



CASC PROJECT

Computational Aspects of Statistical Confidentiality

December 2003

Benchmark report:
Performance of alternative algorithms for secondary cell-suppression
implemented in τ -ARGUS 2.2

Author: Sarah Giessing

Institute: Federal Statistical Office of Germany

Deliverable No:3-D5



Benchmark report: Performance of alternative algorithms for secondary cell-suppression implemented in τ -ARGUS 2.2

Deliverable 3-D5

Sarah GIESSING,
Federal Statistical Office of Germany
65180 Wiesbaden
E-mail: sarah.giessing@statistik-bund.de

1. Introduction

τ -ARGUS is a software package for tabular data protection, offering to protect tables by cell suppression. The process of cell suppression involves mainly two steps: The set-up of the table with the identification of sensitive cells that might reveal individual information, if they were published. τ -ARGUS offers to identify, and suppress those cells. In the second step, in order to prevent these so called “primary suppressions”, or “sensitive” cells, from exact disclosure, or from being closely estimable from the additive relationship between the cells of the table, additional cells (so called “secondary” or “complementary” suppressions) must be suppressed. This second step is called “secondary cell suppression”.

The problem of finding an optimum set of suppressions is known as the ‘secondary cell suppression problem’. It is computationally extremely hard to find exact, or close-to-optimum solutions for the secondary cell suppression problem for large hierarchical tables. τ -ARGUS offers a variety of algorithms to find a valid suppression pattern even for sets of large hierarchical tables linked by linear interrelations. It is up to the user to trade-off quality vs. quantity, that is to decide how much resources (computation time, costs for extra software etc.) he wants to spend in order to improve the quality of the output tables with respect to information loss. The package offers a choice between different approaches:

OPTIMAL Fischetti/Salazar methodology aims at the optimal solution of the CSP [4]. A feasible solution is offered at an early stage of processing, which is then optimised successively. It is up to the user to stop execution before the optimal solution has been found, and accept the solution reached so far. The user can also choose the objective of optimisation, i.e. choose between different measures of information loss. Note that the method relies on high performance, commercial OR solvers. It should also be mentioned here that some problems have not yet been fully solved in the current version of τ -ARGUS: It may happen that the algorithm considers a single respondent cell to be properly protected even though there is only one other suppression in the same row/column/... of the table and this suppression is another single respondent cell (see also method SINGLETON below).

MODULAR The HiTaS method [3] subdivides hierarchical tables into sets of linked, unstructured tables. The CSP problem is solved for each subtable using Fischetti/Salazar methodology [4]. Backtracking of subtables avoids consistency problems when cells belonging to more than one subtable are selected as secondary suppressions.

NETWORK The concept of an algorithm based on network flow methodology has been outlined in [1]. Castro’s algorithm aims at a heuristic solution of the CSP for two-dimensional tables. Network flow heuristics are known to be highly efficient. It may thus turn out that the method is able to produce high quality solutions for large tables very quickly. τ -ARGUS offers an implementation applicable to 2-dimensional tables with hierarchical substructure in one dimension. A license for a commercial OR

solver will not be required to run the algorithm. In the current implementation, the protection of single respondent cells (see above) has not yet been solved.

HYPERCUBE The hypercube algorithm *GHMITER* developed by R.D. Repsilber (see [5,8]) is a fast alternative to the above three OR based methods. This heuristic is able to provide a feasible solution even for extremely large, complex tables without consuming much computer resources. The user, however, has to put up with a certain tendency for over-suppression.

SINGLETON Special application of *GHMITER*, addressing only the protection of single respondent cells. The method is meant to be used as preprocessing for the *OPTIMAL* and *NETWORK* methods, for which a solution for the problem with single respondent cells mentioned above has not yet been implemented.

With respect to *GHMITER*, and the modular method, both involving backtracking of subtables, it should be noted that such methods are not ‘*global*’. This causes a certain disclosure risk (see [2] for problems related to non-global methods for secondary cell suppression).

This document provides a report on the results of tests carried out to compare the performance of these alternative methods with respect to certain quality criteria. Section 2 presents the results of the tests. As a conclusion from the test results, section 3 will provide some guidelines for users which of the alternative methods to apply in a given situation.

