Improving LP performance in Cell Suppression Process

Bei Wang[*]
U.S. Census Bureau[†]
Washington, DC 20233

Abstract
Since 1992, the Economic Census has used cell suppression to protect sensitive information. The disclosure process that suppresses sensitive cells uses a network flow model (MCF) to ensure minimal information loses. However, the answer is often not optimal because the model does not handle large complex table structures very well. Linear Programming (LP) is an alternative methodology that may overcome the limitations of MCF.

A typical LP sets constraints in a multi dimensional grid with one targeted entry (the primary cell) and finds a suppression pattern to protect that target. The cell suppression process completes after solving an LP for each of the primary cells. This is a simple sequential approach. The number of constraints is determined by the size and complexity of the tables being published. The performance depends on both the number of constraints and variables and the number of primary cells. There are two problems of simple sequential approach (1-LP). The first, even if the execution time for each target is fast enough, the time spent on the whole process can be unsupportable. The second, while each LP is optimal it is not optimal globally. This research addresses the first issue; several targets are formulated in one LP such that the computing time for one LP remains small while the whole processing time is reduced. We will explore this partial simultaneous LP (m-LP): how the number of targets in one LP reduces the overall processing time and find the best such number in the trade off with oversuppression. We will also determine how the order of targets to be processed enhances/compromises the objective, i.e., the number of suppressed cells and/or total suppressed value. We will make some comparisons with results from the existing 1-LP program. This research addresses the speed problem LP had and has quantified the amount of oversuppression for selected data.

1. Introduction to Cell Suppression

Census Bureau economic census uses Cell suppression to protect sensitive cells (Ps). Currently, its program is designed sequentially, protecting one P at each LP call, called LP or 1-LP to distinguish the notation used later in this article. A cell suppression process completes after solving an 1-LP for each of the primary cell. We consider two criteria to evaluate a disclosure avoidance methodology, and take as given that the methodology is well founded (no disclosure). One is how much is suppressed (number and value of cells). The other is the complexity of the algorithm. There is no one unique method that meets both. The alternative is to find one that balances the two. Integer Programming (IP) is the best to achieve the first goal because its objective is the best to fit the problem. However, it introduces binary variables, adds complexity and is a NP-hard problem that leads to exponential growth (in time) as the size of the problem increases. Therefore, it is impractical and violates the second criteria. Census Economic Census has been

[*]bei.wang@census.gov

[†]This report is released to inform interested parties of (ongoing) research and to encourage discussion (of work in progress). Any views expressed on (statistical, methodological, technical, or operational) issues are those of the author(s) and not necessarily those of the U.S. Census Bureau.

using a network model which solve the problem within a reasonable time, in $O(nm)$ [1], where n and m are the sizes of problem in the network. But it causes both oversuppression and undersuppression; its methodology is not well founded. Linear Programming (LP) seems a natural choice balancing between IP and network. It takes a similar sequential approach as network does, by protecting one P at a time. First, LP can also be solved in polynomial time $O(N^i M^j)$ where N, M are the dimensions of the problem, however $N >> n$ and $M >> m$, $i, j > 0$. Second, it looks at a global picture and taking account all related cells, while network only concentrates on local optimization. Therefore, it reduces undersuppression. While LP addresses some of the issues and has the same complexity as network has, it is a much larger problem than network. To solve the problem (protecting all sensitive cells) for all Ps, it takes as much as $\#P{*}O(N^i M^j)$, which can also be unsupportable for many of the larger Economic Census data sets. The simultaneous multiple Ps (m-LP) approach is a natural extension of 1-LP. It protects m Ps at a time and takes as little time as a fraction of m.

Section (2) explains m-LP approach, Section (2.4) Table (4) examines some results: how m, infeasiblity affects/improves performance.

