Manuscript of / please cite as
Anholcer, M., Babiy, V., Bozóki, S., Koczkodaj, W.W. [2011]:
A simplified implementation of the least squares solution
for pairwise comparisons matrices,
Central European Journal of Operations Research **19**, pp.439-444.
DOI 10.1007/s10100-010-0134-y

http://www.springerlink.com/content/0535746j30g587x1

A simplified implementation of the least squares solution for pairwise comparisons matrices

Marcin Anholcer
Poznań University of Economics
Al. Niepodległości 10, 61-875 Poznań, Poland
V. Babiy
McMaster University, Computer Science
Hamilton, 1280 Main Street West, Ontario L8S 4L8, Canada
Sándor Bozóki
Laboratory on Engineering and Management Intelligence
Research Group of Operations Research and Decision Systems
Computer and Automation Research Institute, Hungarian Academy of Sciences
1518 Budapest, P.O. Box 63, Hungary
W.W. Koczkodaj *
Laurentian University, Computer Science
Sudbury, 935 Ramsey Lk Rd., Ontario P3E 2C6, Ontario, Canada

Abstract

This is a follow up to "Solution of the least squares method problem of pairwise comparisons matrix" by Bozóki published by this journal in 2008. Familiarity with this paper is essential and assumed. For lower inconsistency and decreased accuracy, our proposed solutions run in seconds instead of days. As such, they may be useful for researchers willing to use the least squares method (LSM) instead of the geometric means (GM) method.

Keywords: Pairwise comparison matrix, Least squares approximation, inconsistency analysis, Generalized Reduced Gradient algorithm, optimization

MSC: 05C78

1 Introduction

Finding a consistent approximation for a given inconsistent pairwise comparisons (PC) matrix by the least squares method for an Euclidean metric

*Corresponding author, email: wkoczkodaj@cs.laurentian.ca

was recently presented in [1]. The inspiration for writing this short note was the 3 *days entry* in Table 2 in [1] as the CPU time required to compute the case of a matrix for $n = 8$. We concluded that not many users are patient enough to wait three days for the results to be computed. It is important to mention here that real-life applications may require modifications of the values in the PC upper triangle many times and 50 or more changes are not uncommon. With each change requiring three days of computations, we would need additional 150 days to complete our project.

The problem under consideration is of the following form (we present the normalized version, see e.g., [1] for details): $\min \sum_{i=1}^{n} \sum_{j=1}^{n} (v_i/v_j - a_{ij})^2$ for $\sum_{j=1}^{n} v_j = 1$ and $v_j > 0, j = 1, \ldots, n$.

As shown in [1], it may have multiple solutions. However, all known examples having "distinct" solutions are far enough from solutions that appear in real-life situations. We are almost sure (subject to further research) that multiple solutions may appear when high inconsistency takes place, as explained below. However, selecting the one which is the closest to a GM solution (or simply the GM) is a practical remedy for this problem.

2 Practical assumptions for the simplification

It is a realistic assumption that the pairwise comparisons method is predominantly used for processing highly subjective assessments. Subjective assessments need this method for processing. For processing measurements or objective data, there are nearly always methods based on mathematical formulas, equations, partial differential equations, or a system of linear equations. So, we decided to decrease the accuracy to two significant figures since subjective assessments do not reach one percent accuracy. We recommend a geometric means (GM) solution as a starting point for better convergence since GM and LSM solutions are identical for fully consistent matrices and they are not drastically different for matrices which have a low inconsistency indicator (as defined in [3]). In a situation where the low inconsistency does not guarantee a single solution or a unique solution, we can always select the one which is closest to the GM solution by the Euclidean distance or revert to the GM solution.

The importance of the inconsistency analysis and control was stressed in [3] but better presented in [2]. By the GIGO (garbage in, garbage out) rule, the search for a very precise solution for a highly inconsistent pairwise comparisons matrix does not make much sense since the high inconsistency indicates the presence of contradictory assessments (probably on the basis of a cycle such as $a > b$, $b > c$, $c > a$) or partial contradictory assessments such as ratios: $A/B = 2$ and $B/C = 2$ yet $A/C = 5$. Apparently, $A/C = 5$ may be correct if either $A/B = 2.5$ or $B/C = 2.5$ were entered into our PC matrix. We may never know which of the three ratios was incorrect,

but identifying such a triad is a considerable step forward. Reducing a triad to two ratios is not possible. For only two ratios, we can have only inaccuracy but not inconsistency. Inconsistency is caused by excessive and conflicting data. The careful reader may have noticed that there is no precise solution to: $A/B = 2$ and $B/C = 2$ yet $A/C = 5$. Only compromised solutions exist. Compromised values may be: $A = 2.15443469$, $B = 1$, and $C = 0.464158883$ with the reconstructed ratios: $A/B = 2.15443469$, $B/C = 2.15443469$, and $A/C = 4.641588834$ which are definitely different from the given ratios $(2, 5, 2)$.

