factorial design are reported in [18], where it is also indicated how observations that run for more than five minutes of CPU time on a VAX/6610 are treated. These observations are called censored.

To determine the significant factors, we apply the ANOVA procedure. This analysis assumes the normality of the corresponding random variables. This is not our case. In fact, our three response variables (one for each algorithm) follow the Gumbel-Max distribution as it appears evident by looking at the graph in Figure 1, which shows its linearization for the algorithm MT1. From this graph

Table I. Design Layout for Random Blocked Seeds

Nuisance Factors / Algorithm	Level 1	...	P
		⋮	
	Level 1	...	Level k
Algorithm 1	Problem a, b, c	...	Problem d, e, f
Algorithm 2	Problem a, b, c	...	Problem d, e, f
Algorithm j	Problem a, b, c	...	Problem d, e, f

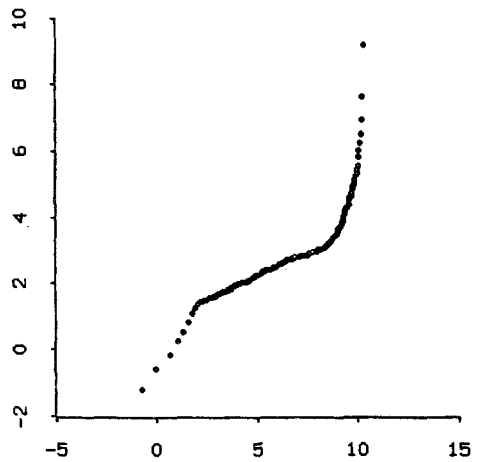

Figure 1. Gumbel-Max Cumulative Function.

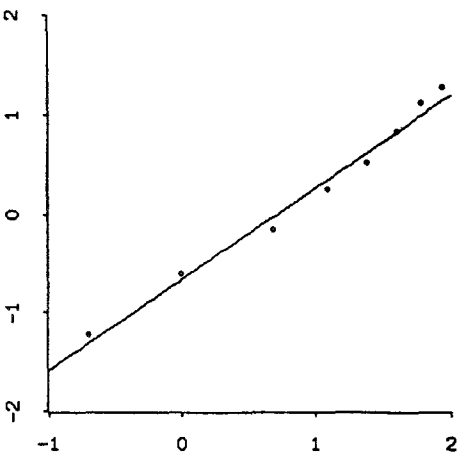

Figure 2. Gumbel-Max first piece.

we distinguish three different linear pieces highlighted in Figures 2, 3 and 4. The cumulative function of the Gumbel-Max distribution is

$$F(x) = \frac{1}{\delta} e^{\frac{-(x-\lambda)}{\delta}} e^{-e^{\frac{-(x-\lambda)}{\delta}}}. \tag{4}$$

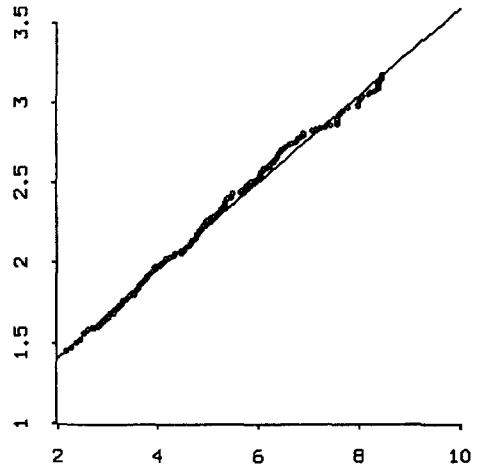

Figure 3. Gumbel-Max second piece.

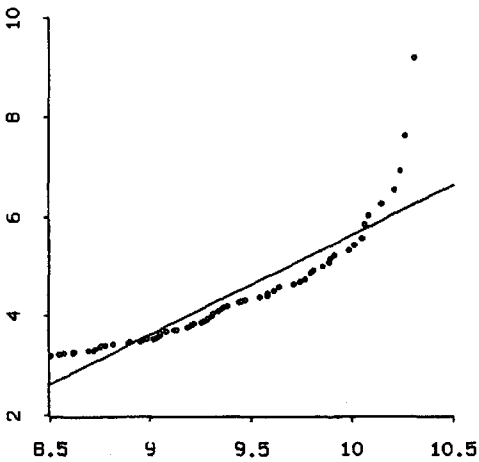

Figure 4. Gumbel-Max third piece.