2. Partial Simultaneous

A simultaneous LP (m-LP) is to protect several (m) Ps, say mPs where $m > 1$, in one LP formulation. It has the same complexity as one P LP (1-LP) because its constraints are still linear without adding additional variables and constraints. However, it reduces the number of times the solver is called, and processes m Ps at the same cost as it does one P. Therefore, it dramatically reduces the running time, usually to a factor of m. The only changes are on the bounds of the protection needed variables (Ps), ie: $x_+ = prot\_required$, and $x_- = 0$ instead of $x_+ \in [0, value]$, and $x_- \in [0, value]$, as described in section (2). There can be two side affects with this approach. One is oversuppression. The other is that it leads to conflicting constraint causing an infeasibility.

Both oversuppression and infeasibility could occur when there are several Ps in a constraint on the mPs list. For example, in a constraint $a = b + c$, where both cell b and c are Ps. In 1-LP, cell b maybe used to protect cell c, and verse versa. However, in m-LP, cell a has to be invoked to protect both b and c, because, by the given constraint, aflow = bprot+cprot. When this happens oversuppression occurs. Furthermore, if cell a is also a P then it is very unlikely that the protection required by cell a matches the sum of required protection from both cell b and cell c. In this case, the constrain: aprot = bprot + cprot, is a conflicted constraint, which causes an infeasiblity. Generally, a feasible flow that protects the Ps jointly also is a feasible flow that protects them separately, and the union of two feasible flows that protect the Ps separately may not be feasible to protect the Ps jointly (the capacity constraints may be violated). Thus the set of feasible flows for the joint problem is smaller than that of the separate problem - causing oversuppression. While oversuppression seems unavoidable, avoiding infeasiblity is not hard. One may switch to a one P process any time an infeasible occurs for a set of P. One also to select the m Ps so that they are mutually exclusive, in the sense that no two Ps are in the same relationship. It is probably even better when all m Ps are "far apart". One idea for implementing "far apart" involves finding shortest path between any two Ps for each dimension and finding m "least connected" Ps. Another idea is "depth" disscussed in Section (2.3). It is designed so that each P

may find its own protection with minimum interference from other Ps. It is much like protecting each P in its own neighborhood, but finding solution for m Ps at the same time. Therefore choice of p may reduce oversuppression. It is no guarantee that each of m-LP is feasible by the second approach (selection of mPs) alone. But combining it with the first approach (convert to a 1-LP), it is guaranteed that every P is protected by some LP solve in a cell suppression process.

2.1 Cell Suppression Process

A Cell Suppression Process, using m-LP model, processes all Ps sequentially m number of Ps at a time until all Ps are done. To improve the process, the program may start with $m = 1$ for the first few (10) Ps, reset to a speicific $m > 1$ after that, and again, reset m back to 1 near the end several (10) loops . This design, at the beginning, helps shake off these Ps that are already protected - skipped Ps (will be explained in Section (2.4)), and near the end, avoids most infeasibilities.

2.2 Setting Up a m-LP - constraints and cost objective function

We are solving a m-LP for m Ps. The goal is to

$$\begin{array}{lcr} \text{minimize} & \sum_{i,j,k} v^r_{ijk}(x^+_{ijk} + x^-_{ijk}) & \\ \text{subject to} & \sum_{j,k}(x^+_{ijk} - x^-_{ijk}) = x^+_{i11} - x^-_{i11} & \forall i \\ & \sum_{i,k}(x^+_{ijk} - x^-_{ijk}) = x^+_{1j1} - x^-_{1j1} & \forall j \\ & \sum_{i,j}(x^+_{ijk} - x^-_{ijk}) = x^+_{11k} - x^-_{11k} & \forall k \end{array} \tag{1}$$

where $x^{\pm}_{ijk}$ are flows through each cell, v_{ijk} generally are the cell value and r is the real exponents.