The hierarchical structure and a limit of pro members in each group is another reasonable limit assumed for our implementation. Seven elements give 21 pairwise comparisons and is a realistic assumption for the human impatience limit (the number of comparisons grows with the square). Neither of our solutions assumes any limit for n, but the growing n is expected to impact the speed of computations and their accuracy.

3 Practical solutions

In order to show the advantages of the idea of using the GM solution we used Solver (standard tool attached to MS Excel) and our randomized algorithm implemented in Java. The details of both attempts are described below.

MS Excel's Solver can be used to obtain results in less than one second for every tested case of n from 3 to 7. The accuracy is higher than we needed. According to Microsoft:

> Excel Solver uses the Generalized Reduced Gradient (GRG2) Algorithm for optimizing nonlinear problems. GRG2 is based on quasi-Newton algorithm as its default choice for determining a search direction. This algorithm was developed by Leon Lasdon, of the University of Texas at Austin, and Allan Waren, of Cleveland State University.

There are, however, two authors in [5] (a downloadable document available on the Internet) with one of the references to [4]. There is no question about the soundness of this well-known optimization algorithm.

Appendix A illustrates an example of how MS Excel Solver can be used for a 4 by 4 case. MS Excel Solver has one significant drawback: it produces numbers which are often hard to use for further specialized processing. Thus, we used another approach to solve this problem: a randomized local search algorithm in Java. It is also fast (milliseconds of CPU time) and accurate enough (we assumed 0.001 accuracy for all coordinates of our solution vector). The main idea of the method is to perform a line search, while each time only one randomly chosen coordinate is changing. The key to success was choosing the GM solution to be a starting point.

Appendix B shows a description of the algorithm used for Java implementation. The authors will eagerly share the Java code with readers but 10 pages of code is too large for including here. It is posted (together with the MS Excel Solver solution) on the Internet at a URL specified in [6] with a Readme.txt file which provides information about files and their use. A combination of a decision table with pseudocode has turned to be the most efficient way of what is a very simple case of a Monte Carlo simulation.

4 Conclusions

This presentation has removed one big shortcoming of LSM which was the substantial CPU time. However, an essential shortcoming of the least squares method is the high sensitivity with respect to outliers. The only solution is to avoid outliers by the inconsistency analysis as explained in section 2.

For subjective assessments, a high accuracy of a solution is not important. Two significant digits give an accuracy of one percent. It is more than sufficient for the input data which is often on a scale of 1 to 5 (used in [3]) or 1 to 9 (used in [1, 2]) and the distance-based inconsistency indicator with the acceptable level assumed to be $\frac{1}{3}$, as explained in [3].

We do not claim that our method may work for every pairwise comparison matrix but it is fast (fractions of a second instead of hours) for not-so-inconsistent pairwise comparison matrices that appear in most real-life problems. For $n = 4$, it required 3090 changes in the solution vector and 22,938 for $n = 7$. If it does not work, it is very likely because of high inconsistency for which using the initial geometric means solution is good enough.

5 Acknowledgments

This research has been supported in part by NSERC grant in Canada and by OTKA grants K 60480, K 77420 in Hungary.

