Some Variants of the Landing Phase of the Cube Method

Fabio Sciamannini[1], Isabella Lari[1], Pier Luigi Conti[1]

[1] Sapienza, Università di Roma, Dip. Scienze Statistiche

pierluigi.conti@uniroma1.it

isabella.lari@uniroma1.it

fabio_scia@hotmail.com

Abstract

The Cube Sampling method is the most used technique for selecting balanced samples. It is composed of two phases: the flight phase, which can be performed efficiently, and the landing phase, which may require the solution of a Linear Programming problem with an exponential number of variables.

In this paper some variants of the landing phase of the Cube method are proposed with the aim of reducing the dimension of the Linear Programming problem to be solved and then improving on the total execution time.

The algorithmic efficiency of the proposed techniques is thoroughly evaluated via an experimental study.

Keywords: Balanced sampling design; Cube method; Landing phase.

1 Introduction

The efficiency of sampling strategies, i.e. of a pairs (*Sampling design*, *estimators*) depends in a crucial way on the criteria used to construct the sampling design and to choose the estimator of the parameter of interest. In particular, the sampling design is constructed on the basis of the available prior information, that is frequently represent by auxiliary variables (usually correlated with the variable of interest) having known values for all population units. Among the plethora of sample designs, a special role is played by the balanced sampling, which is based on a simple, powerful idea: the estimates of the means of the auxiliary characters should coincide with the corresponding means for the whole population. Of course, the construction of the balanced design requires to solve formidable computational problems. The main algorithm to draw a sample according to balanced sampling design is the Cube Sampling Algorithm. It is composed by two phases: the *flight phase* and the *landing phase*. In the present paper we will be mainly concerned with the landing phase. In particular new techniques for the landing phase will be proposed, and their algorithmic efficiency will be thoroughly evaluated via an experimental study.

2 Definitions and Notation

Let us consider a finite population $U = \{u_1, ..., u_N\}$ of N identifiable units, such that each of them can be uniquely labelled by an integer 1, 2, ..., N. Without loss of generality, in the sequel each unit will be identified by the corresponding label.

Let Y be the variable of interest and let $X_1, ..., X_p$ be p auxiliary variables, whose values are known for all units of the population. Denote by x_{ik} the value that the variable X_k assumes for unit i, ($i = 1, ..., N$ and $k = 1, ..., p$). The population means of the variables $X_1, ..., X_p$ are equal to

$$\mu_{x_k} = \frac{1}{N} \sum_{i=1}^{N} x_{ik}, \;\; k = 1, ..., p.$$

The idea behind a sampling strategy based on a balanced sample design is to use the information known a priori on the auxiliary variables, for the selection of a balanced sample.

As already said, the purpose of a sampling strategy so structured is the selection of a sample in which the estimates of the means of auxiliary variables are equal to the actual average of the variables themselves. If in addition, with the aim of completing the sampling strategy, the Horvitz-Thompson estimator is used, a requirement that must be verified by a balanced design is to satisfy the following *balancing equations*

$$\frac{1}{N}\sum_{i\in\mathbf{s}}\frac{1}{\pi_i}x_{ik} = \mu_{x_k},\ k = 1, ..., p$$

for each sample $\mathbf{s}$ belonging to the sample space $\mathcal{S}$. The balancing equations are assumed linearly indipendent. They define an affine subspace Q in $\mathbb{R}^N$ of size $N - p$, which is called *balancing hyperplane*.

For each population unit, let

$$s_k = \begin{cases} 1, & \text{if unit } k \text{ is in the sample} \\ 0, & \text{otherwise} \end{cases}$$

and let $\mathbf{s}$ be the $N-dimensional$ vector of components s_1, ..., s_N. The sample size is then $s_1 + ... + s_N$. Clearly, $\mathbf{s}$ is a vertex of the hypercube $[0, 1]^N$.

A sample design is a probability distribution on the set of possible values of $\mathbf{s} = (s_1, ..., s_N)$, i.e. a probability distribution on the vertices of $[0, 1]^N$. The inclusion probability of unit k is $\pi_k = E[s_k]$, $k = 1, ..., N$.