The same analysis is valid for the other two response variables. For homogeneity the length of the three pieces is kept the same for all the three response variables. Since we are dealing with the transformed response variables $y = \log(CPU(MT1|MT2|NKP01))$ the three recognized segments are

1. $-0.69 \leq y \leq 1.38$
2. $1.39 \leq y \leq 9.99$
3. $10 \leq y \cdot$

Table II reports the values of the parameters δ and λ of the Gumbel-Max distribution for the three pieces, for each of the three algorithms.

Table II. δ and λ parameter values for the Gumbel-Max

Segment \ Algorithms	MT1		MT2		NKP01	
	δ	λ	δ	λ	δ	λ
1	1.58	0.45	4.16	−4.13	1.17	0.59
2	3.57	−3.05	2.81	−1.50	3.67	−3.55
3	0.35	7.99	2.76	−0.61	0.21	8.98

Table III. Proportion for the Considered Algorithm

Algorithm	Proportion		
	p_1	p_2	p_3
MT1	0.9	0.067	0.033
MT2	0.1	0.691	0.209
NKP01	0.86	0.09	0.05

Having determined the values of the parameters for the three segments and computed the general mixed model for the three response variables, it is possible to normalize them to apply the ANOVA procedure. The general mixed model is

$$F(x) = p_1 G_1 + p_2 G_2 + p_3 G_3, \tag{5}$$

where G_i indicates the Cumulative Function for piece i, and $\sum_{i=1}^{3} p_i = 1$. The proportions $p_i, i = 1, 2, 3$ are reported in the Table III for the considered algorithms.

The transformation function is

$$z = \sqrt{2\left(e^{-\frac{x-\lambda}{\delta}} + \left(\frac{x-\lambda}{\delta}\right) - \log\frac{1}{\delta}\sqrt{2\pi}\right)}, \tag{6}$$

where z indicates the normal variable with mean zero and unitary variance.

The results obtained from ANOVA are given in [18]. A derived model, which takes into account all the significant factors and their interaction, is

$$y = \beta_0 + \beta_1 N + \beta_2 Id + \beta_3 C + \beta_4 NC + \beta_5 CId + \beta_6 NId, \tag{7}$$

where N indicates the "number of variables" factor, C indicates the "capacity value" factor, and Id indicates the "length of the uniform interval distribution" factor. An alternative model that puts more "emphasis" on the factor N is

$$y = \beta_0 + \beta_1 N + \beta_2 Id + \beta_3 C + \beta_4 NId + \beta_5 NC + \beta_6 N^2. \tag{8}$$

The optimal designs derives from these two models are presented in section 4.

3. Optimal Design Theory

In this section, we review briefly the theory supporting the optimal design. In what follows, bold face denotes vectors and matrices, a superscript T denotes the matrix transpose operation, and a circumflex denotes expected value. The mathematical model in this case is linear in the coefficients, as in

$$Y(\mathbf{x};\beta) = \beta_1 f_1(\mathbf{x}) + \beta_2 f_2(\mathbf{x}) + \cdots + \beta_m f_m(\mathbf{x}). \tag{9}$$

This allows for a broad class of functions, including multivariate polynomials, such as $Y = \beta_1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_1 x_2$, or functions with non-linear terms, such as $Y = \beta_1 + \beta_2 \ln x_1 + \beta_3 x_1 \ln x_2$. The functions $f_1(\mathbf{x})$ are assumed to be linearly independent. A set of measurements represented by the column vector $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)^T$ is made at a set of specified values of the independent variables $\mathbf{x}$ with a set of random errors $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)^T$, the elements of which are assumed to have been drawn with replacement from a normal distribution with zero mean and constant variance σ^2,

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} f_1(\mathbf{x}_1) & f_2(\mathbf{x}_1) & \cdots & f_m(\mathbf{x}_1) \\ f_1(\mathbf{x}_2) & f_2(\mathbf{x}_2) & \cdots & f_m(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ f_1(\mathbf{x}_n) & f_2(\mathbf{x}_n) & \cdots & f_m(\mathbf{x}_n) \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}. \tag{10}$$

This can be written as $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, where $\mathbf{X}$ is called the design matrix. A key result from regression theory is that the best unbiased linear estimator of the coefficients is given by $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$. The estimated variance of $\hat{\beta}$ is given by $\sigma^2(\hat{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$, and the variance in the fit function is $\sigma^2(\hat{Y}(\mathbf{x})) = \sigma^2\mathbf{f}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{f}$, where $\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T$.