Additionally, for these selected m primary cells, we have

$$\begin{array}{l} x^+_{ijk} + x^-_{ijk} = \text{prot}_{ijk} \\ x^+_{ijk} = 0 \;\text{ or }\; x^-_{ijk} = 0 \end{array} \tag{2}$$

Notices that constraints (2) add $2m$ binary, and makes it a mixed integer programming which is a much harder(complex) problem than LP. To keep it a true linear programming, we simplify constraints (2) to

$$\begin{array}{l} x^+_{ijk} = \text{prot}_{ijk} \\ x^-_{ijk} = 0 \end{array} \tag{3}$$

which force flows in one direction for all m Ps. To solve a cell suppression problem, m-LP solvers are expected to be called approximatly $\#Total\text{P}s/m$ time. Notices that at $m = 1$, m-LP = 1-LP. Ideally, using m-LP will reduce computation time to a fraction of m of 1-LP. Although the simplified constraints (3) keep the complexity of m-LP the same as 1-LP, it evidently creates more oversuppression than constraints (2) because the former creates more needed protection. For example, in a relationship, where there are multiple Ps, constraints (3) ask for the amount of protection needed as $\sum_i \text{prot}_i$, while constraints (2) will negotiate among the Ps and may settle the needed protection with the amount diff$|prot(x_i) - prot(x_j)|$.

Observe that, for 1-LP, constraints (2) and (3) are equivalent within the problem. Therefore it makes sense to use (3) in the model to keep the problem linear.

2.3 Selection of m Ps

Ideally, we like to select m unrelated Ps, so each P is locally isolated. In reality, data cells are all somehow related. We arrange the data based on its relationship in a data structure, picking m Ps from different table "cateegory. We hope each P finds protection from its own neighborhood with minimum interference with other Ps in the mPs list, therefore avoiding infeasibility and reducing oversuppression. Usually, the row relationships follow a strictly hierarchy structure, we assign number to row by its position in hierarchy. We classify each cell by enumerating column relationships, and appending it to row depth. This is done similarly to any other dimension where it applies.

How many Ps should be processed in a m-LP is data dependent. Generally, the larger of m, the less time it takes to solve whole problem. But a larger m may not lead to best cells (counts or value) suppressed, or time because of the infeasibilities it encounters. The suggestion is that the largest m does not exceed the maximum number categories of data structure.

2.4 Results

2.4.1 Test Data: Table (1)

We use economic Census$'$ data to test m-LP. Table (1) shows basic statistics of three sets of data, such as the dimension of data structure, the number of records, the number of sensitive cells (Ps), performance time (for the whole process) and time spent per p. The three data sets are: a subset of Annual Survey of Manufacture for 2010 (ASM10), Disclosure Group 07 for service sector 71 (dg07 71), and Disclosure Group 10 for service sector 71 (dg10 71) which is a larger data set. Some data sets have dense sensitive cells (see density %) such as dg07 and dg10, and asm10 is the least dense set with 10% Ps. These are just a small selection of small to medium size economic census data. As the data size gets big, the performance time becomes increasingly important.

Performance time is determinded by the size of LP (number of cells and dimensions which translate to number of variables n and constraints L) and number of Ps to be processed. The number of cells, accordingly the number of variables, carries more weight than the number of constraints does. It may be of $\mathrm{O}(\mathrm{n}^{1.5}\mathrm{L})$. Our R&M team worked very hard to improve the performance by reducing number of variables (unduplicating)- preprocessing the input data to elliminate/substitute duplicates, and number of LP calls by "skipping P". We test each P to see whether it receives enough flow (>protect required) for each LP solution. Then Ps having enough flow are marked protected in the list of Ps to be processed; they are skipped in the process as it sequentially proceeds down the list. We also mark a P "protected" if the P is in an all-P relation, i.e. all cells in the relationship are Ps, for all relationships the Ps involved in. This last step is done before the 1st call of LP solver. We can see how number of Ps is reduced at the start in Table (1) on "Ps at start" column. Constraints are added only when cell data are presented and acquired in their respective input files. Those are our LP model based on as the columns, in Table (1), "#constraints, #variables, and #Ps at start", have shown.

2.4.2 Table (2) - (4)

The m-LP results, for the selected m, are listed on Table (2), (3),and (4) with suppression, running time, number of m-LP called (n_{lp}), infeasibility encountered (n_I),

Data Name	#Cells (n)	Dims: Rowx-ColxLev	Total$P$s (density%)	TotalConstraints(L) =(row+col+lev)	#Ps at Start	T_{1-lp}	#LP calls	t_{lp}
ASM10	21781	474x13x57	2334 (11)	12721 =5311+7081+329	2334	43:17m	1214	2.14
dg07 71	17467	61x472x3	11037 (63)	23045 =9514+5960+7571	9263	19:16m	1410	.82
dg10 71	62112	62x1858x3	41271 (66)	81555 =34797+19592+27166	34277	5hrs	3715	4.8

Table 1: Test Data Statistics (T_{1-lp}: Total Time, t_{lp}: Average time for each solve)

and oversuppression (%) compared with 1-LP which is also represented on graphs (1) - (3). Similar information is also provided for network model for comparison.