The balanced sampling design is constructed in such a way to satisfy two basic requirements.

1. $E(\mathbf{s}) = \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s}p(\mathbf{s}) = \boldsymbol{\pi}$, where $\boldsymbol{\pi} = (\pi_1, ..., \pi_N)$ is the N-dimensional vector of the inclusion probabilities of the first order of each unit of the population.

2. The balancing equations below, are met, "as far as possible"

$$\frac{1}{N}\sum_{i\in\mathbf{s}}\frac{1}{\pi_i}x_{ik} = \mu_{x_k},\ k = 1, \ldots, N. \tag{1}$$

A first way to find a balanced sampling design, widely used in the literature, consists in defining a cost function $Cost(\mathbf{s})$ for all possible samples of $\mathcal{S}$, that provides a measure of how the sample is distant from the balancing plane. The selection of the cost function is an arbitrary decision that depends on who make the survey. The cost must be such that

- $Cost(\mathbf{s}) \geq 0$, for all $\mathbf{s} \in \mathcal{S}$,
- $Cost(\mathbf{s}) = 0$, if $\mathbf{s}$ is balanced.

In this paper we consider a simple cost function, already used by Deville and Tillé in [8], defined as

$$Cost(\mathbf{s}) = (\mathbf{s} - \pi)'\mathbf{A}'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}(\mathbf{s} - \pi)$$

where $\mathbf{A} = (\frac{x_1}{\pi_1}, ..., \frac{x_i}{\pi_i}, ..., \frac{x_N}{\pi_N})$.

The choice of a so structured function has a natural interpretation as an Euclidean distance in $\mathbb{R}^N$.

At this point, a sample can be then selected by solving the linear programming problem

$$\begin{aligned} &\min_{p(s)} \sum_{s \in \mathcal{S}} p(\mathbf{s}) Cost(\mathbf{s}) \\ &\text{satisfying the constraints} \\ &\sum_{\mathbf{s} \in \mathcal{S}} p(\mathbf{s}) = 1, \\ &\sum_{\mathbf{s} \in \mathcal{S}} \mathbf{s} p(\mathbf{s}) = \pi, \\ &p(\mathbf{s}) \geq 0, \mathbf{s} \in \mathcal{S}. \end{aligned} \tag{2}$$

3 The Cube Sampling Algorithm

In 2004 Deville and Tillé proposed a first implementation of the Cube Method, the main algorithm for selecting a sample with balanced design, composed of two phases: the *flight phase* and the *landing phase*.

The flight phase consists of a random walk that starts at the vector of the inclusion probabilities π and remains in the intersection of the cube and the affine subspace Q. In other words, using the balancing equations defined in the theoretical formulation of the problem (1) and the first order inclusion probabilities, the algorithm performs a random walk in the intersection between the balancing plane and the hypercube until it reaches a vertex of the subspace $K = [0,1]^N \cap Q$. The fast implementation of the flight phase presented in Chauvet and Tillé [4] requires a $O(Np^2)$ execution time.

If at the end of the flight phase, the obtained vector π^* of inclusion probabilities is a vertex of the hypercube, we obtain the desired sample. Otherwise π^* is not integral and it is easy to prove that the number q of fractional components is at most equal to the number of balancing variables p. In this case the algorithm would enter the landing phase. The aim of this phase is to solve the linear programming problem (2) in a reduced form, that is inherent only to the fractional components of vector π^*. The number of variables of this problem is equal to 2^q, that is the number of vertices of the q-dimensional unit cube and then, depending on value of q, finding an optimal solution could be impossible from a computational viewpoint.

With the aim of studying how to improve the efficiency of the landing phase, we have proposed some alternative methodologies for the selection of a balanced sample. Our specific goal is to try to change, without incurring in excessive loss of information, the criteria to be considered when one has to handle a non integral vector π^* obtained at the end of the flight phase.

Of course, we do not claim that our study is complete and we are aware that further depth work will be required in the case of broader populations and with a number of balancing variables greater than those used in our experimental study.