3.1. Optimality Criteria

Several optimality criteria are evident [27]. Three of the most useful are the following:

- *D-optimality*: the determinant of $(\mathbf{X}^T\mathbf{X})^{-1}$ provides a measure of the overall uncertainty of the parameter estimates, and a design that minimizes this determinant is called *D-optimal*. This criterion is equivalent to minimizing the volume of the confidence regions for finding the actual parameters [9].
- *G-optimality*: a design that minimizes the worst-case expected error in prediction is called *G-optimal* [13]. A theorem due to Kiefer and Wolfowitz establishes the equivalence of G- and D-optimal design in the limiting case that the number of experiments can take on non-integer values [15], the so-called approximate design.
- I-optimality: when the goal is to minimize the average variance in prediction over the entire range of $\mathbf{x}$, an appropriate objective function is [2]

$$\min_{w} \int_{\mathbf{x}\in X} E\{[\hat{Y}(\mathbf{x}) - Y(\mathbf{x})]^2\}\, \mathrm{d}\mu(\mathbf{x})$$

$$= \min_{w} \int_{\mathbf{x}\in\mathsf{X}} \mathbf{f}^T(\mathbf{x})(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{f}(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x})$$

$$= \min_{w} \text{ trace } \mathbf{B}(\mathbf{X}^T\mathbf{X})^{-1},$$

$$\text{where } \mathbf{B} = \int_{\mathbf{x}\in\mathsf{X}} \mathbf{f}(\mathbf{x})\mathbf{f}^T(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x})$$

is a matrix containing all the dependence on the model, and $\min_w$ indicates that the experimental design w is sought that minimizes the integral over the set of points $\mathbf{x} \in \mathsf{X}$. Weighting of different regions of the response is accomplished through the differential $\mathrm{d}\mu(\mathbf{x})$. This criterion is called **I-optimality** and has been detailed in the design-of-experiments literature [11].

3.2. Available Software

Finding optimal designs of experiments is a computationally intensive task, which has been well established [11]. Fortunately, recent advances in speed of computation, coupled with new algorithms such as simulated annealing [16] [25] are bringing the determination of optimal designs within the range of available capabilities, without undue expenses [11] [23]. Available software for finding optimal designs of experiments has been reviewed by Nachtsheim [24]. Software for finding D-optimal designs on finite grids has become widely available, and indeed included in popular statistical-software systems such as RS-Discover [28]. Meyer and Nachtsheim have discussed software for finding D-optimal designs on continuous spaces [23]. Welch's ACED software [31] finds designs approximating I- and G-optimal designs, in which potential design points are restricted to a fairly coarse grid. However, until very recently, virtually nothing existed for determining I- or G-optimal designs on continuous spaces – with the exception of a series of programs by Haines [11] that were used for two small classes of functions.

The I-OPT program, which is an extension of the work of Haines [11], finds I-, D- and A-optimal and near-optimal designs in cuboidal regions. (An A-optimal design minimizes the trace of $(\mathbf{X}^T\mathbf{X})^{-1}$). I-OPT was first presented in October 1989 at the First Great Lakes Computer Science Conference in Kalamazoo, Michigan, and was used in 1990 to optimize crystal-growth conditions in experiments by Sherwin *et al.* [29]. Originally, I-OPT used only simulated annealing, but it now includes a downhill search option that allows for accurate determination of objective-function minima. I-OPT has been available for researchers since its announcement in June 1991 [5] [27]. Since January 1992, I-OPT has been available in a workstation version that uses a hybrid of the simulated-annealing and downhill-search methods. It treats arbitrary multivariate polynomial models with arbitrary n. Extensions to I-OPT include capabilities for finding the following types of designs: compound I-, A- and D-optimal designs; weighted-integral I-optimal designs; designs with heteroschedastic error models; and Bayesian optimal designs, with a prior given for the information matrix $(\mathbf{X}^T\mathbf{X})$. This last capability allows for sequential designs