We choose m up to 10 mostly to see how performance altered. For the larger data set, dg10, we tested $m = 24$ to cut down its performance time. Generally, the larger the data set is, the bigger the m can be choosen.

By trial and error we created some gold standard cases where no infeasible occurres in the process, see the bottom part of Table (4), where mPs are picked by using some sort of classification scheme discussed early. It appears the running time reduced proportionally as m increases, as we expected. It also suggests that the number of infeasible take back performance time gained from m-LP shown from other cases, see the first ten cases of Table (4).

When Cell suppression runs on 1-LP solver, for example, it takes 5 hours for dg10 sector 71 data. In ideal situation, when each set of m Ps is selected such that no infeasible occurs, using m-LP solver takes a fraction of m time. Such are the cases at $m = 5, m = 10$, and $m = 24$, where infeasibility encountered is zero, as shown in Table (4), each taking 1:02 hour, 33:23 and 16:40 minutes respectively. However the number and value suppressed are compromised. asm10 suffered 8% and 4% increase, respectively, dg07 1% and 3%, and dg10 1% and 9%, at the worst case. In this aspect, network has more, about 21% and 47% for dg07, and 19% and 62% for dg10 [1]. However network does have the huge advantage with time and it takes 2min. for dg10. The best possible time m-LP has is 16min. from the existing testing. Most of asm10 tests don't have any infeasibles. I guess this is because its Ps density (11% Ps) helps isolate mPs naturally. The other two data sets have the density more than 60%.

2.4.3 Table (5)

A regression model is developed to predict run time, see Table (5), which is a function of infeasibles and LP calls. There is also a column showing the time per call from 1-LP process. The model also proves that infeasibles cost us and, without infeasibles, time is generally a portion of 1-LP. It is important to develope and improve the "classification" scheme that allows a true m-LP cell suppression process - a process with no infeasibles.

[1]production results may vary because of the weights and other parameters used in the production.

	ASM10						
m(Ps)	Total Cs	Total C Value	T_{m-lp}	n_I	n_{lp}	Changes(%)	
1p	2301	670274340	43:17	0	1214	Cells	Value
2p	2306	673580156	22:19	0	626	.22	.49
3p	2355	676518979	14:47	0	410	2.35	.93
4p	2473	678182821	11:01	0	204	7.48	1.18
5p	2284	676028884	9:20	0	258	-.74	.86
6p	2424	688385318	7:29	1	204	5.35	2.7
7p	2375	682393273	6:41	0	182	3.22	1.81
8p	2443	694674105	6:05	1	164	6.17	3.64
9p	2473	685847489	5:18	1	142	7.48	2.32
10p	2474	695176051	4:44	1	126	7.52	3.72