4 Some Variants of the Landing Phase of the Cube Method and Experimental Results

The aim of this section is to introduce the alternative procedures of the landing phase of the Cube Method considered in this paper and to describe the results obtained in the experimentations realized in order to prove their efficiency.

We propose two classes of procedures: the first class consists in rounding two or more units of the vector of fractional components obtained at the end of the flight phase, considering their numerical relevance; the second class consists in trying to reduce the number of vectors to be analyzed in the optimization step of the Cube sampling algorithm, generating at random a subset of the vertices of the q-dimensional hypercube and solving a Goal programming problem.

The experimental analysis was carried out through the use of the free software R, using the package "sampling", first implemented by Tillé and Matei in 2007 [17].

The dataset used for the experimentation is "$MU284$", which refers to a population of 284 Swedish municipalities (Särndal et al; 1992).

The dataset, in its original form, contains 284 units, each corresponding to a Swedish municipality, and 11 different variables of which we give a brief description:

- LABEL, which represents the identification number of the variable;
- P85, which represents the amount of the population in 1985 in thousandths;
- P75, which represents the amount of the population in 1975 in thousandths;
- RMT85, which represents the income related to municipal taxation in 1985, in millions of Swedish kronor (SEK);
- CS82, which shows the number of seats in the municipal council of the Conservative Party;

- SS82, which shows the number of seats in the municipal council of the Social Democratic Party;
- S82, which represents the total number of seats in the municipal council;
- ME84, which represents the number of municipal employees in 1984;
- REV84, which represents the real estate values, based on an assessment of 1984, in millions of Swedish kronor (SEK);
- REG, which represents the code associated with the corresponding region of the unit we are considering;
- CL, a Cluster indicator, in which with cluster we refer to a set of adjacent municipalities.

Let us consider as the reference population the totality of the Swedish Municipalities in 1985, taking as a variable of interest the incomes related to the 1985 municipal taxation expressed in million SEK and considering as balancing variables P75, CS82, SS82, ME84 and REV84.

As far as the notation used in the above sections is concerned, the following symbols have been used:

- X_j, populations total for each variable j;
- $\hat{X}^i_{j,HT}$, estimated value of the variable j of the population by the Horvitz-Thompson estimator, in sample ith.

Moreover, let us create the vector of the inclusion probabilities of the first order as

$$\pi_i = \frac{n \cdot P85}{\sum_{i=1}^{284} P85_i}.$$

The experimental results related to the proposed methodologies are obtained by replicating the selection algorithms 1000 times for three different sample sizes ($n = 20, n = 30, n = 50$).

At the end of each iteration, for each implemented method, the software provides an output describing the quality of balancing, which includes for each considered variable (variable of interest and balancing variables):

- the population totals;
- the estimated total by the Horvitz-Thompson estimator;
- the relative deviation in percentage defined by

$$\sigma_j = 100 \cdot \frac{\mid \hat{X^i}_{j,\,HT} - X_j \mid}{X_j}.$$

Once the entire process finished, that is after 1000 consecutive applications of all the proposed methods for each sample size, we analyze the experimental results with the aid of the following indices of goodness:

- the true value of the population total;
- the mean of estimated population, defined as

$$\sum_{i=1}^{1000} \hat{X^i}_{j,\,HT};$$

- the relative deviation between the mean of estimated population and the mean of the total population;
- the quality of the estimator, evaluated in terms of efficiency (variance), computed as

$$\frac{\sum_{i=1}^{1000} \left(\hat{X^i}_{j,\,HT} - E\left(\hat{X^i}_{j,\,HT} \right) \right)^2}{1000};$$

- the bias of the estimator, computed as

$$E\left(\hat{X^i}_{j,\,HT} \right) - X_j;$$

- the mean square error, computed as

$$\sum_{i=1}^{1000} \left(\hat{X^i}_{j,\,HT} - X_j \right)^2;$$

- the absolute value of the maximum distance between the estimates obtained through the Horvitz-Thompson estimator and the true value of the variables;

- the absolute value of the maximum variation of the inclusion probabilities;
- the number of vertices generated on average;
- the actual sample size mean;
- the number of the variables pushed to zero on average (only for the methods based on the thresholds);
- the execution time of the algorithm.