2. Settings of the tests

For the testing, we used a variety of hierarchical tables generated from the synthetic micro-data set supplied as CASC deliverable 3-D3. In particular, we generated 2- and 3-dimensional tables, where one of the dimensions had a hierarchical structure. Manipulation of the depth of this hierarchy resulted in basically 6 different tables, with a total number of cells varying between 460 and 150 tsd. cells.

Except for the network flow method which was not yet available for application to hierarchical tables, all the algorithms listed above have been applied to those six tables. Using CPLEX 7.5 as OR-solver for methods ‘*Optimal*’ and ‘*Modular*’, runs were carried out on a Windows NT PC, Intel Pentium III processor, 261 MB Ram. For the largest table, runs with Linear Programming based methods could not be completed.

Table 1. Structure of test tables

Table	Hierarchical levels	Number of Cells		
		Total	Zero	Primary Suppressions
2-dimensional tables				
1	3	460	71	18
2	4	1050	168	60
3	6	8230	2061	989
3-dimensional tables				
4	3	8280	2740	848
5	4	18900	6937	2198
6	6	148140	78008	19029

3. Test results

In the following, we present results with respect to CPU usage (table 2), and information loss due to suppression.

Table 2. CPU times (h:mm)

Table	Hier. Levels	No Cells	Hyp	Mod	Opt	Si/Opt
2-dimensional tables						
1	3	460	< 0:01	< 0:01	0:02	0:00
2	4	1050	< 0:01	< 0:01	0:03	0:01
3	6	8230	< 0:01	< 0:01	0:10	1:02 (1:31)
3-dimensional tables						

4	3	8280	0:01	1:12	1:33	3:00
5	4	18900	0:01	3:11	12:10	15:00
6	6	148140	0:05	-	-	-

While for the 2-dimensional tables processing times were short enough to be of no concern, for the 3-dimensional tables, application of LP-based methods takes considerably more time than a run of the hypercube method. With increasing depth of the hierarchical structure, the effect of the modular implementation regarding reduction of execution time grows: for the 4-levels table 5, execution time is reduced by 3 hours (from 12:10 to 3:11), while for the 3-levels table 4 the reduction is only 21 minutes (from 1:33 to 1:12).

As mentioned above, with method ‘Optimal’ it is up to the user to stop execution before the optimal solution has been found, and accept the solution reached so far. It should be noted that we actually made use of this option. Thus, not all the suppression patterns generated by this method can be considered truly ‘optimal’.

3.2.1 Information Loss

In the following, performances of the algorithms on the test tables are compared with respect to number and added values of the secondary suppressions. Concerning the LP-based methods (Mod, Opt, Si/Opt), results presented in table 3 below were obtained when using the response variable as cost function.

Information Loss due to Secondary Suppression

Table 3.

Table	Hier. Levels	No Cells	No Suppressions (%)				Added Value of Suppressions (%)			
			Hyp	Mod	Opt	Si/Opt	Hyp	Mod	Opt	Si/Opt
2-dimensional tables										
1	3	460	6.96	4.35	4.78	7.61	0.18	0.05	0.03	0.05
2	4	1050	10.95	8.29	7.43	11.52	0.98	0.62	0.58	0.71
3	6	8230	14.92	11.48	15.36	17.97	6.78	1.51	1.64	2.06
3-dimensional tables										
4	3	8280	14.63	10.72	14.96	16.44	6.92	1.41	0.63	1.58
5	4	18900	17.31	15.41	19.19	20.00	12.57	3.55	2.32	4.48
6	6	148140	15.99	-	-	-	23.16	-	-	-

With respect to the number of secondary suppressions, method ‘Modular’ performed best on all tables except for table 2, where method ‘Optimal’ suppressed 9 cells less. Except for the 2 smallest tables, ‘Optimal’ with cell value as cost function performs even worse than the hypercube method.

With respect to the added value of the secondary suppressions, method ‘Optimal’ gave better results – with exception of table 3, where ‘Modular’ suppressed 0.92 % of the added value of the ‘Optimal’ suppression pattern. This might be explained by firstly considering that we stopped execution of ‘Optimal’ before the optimal solution was found, and secondly, results of an audit (see [7]) prove that for this instance the suppression pattern computed by the modular method does not satisfy the protection requirements (see remarks in section 1 on non-globality of the method), and thus was to be rejected by the ‘Optimal’ method.