Table 2: How m & Infeasibles Affecting Performance

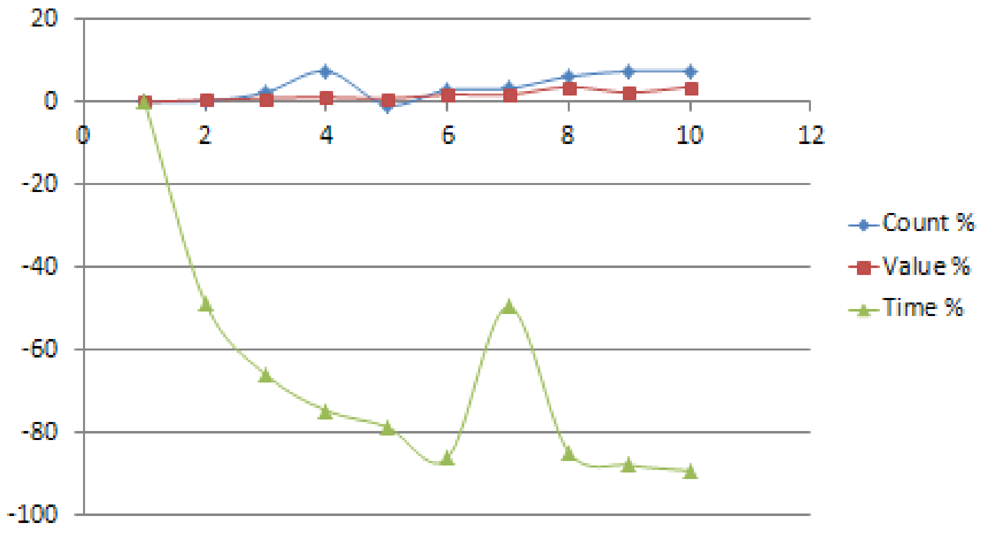

Figure 1: Suppression comparison with 1-LP for ASM10

	dg07 71						
m(Ps)	Total Cs	Total C Value	T_{m-lp}	n_I	n_{lp}	Changes(%)	
1p	3408	62017093	19:16	0	1410	Cells	Value
2p	3415	63727898	9:50	12	698	.21	2.76
3p	3410	64132289	6:56	14	486	.059	3.41
4p	3411	62599725	6:00	45	410	.088	.94
5p	3385	63160838	5:36	58	362	-.67	1.84
6p	3444	66027101	4:44	52	300	1.06	6.47
7p	3438	65581098	4:57	78	308	.88	5.75
8p	3441	65546601	5:19	117	316	.97	5.69
9p	3436	65865125	5:38	123	326	.82	6.2
10p	3434	64167178	4:50	105	292	.76	3.47
network	4124	91487060	secs			21	47.5

Table 3: How m & Infeasibles Affecting Performance

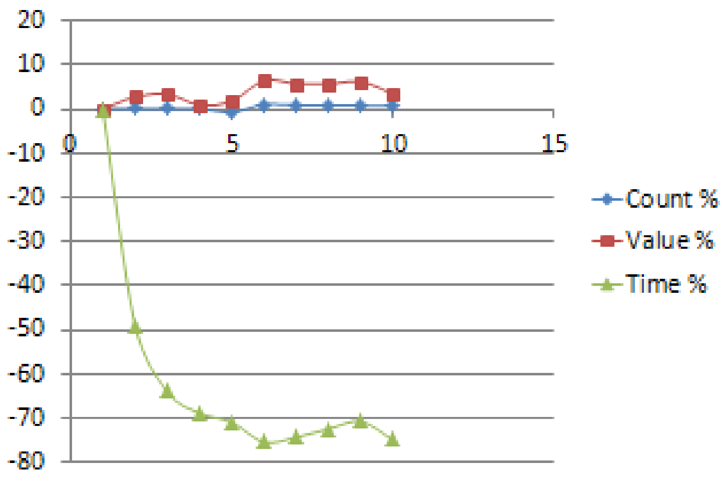

Figure 2: Suppression comparison with 1-LP for dg07

	dg10 71						
m(Ps)	#Total Cs	Total C value	T_{m-lp}	n_I	n_{lp}	changes(%)	
1p	11300	358326988	5hrs	0	3715	Cells	Value
2p	11335	383086282	2:17	44	1750	.31	6.9
3p	11402	383748045	1:38	91	1208	.90	7.09
4p	11457	384419717	1:25	151	986	1.39	7.28
5p	11376	383902919	1:14hrs	177	862	.36	7.14
6p	11302	388953847	1:13	218	800	0	8.54
7p	11356	391126410	1:18	312	822	.496	9.15
8p	11384	386778237	1:21	370	824	.743	7.94
9p	11447	386881431	1:24	415	852	1.3	7.96
10p	11376	386589160	1:24	431	834	.67	7.8
m Ps selected in some sort of classification scheme							
5p	11548	375844491	59min	0	784	2.195	4.89
10p	11716	389758467	33:23	0	402	3.68	8.77
24p	11917	392527618	16:40m	0	180	5.46	9.54
network	13470	579433340	1m43s			19.2	61.7