The results of out experimental study we are shown in Tables 1, 2 and 3.

All the experiments were performed on a PC equipped with an INTEL INSIDE Core i7 Dual-core, with 4 GB of RAM and the operating system Windows 8.

4.1 Landing Phase by pushing to 0 or 1 some Inclusion Probabilities

Let us now consider the output vector of the flight phase.

As known, there are at most $q \leq p$ non-integral components; let us take into account only these components and put them into a new vector π^*, which will have size q.

At this point, we can think to use the numerical relevance assumed by each element of the new vector to force some of the components to 0 or 1, respectively, depending on whether they are smaller or larger than a certain numerical threshold, to which we refer as α and β. We set these thresholds at $\alpha = 0.2$ and $\beta = 0.8$, respectively, and then we push those components that assume a value smaller than 0.2 to 0, and those that assume a value greater than 0.8 to 1. What we are doing is nothing more than considering the value of each component of the vector as the probability of each element to be equal to 1, and depending on the probabilistic information contained in any fractional unit, pushing, respecting the established thresholds, each value to the integer (0 or 1) closer to that probability. For those components that, at the end of this rounding process,

are proved to be not yet an integer, we will just adopt the same procedure used by Deville and Tillé. That is, we will consider again a new vector π^* whose components are the elements still different from 0 and 1 and we will compute the sum of these components. Then we will generate all binary vectors having a number of 1's less than or equal to this sum; these vectors will form our new sample space. Finally, we will determine the probabilities of each sample in the selected sample space by solving the resulting linear programming problem (2) reduced in size.

The results of the experimentation inherent to this first alternative, show an underestimation of the variable of interest associated to a good reduction of the execution time of the algorithm. This behavior can be attributed to the fact that in vector π^*, the number of components close to 0, i.e. smaller than the lower threshold, is much higher than the number of elements close to 1, i.e. greater than the upper threshold. Realizing this, we tried to intervene on the idea previously expressed, refining the method of selection of the rounding thresholds in order to try to reduce the bias between the true value of the population mean with regard to the variable of interest and the estimated value thereof. So we have tried to improve the implementation in three different ways.

Let us consider the vector π^* of the non-integral components obtained at the end of the flight phase. At this point, for a predetermined value $\alpha \in (0, 1)$, we can think to push to 0 all those components smaller than the first α-quantile, and push to 1 all those components greater than the last α-quantile. The first variant is based on this idea and, since in our experimental study $q \leq 5$, consists in searching among the components of π^*, the minimum and the maximum elements and push them to 0 and 1, respectively, so as to balance, at least in quantity, the elements forced to the extreme values. In this case, we obtained an overestimation of the mean of the population associated with the variable of interest without obtaining an actual improvement. The reason for such behavior can be attributed to the fact that the order of magnitude of the difference between 1 and the maximum component of the vector π^* is much bigger than the order of magnitude of the minimum component of the same vector.

We then performed a further refinement inherent to the method of selection of the elements to be pushed to the extremes, in order to offset the probabilities that they represent. The second variant consists in searching among the fractional components the biggest one. Once selected, we will compute the difference between 1 and this component and we will use this distance to search for the components to be set to 0, in such a way that their sum is as close as possible, in order of size, to that difference. The results provided by this method of selection are good. In fact, in addition to a decrease in the execution time of the algorithm compared to the one already implemented in R by Deville and Tillé, we obtained also a significant decrease in the bias of the estimator with respect to all other methods analyzed in this work.

The third variant implemented is simply the symmetrical process of the previous one. It consists in searching for the smallest element of the output vector of the flight phase only among the fractional components in it. This will represent exactly the deviation from 0. At this point we will use this deviation for the research of the components to be pushed to 1, in a similar way to the previous case. In this respect, the results are not as good as those obtained for the second variant. In fact we observe an overestimation of the population mean in the variable of interest, probably due to the fact that the order of magnitude of the probabilities close to 0, is on average much smaller than the order of magnitude of the probabilities close to 1.