Because the disclosure risk problem with single respondent cells also mentioned in section 1 has not yet been solved for method ‘Optimal’¹, it should be used in practice only in combination with ‘Singleton’ pre-processing. With respect to number of suppressions, the combination ‘Singleton/Optimal’ seems to perform poorly. Looking at the added value of the suppressions, however, the picture changes a little: while methods ‘Modular’ and ‘Optimal’ outperform the combined method here as well, we find that the suppressions selected are much smaller than those obtained by ‘Hypercube’.

Overall, the differences observed in performance are larger for the criterion ‘added value of suppressions’ as for ‘number of suppressions’. However, in our experience ‘number of suppressions’ is a quality criterion

¹ Out of lack of suitable software we did not systematically check the results of applications of method ‘Optimal’ for all instances. However, even for the smallest of the test tables (table 1) it was easy to find a case where respondents to two of the single respondent cells by an analysis of the table protected by ‘Optimal’ would have been able to disclose each others contribution exactly.

that matters a lot to statisticians, especially the number of suppressions on the higher levels of a table. Table 4 below presents results for 2 selected combined levels I, and II of the tables. Level I rows summarize results for the top level ('total') cells of the non-hierarchical variable(s), level II rows refer to those cells on the two top levels of the hierarchical variable which are 'inner' cells with respect to the non-hierarchical variable for the 2-dimensional tables, and with respect to one (and only one) of the non-hierarchical variables for the 3-dimensional tables. Suppressions on these high levels of a table are usually considered highly undesirable. Rows with zero entries for all 4 methods were dropped from table 4.

Table 4. Number of secondary suppressions for selected combined levels (in %)

Table	Combined Level	Hypercube	Modular	Optimal	Singleton/Optimal
2-dimensional tables					
1, 2	II	7.41	4.44	4.44	7.41
3	I	5.83	2.19	2.67	2.92
	II	12.59	6.67	6.67	17.04
3-dimensional tables					
4	II	12.05	5.13	9.49	11.03
5	I	14.29	0.00	0.00	3.81
	II	15.64	8.46	9.74	8.72

Table 4 again proves the superior performance of the optimization methods over the hypercube method. It also exhibits that, although overall the combined Singleton/Optimal method tends to suppress more cells than the hypercube method, on the top levels of the tables it behaves better.

In our experience, both criteria, number, as well as size of suppressed cells matter to statisticians. Obviously, in comparison to 'Optimal', results from the method 'Modular' tend to offer a better compromise between these two criteria. Presuming that another cost function for 'Optimal' might lead to a better compromise, we tested three alternative cost functions: number of suppressions (No), $\sqrt{\text{cell value} + \text{const.}}$ (c1), and $\sqrt{\text{cell value}}$ (c2). Results are available, however, only for table 3. For table 3, we also obtained results from two different applications of the Singleton/Optimal method (cost function: cell value), one stopped after 1 hour, the other one after 1 ½ hours. Table 5 presents the results.

Table 5. Performance with alternative cost functions, and increased execution times (table 3 data)

Combined Level	Hyp		Opt				Si/Opt	
	Val	Mod	Cost Function				CPU (min)	
			Val	c1	c2	No	62	91
No Suppressions (%)								
Overall	14.92	11.48	15.36	12.48	16.35	12.64	17.97	16.23
I	5.83	2.19	2.67	2.79	2.07	5.59	2.92	2.31
II	12.59	6.67	6.67	15.28	11.11	11.11	17.04	14.81
Added Value of Suppressions (%)								
Overall	6.78	1.51	1.64	4.75	1.74	6.05	2.06	1.86
I	2.35	0.09	0.06	0.20	0.06	0.54	0.11	0.11
II	0.04	0.01	-	1.55	0.01	1.32	0.03	0.02

In this experiment, using 'number of suppressions' as cost function resulted in a behavior similar to that of the hypercube method, with many suppressions on the top level (I). Cost functions c1, and c2 lead to less suppressions on this level. Cost function c2, however, results in poor performance regarding number of suppressions of the lower levels of the table. The modular methods clearly offers a better compromise between criteria 'number of suppressions' and 'added value of suppressions'.