and for designs that take advantage of prior information on the distribution of β. Stopping criteria based on the probability that the global minimum has been attained or approached within a user-specified tolerance have also been added to the software. Interested parties may obtain the software by anonymous FTP to `freebie.engin.umich.edu` in a directory `pub/crary`, after consulting README file for downloading instructions.

Hardin and Sloane have written a C-language program named Gosset [12] that is capable of finding A-, D- and I-optimal and near-optimal designs on discrete or continuous spaces using multiple runs of a downhill-search technique initiated at a number of different starting designs. Gosset is able to find designs in which subsets of the independent variables can be constrained to spherical or cuboidal regions with linear inequality constraints.

3.3. I-OPT ALGORITHM

I-OPT is an interactive program that solicits the model function, number of experiments, and optimization method from the user. The optimization space has dimension d given by the product of the dimension of $\mathbf{x}$ and the number of experiments, and the optimization can be performed using one of the following user-selected methods: simulated annealing only, downhill search via a variant of Powell's method only, or simulated annealing followed by the variant of the Powell's method. The optimization algorithms have not been optimized.

Simulated annealing is accomplished using the variable-step-length generalized simulated annealing approach (VSLGSA) described by Sutter and Kalivas [30] that generalizes simulated annealing to continuous spaces. Additional detail on the implementation is available in [6]. The variant on Powell's method is similar in many respects to that found in standard references [26]. It works by making a set of line minimizations, constrained by the cuboidal boundaries of the space. I-OPT determines the constraints on the line minimizations. First, the limits of the line minimizations are determined by calculating the intersections of the line-minimizations line with the boundaries. Line minimizations use Brent's method [26], although we have extended this to allow for an active constraint. Initially line minimizations are made along the d Cartesian directions, and then a single line minimization is made along the new direction defined by the overall direction taken by the preceding set of d line minimizations. This set of $d + 1$ line minimizations defines an iteration. For the next iteration, the new direction replaces the previous one for which the decrease in I was the greatest. Thus, the search is not constrained to Cartesian directions. After $d + 1$ iterations, all directions are reset to the Cartesian directions, as suggested by Brent [3], in order to overcome the tendency in the Powell method for the set of directions to lose linear independence after several iterations. The resetting defines what we call a Brent cycle. We find that convergence occurs on the scale of one to a few Brent cycles.

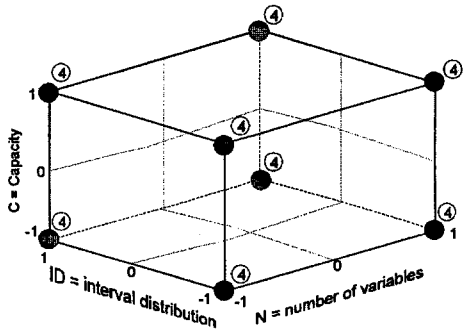

Figure 5. Model 1, D-optimal design, 32 points. Integrated Variance $= 2/27$, Probability that this is the global minimum $= 91\%$.

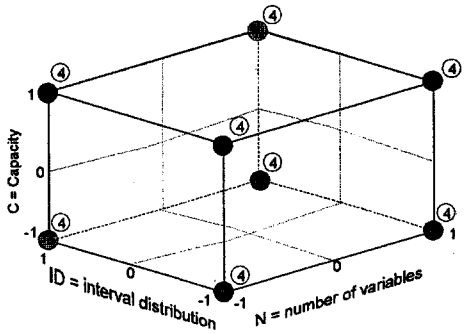

Figure 6. Model 1, I-optimal design, 32 points. Integrated Variance $= 2/27$, Probability that this is the global minimum $= 95\%$.