4.2 Landing Phase by generating a subset of the Hypercube Vertex set

In this paper we have also implemented a method for generating a reduced set of vertices differently from those generated by a total enumeration method used by Deville and Tillé.

The alternative proposed is to limit the number of binary vectors to be analyzed in the optimization step taking into account the probabilistic value of the fractional components contained in π^*. The elements of the sample space

are randomly generated using the values of the components of π^* as probabilities of pushing them to 1.

Finally, our goal is to solve the linear programming problem (2) defined on the obtained sample space, in order to be able to determine a balanced sample.

However, unlike the generation via total enumeration, we are not sure it is possible to get exactly

$$E(\mathbf{s}) = \sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s}p(\mathbf{s}) = \pi,$$

and then to be able to get a feasible solution of problem (2).

For this reason we consider the following *Goal Programming problem.*

$$\begin{aligned}
&\min_{p(\mathbf{s}),\,\varepsilon^+,\,\varepsilon^-} \sum_{\mathbf{s}\in\mathcal{S}} p\,(\mathbf{s})\,Cost\,(\mathbf{s}) + M\left(\sum_{i=1}^{q}\varepsilon_i^+ + \sum_{i=1}^{q}\varepsilon_i^-\right)\\
&\qquad\qquad \text{subject to}\\
&\qquad\qquad \sum_{\mathbf{s}\in\mathcal{S}} p\,(\mathbf{s}) = 1,\\
&\sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s}p\,(\mathbf{s}) - \varepsilon^+ + \varepsilon^- = \pi,\\
&\varepsilon_i^+ \geq 0, \quad \forall\, i = 1,\ldots,q,\\
&\varepsilon_i^- \geq 0, \quad \forall\, i = 1,\ldots,q,\\
&\qquad p\,(\mathbf{s}) \geq 0, \quad \forall\, \mathbf{s}\in\mathcal{S},
\end{aligned} \tag{3}$$

where M is a large penalty on the violation of the constraints $\sum_{\mathbf{s}\in\mathcal{S}} \mathbf{s}p\,(\mathbf{s}) = \pi$, which will allow us to obtain an optimal solution to problem (2) if it is feasible, or, if it is not the case, a solution that is as close as possible to the objectives we are pursuing.

However, an algorithm so constructed, has not led us to any improvement, neither from the point of view of execution time nor from the reduction of discrepancy between the real value of the population total and the mean of the estimated population total. Ultimately, we sought to refine this implementation, trying to create the hypothetical vertices totally at random. Even in this second case we incurred in a overestimate of the variable of interest and in a considerable increase of the execution time of the algorithm, without reaching

any improvement.

5 Conclusions and future works

In conclusion, the methods we examined to produce optimal samples mainly refer to two classes of procedures.

The first is a sort of rounding of two or more units of the vector of fractional components, taking into account their numerical relevance. The second does not consist in a total enumeration of all possible vertex of the N-dimensional cube with the properties described above, but reduces the amount of vectors to be analyzed in the optimization problem, using an algorithm based on a random generation of the vertices of the hypercube and solving the Goal Programming problem (3).

From the obtained results it is clear that the first class of methods, both for the execution time and efficiency of the estimator, is better than the second one, for which the results do not seem satisfactory at all. In particular, we have also seen that an improvement is obtained, when we compared one of the variants of the first alternative presented implementation with the landing phase proposed by Deville and Tillé in the case of sample size $n = 30$, as shown in Table 2. The idea behind this improvement is that, if in addition to balancing the number of components pushed to 1 or 0, we can also consider the amount of rounding, we observe a significant decrease in the running time of the algorithm and a marked decrease of the generated samples to analyze. Also a considerable reduction of the bias of the estimates associated with the variable of interest is obtained.

It should be kept in mind that the experimentation presented in this paper was performed on a not particularly high population size and that the number of balancing variables is small. These characteristics have limited the possible refinement techniques to be proposed.