Comparison of the performance of method 'Optimal' for different execution times presented in the last two columns of table indicates that spending more time for the optimization indeed improves results, although the effect of changing the cost function we observed in our experiment was stronger.

4. Summary and Conclusions

Results of this study prove that, in a situation where a user is interested in obtaining a suppression pattern for a single table with rather few, rather small secondary suppressions, preferably on the lower levels of the table, the best choice by far is to use the method 'Modular'. For medium sized 3 dimensional tables, long CPU times (compared to the hypercube method) are a nuisance, but quality of the results clearly justify the additional computational effort.

Results obtained by method 'Optimal' on the other hand were less convincing: firstly, the disclosure risk problem for single-contributor cells is not solved in the current implementation. Secondly, results depend strongly on the particular cost function employed. If the cost function does not fully reflect the users idea of a good suppression pattern, performance of method 'Optimal' will not be worth the additional computational effort (compared to method 'Modular'), which is quite considerable for 3-dimensional tables with elaborate hierarchical structure.

However, audit results (see [7]) prove that users of the modular, and the hypercube method face some disclosure risk: In our tables, we found up to 6 percent of primary suppressions where protection lacked. If a data provider cannot not put up with this, it could be an option to use method 'Optimal' for post-processing: After auditing the suppression pattern obtained from either of the heuristic methods, method 'Optimal' could be applied to the resulting table, addressing for protection only those primary suppressions with insufficient protection from the first run, while considering the bounds obtained from the audit as constraints on the cell values of the other (primary and secondary) suppressions. Because computation times for method 'Optimal' seem to depend strongly on the number of primary suppressions which would not be many for such a post-processing, computation times for the post-processing might be acceptable even for larger tables.

In a situation where multiple linked, or extremely large 3-, or more-dimensional tables have to be protected, with the current version of τ -ARGUS, the user is confined to use method 'Hypercube'. For suggestions how to improve the performance of this method for linked tables by specialized methods of processing the data see [6].

References

1. Castro, J. (2002), 'Network Flows Heuristics for Complementary Cell Suppression: An Empirical Evaluation and Extensions', In: *'Inference Control in Statistical Databases'* Domingo-Ferrer (Ed.), Springer (Lecture notes in computer science; Vol. 2316)
2. Cox, L. (2001), 'Disclosure Risk for Tabular Economic Data', In: *'Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies'* Doyle, Lane, Theeuwes, Zayatz (Eds), North-Holland
3. De Wolf, P.P. (2002), 'HiTaS: A Heuristic Approach to Cell Suppression in Hierarchical Tables', In: *'Inference Control in Statistical Databases'* Domingo-Ferrer (Ed.), Springer (Lecture notes in computer science; Vol. 2316)
4. Fischetti, M, Salazar Gonzales, J.J. (2000), 'Models and Algorithms for Optimizing Cell Suppression Problem in Tabular Data with Linear Constraints', in *Journal of the American Statistical Association*, Vol. 95, pp 916
5. Giessing, S., Repsilber, D. (2002), 'Tools and Strategies to Protect Multiple Tables with the GHQUAR Cell Suppression Engine', In: *'Inference Control in Statistical Databases'* Domingo-Ferrer (Ed.), Springer (Lecture notes in computer science; Vol. 2316)
6. Giessing, S. (2003), 'Co-ordination of Cell Suppressions: strategies for use of GHMITER', paper presented at the Joint ECE/Eurostat Worksession on Statistical Confidentiality in Luxembourg, 7.-10. April 2003
7. Giessing, S. (2004), 'Survey on methods of τ -ARGUS', to be presented at the PSD'2004 conference in Barcelona, June 2004
8. Repsilber, D. (2002), 'Sicherung persönlicher Angaben in Tabellendaten' - in *Statistische Analysen und Studien Nordrhein-Westfalen, Landesamt für Datenverarbeitung und Statistik NRW*, Ausgabe 1/2002 (in German)