Table 4: How Number m & Infeasibles Affecting Performance

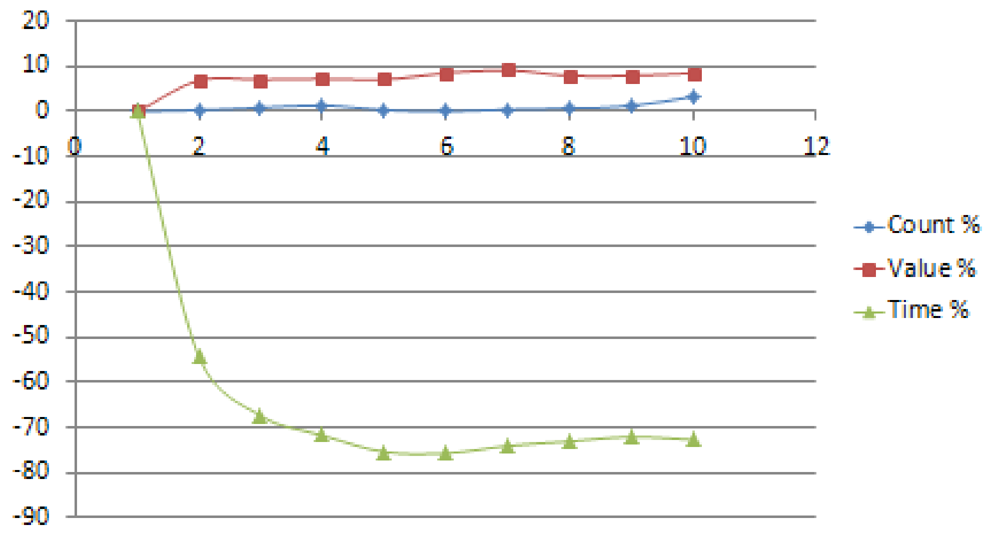

Figure 3: Suppression comparison with 1-LP for dg10

Data Name	t_{lp}	Prediction Time Model
asm10	2.1392	T = 3.29 + 12.913n_I + 2.1385n_{lp}
dg07 07	.8199	T = 16.993 + .4103n_I + .8085n_{lp}
dg10 07	4.8452	T = -43.89+2.34n_I + 4.8149n_{lp}

Table 5: Prediction Time Model as a Function of number of Infeasible(n_I) and LP (n_{lp}) (t_{lp}: Average time per solver based on $m = 1$)

3. Conclusions

As it is predicted, generally, the running time is reduced to $1/m$ of its 1-LP process under ideal conditions. Larger m can be used as data size increases. So the running time can be affordable. Still to be studied is to find/develope "classification - m Ps selection algorithm" to minimize infeasibles and possibly optimize suppression.

While the savings on time is significant, the obvious trade-off is that it leads to more suppresed cells and value which is a known fact. However, it is much better than network process in terms of suppression on both number of cells and value.

References

[1] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest Introduction to Algorithms. Chapter 27 The MIT Press 1992

[2] Fischetti M., and Salazar-Gonzalez J-J. Combining Complete and Partial Cell Suppression Methodologies in Statistical Disclosure Control. Statistical Journal of the United Nations ECE 18 (2001) 355-361

[3] Fischetti M., and Salazar-Gonzalez J-J. Models and Algorithms for the 2-dimensional cell suppression problem in statistical disclosure control. Math. Program. 84:283-312(1999)

[4] Filipa D. C., Nico P. D., Margarida S. O. Statistical Disclosure in Two-Dimensional Tables Journal of the American Statistical Association. Vol. 89 (1994), No. 428, 1547-1557

[5] Massel P. Using Linear Programming for Cell Suppression in Statistical Tables: Theory and Practice. Proceeding of JASA, 8/2001

[6] Cox L.H. Suppression Methodology and Statistical Disclosure Control. Journal of the American Statistical Association 75: 377-385

[7] Wang B. Disclosure Protection A New Approach to Cell Suppression. Proceeding of JSM 2008