Table 1

n=20							
	Cube Method	*Cube Method with prefixed tresholds*	*Cube Method with the selection of Min & Max*	*Cube Method with selection of Max to push to 1 and of the element to push to 0 depending on |1-Max|*	*Cube Method with selection of Min to push to 0 and of the element to push to 1 depending on |Min|*	*Cube Metho via Random Vertices Generation*	*Cube Method via Totally Random Vertices Generation*
	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85
Population Total	69605	69605	69605	69605	69605	69605	69605
Mean of Estimated Population	69535,18	69236,91	70076,16	69653,75	70184,77	69776,37	70654,32
Relative Deviation	0,10031	0,528824	0,6769074	0,07003357	0,8329388	0,2461974	1,507534
Variance	6,880025e-14	1,00636e-20	1,062518e-20	4,838337e-20	1,029574e-20	4,76456e-23	6,004537e-20
Bias	-69,82107	-368,0881	471,1614	48,74687	579,767	171,3657	1049,319
MSE	4874,98	135488,90	221993	2376,26	336129,80	29366,20	1101070
Absolute Value of Maximum distance between estimates and true value of variables	5533,025	6043,881	6.895.948	5125,496	6730,174	12341,46	9565,751
Absolute value of Maximum variation of Inclusion Probabilities	0,02932498	0,03322653	0,02554515	0,03699245	0,03777347	0,03777347	0,07021142
Number of vertices generated on average	31,934	12,941	10,669	10,362	10,687	23,239	35,125
Actual Sample Size mean	19,405	19,311	19,543	19,419	19,568	19,509	19,805
Number of Variables pushed to Zero on average	----------	1,597	----------	----------	----------	----------	----------
Execution Time of the Algorithm	95,28 secondi	87,47 secondi	86,92 secondi	80,17 secondi	80,03 secondi	107,97 secondi	96,56 secondi

Table 2

n=30							
	Cube Method	*Cube Method with prefixed tresholds*	*Cube Method with the selection of Min & Max*	*Cube Method with selection of Max to push to 1 and of the element to push to 0 depending on |1-Max|*	*Cube Method with selection of Min to push to 0 and of the element to push to 1 depending on |Min|*	*Cube Metho via Random Vertices Generation*	*Cube Method via Totally Random Vertices Generation*
	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85
Population Total	69605	69605	69605	69605	69605	69605	69605
Mean of Estimated Population	69612,49	69364,16	69869,87	69600,3117	69973,36	69729,49	70253.87
Relative Deviation	0,01076	0,346012	0,3805294	0,00006735577	0,5292121	0,1788542	0,9322177
Variance	4,981083e-20	8,009226e-22	7,725152e-21	1,175311e-21	3,594469e-20	1,103862e-19	3,354335e-20
Bias	7,488753	-240,8422	264,8675	-4,688298	368,3581	124,4915	648,8702
MSE	56,08141	58004,95	70154,77	21,98	135687,70	15498,13	421032,50
Absolute Value of Maximum distance between estimates and true value of variables	4328,145	5067,991	4118,772	4327,82	4051,456	8542,263	8246,05
Absolute value of Maximum variation of Inclusion Probabilities	0,05006176	0,02871987	0,03917004	0,03893824	0,02930459	0,05881712	0,09881712
Number of vertices generated on average	31,682	13,637	11,207	10,682	11,144	23,208	35,174
Actual Sample Size mean	28,124	28,009	28,25	28,097	28,282	28,189	28,478
Number of Variables pushed to Zero on average	----------	1,565	----------	----------	----------	----------	----------
Execution Time of the Algorithm	97,67 secondi	87,11 secondi	86,50 secondi	80,65 secondi	81,82 secondi	106,40 secondi	96,76 secondi