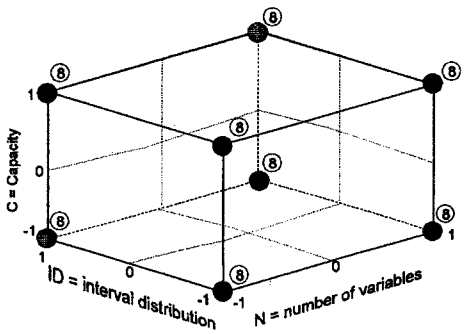

Figure 7. Model 1, D-optimal design, 64 points. Integrated Variance $= 1/27$, Probability that this is the global minimum $= 84\%$.

4. Design Layouts

Here we present the results obtained by using the procedures to compute the D-optimal and I-optimal designs for the model 7 indicated in section 2. We have decided to limit to 32 and 64 the number of replicated instances. To obtain some significant results we ran 7500 instances. The first model is a second-order function that includes the linear terms and all the interactions among the factors. As reported in Figures 5 to 8, D-optimal and I-optimal give the same design. These are fully

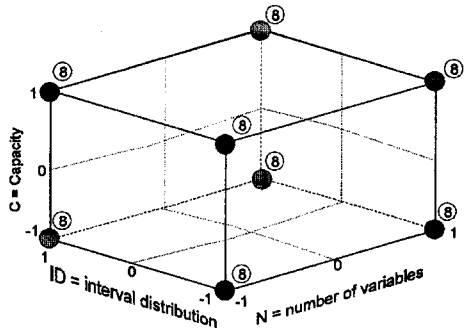

Figure 8. Model 1, I-optimal design, 64 points. Integrated Variance $= 1/27$, Probability that this is the global minimum $= 90\%$.

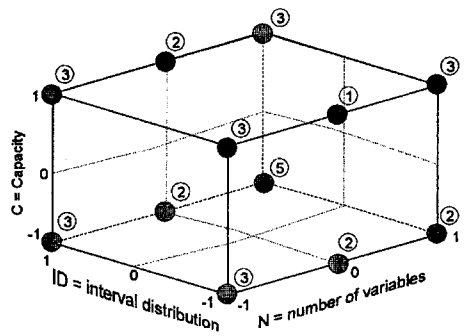

Figure 9. Model 2, D-optimal design, 32 points. Integrated Variance $= 0.1284$, Probability that this is the global minimum $= 49\%$.

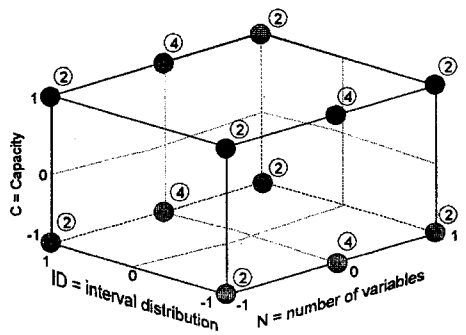

Figure 10. Model 2, I-optimal design, 32 points. Integrated Variance $= 73/720$, Probability that this is the global minimum $= 75\%$.

symmetric and determine instances at each of the eight vertices of the hypercube defined by the three nuisance factors.

More interesting results are obtained when we use model 8 of section 2. This model includes the linear terms, but the second-order terms are limited to those in which the factor N appears. The D- and I-optimal designs are shown in Figures 9 to 12. The first observation is that these designs do not coincide. The second is that

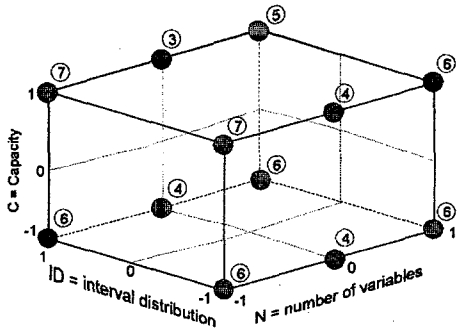

Figure 11. Model 2, D-optimal design, 64 points. Integrated Variance = 0.06168, Probability that this is the global minimum = 45%.