References

[1] Bozóki S (2008) Solution of the least squares method problem of pairwise comparisons matrices. Central Eur J Oper Res 16:345–358

[2] Bozóki S, Rapcsák T (2008) On Saaty's and Koczkodaj's inconsistencies of pairwise comparison matrices. Journal of Global Optimization 42(2):157–175

[3] Koczkodaj WW (1993) A New Definition of Consistency of Pairwise Comparisons. Mathematical and Computer Modelling 18(7):79–84

[4] Lasdon LS, Warren AD, Jain A, Ratner M (1978) Design and testing of a generalized reduced gradient code for nonlinear programming. ACM Trans Math Software 4:34–50

[5] Lasdon LS, Warren AD (1997) GRG2 user's guide (posted on the Internet), p. 50.
http://www.cadfamily.com/download/CAE/isight8/Grg2User.doc
http://www.docstoc.com/docs/2137299/GRG2-user-guide
(accessed 2009-12-30)

[6] http://www.cs.laurentian.ca/wkoczkodaj/LSM2implementation.zip
(posted 2009-12-30)

Appendix A: Using MS Excel solver

1	A	B	C	D	E	F
2	**Matrix A**				**GM**	**GM norm**
3	**1**	2	5	9	3.0800703	0.521813
4	0.5	**1**	3	8	1.8612097	0.315318
5	0.2	0.333333	**1**	4	0.7186082	0.121744
6	0.111111	0.125	0.25	**1**	0.2427459	0.041125
7				**Sum:**	**5.9026341**	
8	0.4599501	0.3736534	0.1183795	0.048017		
9	**Matrix B**					
10	**1**	1.230954	3.885385	9.578903		0.4599501
11	0.812378	**1**	3.156402	7.781692		0.3736534
12	0.257375	0.316816	**1**	2.465368		0.1183795
13	0.104396	0.128507	0.405619	**1**		0.048017
14					**Sum:**	**1**
15	**Matrix A-B**					
16	**0**	0.591432	1.242367	0.335129		
17	0.09758	**0**	0.024462	0.047658		
18	0.003292	0.000273	**0**	2.355096		
19	4.51E-05	1.23E-05	0.024217	**0**		
20			**Sum:**	**4.721563**		

- Enter matrix A into A3 :D6 by setting "1"s on the main diagonal, filling the upper triangle with input numbers, and $a_{ji} = 1/a_{ij}$ in the lower triangle (e.g., A4 = 1/B3, A5 = 1/C3).
- GM = vector of geometric means, e.g. E3 = (A3∗B3∗C3∗D3)^(1/4). Copy the formula to E4 :E6.
- E7 =SUM(E3 :E6) is the sum of all GMs.
- Enter F3 =E3/$E$7 (remember to use absolute reference in $E$7). Copy the formula to F4 :F6.
- Copy matrix A (A3 :D6) to $A8$
- Copy 'special' (F3 :F6) into F10 :F13 as values and sum them up in F14 (the sum should be equal to 1).
- Enter the values of GMs into A8 :D8 using formulas, e.g. A8 =F10, B8 =F11 and so on.
- Reconstruct the upper triangle of A10 :D13 from F10 :F13 and A8 :D8 by dividing v_i/v_j (v_i are in F10 :F13 and v_j are in A8 :D8)
- Set A16 = (A3−A10)^2 and copy to A16 :D19.

- Set D20 =SUM(A3 :D16).
- Start the Solver
- 'Set Target Cell' to $D$20;
- Select Min for 'Equal to';
- Set 'By changing cells' to $F$10 :$F$13;
- Set 'Subject to constrains:' to $F$14 = 1;
- Choose options. Check the checkbox 'Assume non-negative'. Go back to the main window. Press the command button 'Solve'.
- The solution is in the cells $F$10 :$F$13. The optimal (minimal) value of LSM is in D20.

Appendix B: Algorithm for Java implementation

The vector $v = [v_1, .., v_n]$ is used for storing a solution; initially it is a normalized vector of geometric means.

The matrix B is reconstructed from $[v_i/v_j]$.

The i_{rand} is a random integer number in the range from 1 to n where n is the length of the matrix A.

The *delta* value initially assumed to be 0.001 for changing $v_{i_{rand}}$.

The *SumSQ* is computed as $\sum(a_{ij} - b_{ij})^2$.

Monte Carlo LSM		Rules				
Conditions	*SumSQ* is decreasing for *delta*	Y	N	N	Y	N
	SumSQ is decreasing for $-$*delta*	N	Y	N	N	Y
	delta unchanged 5 times	N	N		N	N
Actions	$v(i_{rand}) := v(i_{rand})$ + *delta*	X			X	
	$v(i_{rand}) := v(i_{rand})$ - *delta*		X			X
	delta := *delta*/2			X	X	X

while *SumSQ* not minimal **do**
  perform actions specified in the decision table
end while