Table 3

n=50							
	Cube Method	*Cube Method with prefixed tresholds*	*Cube Method with the selection of Min & Max*	*Cube Method with selection of Max to push to 1 and of the element to push to 0 depending on |1-Max|*	*Cube Method with selection of Min to push to 0 and of the element to push to 1 depending on |Min|*	*Cube Metho via Random Vertices Generation*	*Cube Method via Totally Random Vertices Generation*
	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85	RMT85
Population Total	69605	69605	69605	69605	69605	69605	69605
Mean of Estimated Population	69595,61	69434,82	69869,87	69577,57	69783,75	69773,23	70020,12
Relative Deviation	0,013484	0,244488	0,3805294	0,03940464	0,2568031	0,2416966	0,5963918
Variance	6,976926e-20	1,431486e-22	7,725152e-21	7,82794e-20	8,139987e-20	3,537252e-20	2,117582e-21
Bias	-9,385853	-170,1759	264,8675	-27,4276	178,7478	168,2329	415,1185
MSE	88,09424	28959,84	70154,77	752,27	31950,79	28302,32	172323,40
Absolute Value of Maximum distance between estimates and true value of variables	2581,928	3141,206	4118,772	2412,088	2688,556	3625,136	3911,189
Absolute value of Maximum variation of Inclusion Probabilities	0,03734249	0,03612639	0,03927317	0,04780429	0,04119978	0,07002854	0,08102854
Number of vertices generated on average	31,962	14,474	11,342	11,216	11,885	23,98	35,417
Actual Sample Size mean	45,161	45,063	45,274	45,136	45,317	45,332	45,546
Number of Variables pushed to Zero on average	----------	1,534	----------	----------	----------	----------	----------
Execution Time of the Algorithm	96,03 secondi	87,57 secondi	84,51 secondi	82,73 secondi	83,56 secondi	109,52 secondi	97,51 secondi

References

[1] Ardilly, P., Tillé, Y., Sampling Methods, Springer, New York. (2006)

[2] Boyd, S., Vandenberghe, L., Convex Optimization, Cambridge University Press. (1977)

[3] Charnes, A., Cooper, W. W., Goal programming and multiple objective optimization, Part I, European Journal of Operational Research, 1, 3954. (1977)

[4] Chauvet, G., Tillé, Y., A fast Algorithm of Balanced Sampling, Journal of computational Statistics. (2006)

[5] Cochran, W. G., Sampling techniques, Wiley, New York. (1977)

[6] Conti, P. L., Marella, D., Campionamento da popolazioni finite: il disegno campionario, Springer-Verlag Italia. (2012)

[7] Coxeter, H.S.M., Regular Polytopes, Methuen e Co Ltd, London. (1948)

[8] Deville, J. C., Tillé, Y., Efficient Balanced Sampling: the Cube Method, Biometrika, 91, 893-912. (2004)

[9] Deville, J. C., Tillé, Y., Selection a several unequal probability samples from the same population, Journal of Statistical Planning and Inference, 86, 89-101. (2000)

[10] Deville, J.C., and Tillé, Y., Variance approximation under balanced sampling, Journal of Statistical Planning and Inference, 128, 569-591. (2005)

[11] Eggleston, H.G., Convexity, Cambridge, England. (1958)

[12] Horvitz, D.G., Thompson, D.J., A generalisation of sampling without replacement from a finite universe, Journal of the American Statistical Association, 47, 663-685. (1952)

[13] Ignizio, J. P., Introduction to linear Goal programming, Beverly Hills, CA, Sage Publishing. (1985)

[14] Sarndal, C. E., Swensson, B. e Wretman, J., Model Assisted Survey Sampling, Springer-Verlag, New York. (1992)

[15] Tillé, Y., Balanced sampling by means of the cube method, Joint Statistical Meeting of the American Statistical Association. (2006)

[16] Tillé, Y., Ten years of balanced sampling with the cube method: An appraisal, Survey Methodology, 2, 215-226. (2011)

[17] Tillé, Y., Matei, A., The R Package Sampling, The Comprehensive R Archive Network, Manual of the Contributed Packages. http://cran.r-project.org/web/packages/sampling/sampling.pdf. (2007)

[18] Tillé, Y., Sampling Algorithms, New York, Springer. (2006)

[19] Valliant, R., Dorfman, A. H., Royall, R. M., Finite Population Sampling and Inference: A prediction Approach, Wiley, New York. (2000)