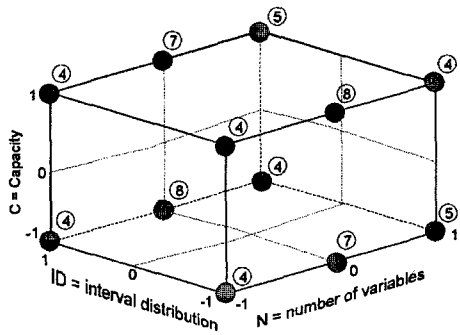

Figure 12. Model 2, I-optimal design, 64 points. Integrated Variance = 0.05053, Probability that this is the global minimum = 35%.

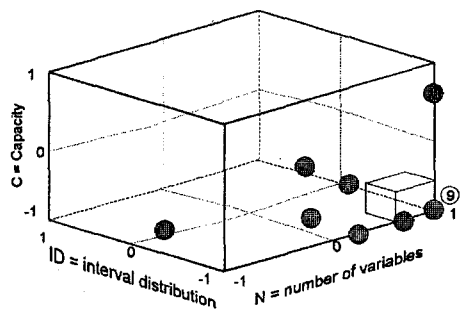

Figure 13. Model 2, I-optimal design, 16 points, integration over small cube only. Integrated Variance = 0.07977, Efficiency w.r.t. I-optimal design w. Integration over large cube = 5.75.

is that the D-optimal designs are not symmetric, while the I-optimal designs are symmetric about planes of the cube.

The last design we present refers to model 8, and it is obtained by putting more weight in a specified region of the design space. This is helpful in our case because we might a priori know which characteristics have the instances difficult to solve. In Figure 13 the I-optimal design with 16 points is derived assigning more weight to the region around the lower right vertex.

cut-off (%)	CPU sec.
5	0.91
10	3588.38
15	0.08
20	0.05
25	0.05
30	0.17
35	1.07
40	15.43
45	3.62
50	7.9
55	17.58
60	13.1
65	4.29
70	0.81
75	0.06
80	0.06
85	8.87
90	2925.83
95	1.39

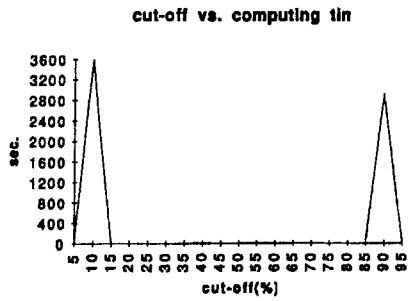

Figure 14. Time according to the cut-off.

5. Summary

In this paper we have shown how to construct optimal experimental designs to generate input instances to measure the empirical performance of algorithms. We have also commented on how computational testing may be conducted following a rigorous scientific methodology. We have left out some considerations and comments on the importance of computational testing to derive new theoretical results.

In this conclusions we briefly mention one case. From the analysis of the computational results we observed that on the same instance the CPU time varied according to the value of the capacity. This value takes the name of cut-off. Figure 14 shows the CPU times in seconds when the cut-off is positioned around a given percentage of included elements in the "greedy solution", i.e., the solution obtained by running a greedy algorithm that gives an upper bound value for the optimal solution. A further analysis has brought us to discover that for instances taking high CPU time, the elements around the cut-off have the value of the ratio p_i/w_i almost identically. "Almost identically" means that their values differ by a quantity ϵ, where ϵ is a small positive number. The more the interval $[p_i/w_i \pm \epsilon]$ around the cut-off is dense, the more CPU time the algorithm requires to obtain the optimal solution. This allows us in [18] to conjecture the time complexity of B&B methods as function of the density around the cut-off.

References

1. Balas, E. and E. Zemel. 1981. "An Algorithm for Large Zero-One Knapsack Problems", *Operation Research.* **28**, 1130–1154.
2. Box, G.E.P. and N.R. Draper. 1959. *J. Amer. Statis. Assoc.*, **54**, 622–654.
3. Brent, R.P. 1973. *Algorithms for Minimization Without Derivatives*, Englewood Cliff, NJ. Prentice-Hall.
4. Chavtal V. 1990. "Hard Knapsack Problem", *Operation Research*, **28** (6).
5. Crary, S.B. 1991. *Proceedings 1991 Int'l. Conference on Solid State Sensors and Actuators*, S. Francisco, CA. June 23–27, pp. 404–407.

6. Crary, S.B., L. Hoo and M. Tennenhouse. 1992. "I-Optimality Algorithm and Implementation", published in *Computational Statistics*, Proceedings of the 10th Symposium on Computational Statistics (COMPSTAT), Neuchâtel, Switzerland, August 24–27, 1992, v.2, pp. 209–214.
7. Crary, S.B. and Y. Jeong. 1995. *Proceedings 1995 Int'l. Conference on Solid State Sensors and Actuators*, Stockholm, Sweden, June 25–29, Vol. 2, pp. 48–51.
8. Crowder, H.P., R.S. Dembo and J.M. Mulvey. 1978. "Reporting Computational Experiments in Mathematical Programming", *Mathematical Programming*, **15**, 315–329.
9. Fedorov, V.V. 1972. *Theory of Optimal Experiments*. Academic: New York.
10. Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability*. W.H. Freeman and Company New York.
11. Haines, L.M. 1987. *Technometrics*, **29**, 439–447.
12. Hardin, R.H. and N.J.A. Sloane. 1993. A New Approach to the Construction of Optimal Designs", *Journal of Statistical Planning and Inference*, **37**, 339–369.
13. Hedayat, A. 1980. *Proceedings of the International Symposium on Statistics and Related Topics*, M. Csörgő *et al.* (eds.), Ottawa, Ont., 39–56.
14. Hooker, J. 1994. "Needed: an Empirical Science of Algorithms", *Operation Research.*
15. Kiefer, J and J. Wolfowitz. 1960. *Canad. J. of Math.*, **12**, 363.
16. Kirkpatrick, S., C.D. Gelatt, Jr., and M.P. Vecchi. 1983. *Science*, **220**, 671–680.
17. Lin, B.W., R.L. Rardin. 1978. "Statistical Comparison of Integer Programming Algorithms", *Management Science*, 1978.
18. Maddaloni, A., C.K. Murty, L. Pagliai and C. Spera. 1995. "Empirical Analysis of Algorithms for Combinatorial Problems: The Knapsack Case", Research Report, 1995, submitted for publication.
19. Martello, S. and P. Toth. 1988. "A New Algorithm for 0–1 Knapsack Problem", *Management Science*, **34**.
20. Martello, S. and P. Toth. 1991. *Knapsack Problems*. John Wiley & Sons: New York.
21. Martello, S. and P. Toth. 1982. "Algorithm for the Solution of the 0–1 Knapsack problem", *Computing*, **28**, 269–287, 1982.
22. McGeoch, C. 1994. "Graphical Methods for Assessing Functional Relationship", presented at 15th International Symposium on Mathematical Programming, Ann Arbor, MI, USA, 15–19 August 1994.
23. Meyer, R.K. and C.J. Nachtsheim. 1988. *Amer. J. of Mathemat. and Management Sciences*, **8**, 329–359.
24. Nachtsheim, C.J. 1987. *J. of Quality Technology*, **19**, 132–160.
25. Otten, R.H.J.M. and L.P.P.P. van Ginneken. 1989. *The Annealing Algorithm*. Kluwer Academic: Boston.
26. Press, W.H., B.P. Flannerty, S.A. Teukolsky and W.T. Vettering. 1989. *Numerical Recipes*. Cambridge University Press: New York.
27. Pukelsheim, F. 1993. *Optimal Design of Experiments*. John Wiley & Sons: New York.
28. RS/Discover, BBN Software Products Corp., 10 Fawcett St., Cambridge, MA. Version 2.0, 1989.
29. Sherwin, M.B., G.O. Munns, M.E., Elta, E.G. Woelk, S.B. Crary, F.L. Terry, and G.I. Haddad. 1991. *J. Crystal Growth*, **111**, 594–598.
30. Sutter, J.M. and J.H. Kalivas. 1991. *Analytical Chemistry*, **63**, 2382–2386.
31. Welch, W.J. 1984. *Technometrics*, **26**, 217–224; ACED, version 1.6, 1987, available from W.J. Welch, Dept. of Statistics and Actuarial Science, Univ. of Waterloo, Ont., Canada N2L 3G1.