

# Prioritizing Decisions using Chainwise Group Comparison

Steve Sherman

*Steve Sherman Publishing, LLC, 81 Applewood Drive, Marlborough, MA 01752,  
email: steve@sksherman.com*

## ABSTRACT

This paper introduces Chainwise Group Comparison as an integral part of an alternative multiple criteria analysis method which uses a rule-based, synthesis approach.

**Keywords:** Multi-Criteria Decision Making, Analytical Hierarchy Process, Chainwise Paired Comparison, Chainwise Group Comparison, Synergy.

## INTRODUCTION

Among the available approaches to multiple criteria decision making (MCDM) [1], Analytical Hierarchy Process (AHP) [2] seems most popular. AHP has been successfully applied using Chainwise Paired Comparison (CPC) [3] [4] to reduce computation time. This paper describes an alternative, less sophisticated approach to MCDM than AHP and introduces Chainwise Group Comparison.

One of the main criticisms of AHP involves its use of an arbitrary scale for pairwise comparison [5]. This paper explores a method inspired by AHP but which dispenses with its arbitrary scaling method. In dispensing with the arbitrary scale, another approach is enabled which also dispenses with the AHP method of calculating eigenvectors and eigenvalues. Instead, an iterative, rule-based, synthesis approach is used to resolve inconsistency. In so doing, users may compare criteria based on relative value of criteria without involving an arbitrary scale.

## CHAINWISE GROUP COMPARISON

Chainwise Group Comparison (CGC) is similar to CPC in that it simplifies comparisons. CPC requires the same order of comparisons as the number of criteria, avoiding the  $O(n^2)$  complication of AHP pairwise comparison. CGC involves comparing each criterion with other criteria in groups without assigning numeric values, so the comparisons of CGC are somewhat simplified and reduced over the normal pairwise comparisons of CPC and AHP.

CPC and CGC both involve weight calculation based on relative weights of successive criteria. CGC involves weight calculation based on relative importance of criteria but involves evaluation of weight in terms of groups of lower-priority criteria.

CPC and CGC both avoid the cost of calculating reciprocal values. It may be argued that this loss of redundancy also loses the feature in AHP of double-checking user input. However, CGC somewhat makes up for this in setting up potentially overlapping groups, requiring the user to consider each criterion preference more carefully.

## CGC Matrix

The CGC matrix contains criteria rows with boolean iterations of lesser criteria. It may be thought of as a block diagonal binary matrix imposed upon the identity matrix and perhaps a filled superdiagonal. All diagonal values are initially set to 1 and other iterations cleared to represent ungrouped, unprioritized criteria. Synergy for each criterion will be calculated for each respective row of the CGC matrix.

The CGC matrix is asymmetric, with each row having an entry that represents the respective criterion (with a corresponding 1 on the diagonal) as well 1-values iterating the criteria of other rows (iterations), the sum of whose weight the weight of this row should be greater than. Thus, 1-values in the lower triangle of the matrix represent criteria that are out of order. The lower triangle should be comprised only of 0-values when the matrix is consistent. 1-value criteria in the upper triangle denote criteria groupings from which synergy may be derived.

### Synergy

CGC introduces the concept of “synergy” during criteria comparison. The term “synergy” is used because the basic idea is to express value that exceeds the sum of parts. In this context, sums are calculated of weights of lesser priority criteria.

Rather than assign an arbitrary value to indicate relative importance as is done with AHD and CPC, a criterion is assigned a “synergy” value indicating its relative importance to a successive count of criteria of lesser importance per Table 1 where  $n$  is the number of criteria.

**Table 1:** Potential Synergy Values for  $n$  Criteria

Synergy Value	Weight
0	1
1	Lesser criteria weight + 1
2	Sum of lesser 2 criteria weights + 1
...	
$n - 1$	Sum of lesser $n - 1$ criteria weights + 1

#### *Synergy 0*

Synergy of value 0 means that weight will be the least value of 1. Either all criteria should have synergy 0 (equivalent weight of 1, identity matrix condition) or there should be only one entry with synergy value of 0 after the consistent matrix has been filled and analyzed. In the latter case, the synergy 0 row is the only row that will not have a corresponding 1-value in the superdiagonal.

When priority exists, having only one, lowest priority criterion of synergy 0 is required so that there is no ambiguity about prioritized sequence. Otherwise, multiple criteria with 0 synergy values would be allowed to swap unpredictably as they would be of the same priority.

In practice, criteria are organized by hierarchy. Thus, there may be levels in the hierarchy where priority is the same for all criteria and thus appropriate for all criteria to be represented with synergy 0.

#### *Synergy Block Overlap*

Synergy blocks may not overlap beyond their boundary rows. Overlap would result in criterion weight calculation involving more than one synergy value, introducing inaccuracy.

#### *Synergy 1*

Synergy of value 1 means that the weight will be incremented relative to weight of the lesser criterion immediately below. By default, values may be set to 1 in the superdiagonal of the matrix when a new list of criteria is loaded. However, analysis will have greater freedom to resolve inconsistency if the superdiagonal is not yet filled. The values on the diagonal and the superdiagonal are not considered to comprise a synergy block. Thus, they will never overlap synergy blocks.

#### *Synergy Above 1*

Synergy of value greater than one means that there is a synergy block present extending to the criteria immediately below and that the weight of the top criterion in the synergy block will be incremented relative to the sum of weights of the lesser criteria in the synergy block.

## **Deriving Synergy from the CGC Matrix**

Synergy is calculated in synergy blocks by counting the number of sequential iterations to the right of the diagonal on each row. Synergy blocks may not overlap except for block boundary criteria.

### ***Filling in the CGC Matrix***

The matrix may be initially created with synergies of value 0, assuming that the criteria of a list are unprioritized and ungrouped. The resulting identity matrix is a consistent CGC matrix, so processing may conclude with this matrix.

The matrix may be filled per existing synergy values. This involves adding sequential iterations for the length of each synergy value on the respective row. As comparisons are performed and results analyzed, consistence and order may be determined.

The initial order of criteria in a list may indicate initial priority. The superdiagonal values of the matrix may all be set to 1, indicating the relative priority for all criteria of the matrix. That is, the criteria are kept in order of relative priority, each being an incremented weight, the most important criterion having highest weight at the top row of the matrix. This is also a consistent CGC matrix.

Further filling of the matrix involves up to  $n$  dialogs, one for each criterion. In each dialog, the user is presented with a criterion and with a list that includes the other criteria sorted by current priority. From this list, the user may define order and synergy by setting values in the CGC matrix. The order along the superdiagonal can be thought of as the “chain” while selection of additional criteria can be thought of as “group comparison”.

### ***Checking CGC Matrix Consistence***

Due to user selections, the CGC matrix may have inconsistency. These can be detected and resolved through analysis of the matrix. Rows may be moved or iterations cleared to attain consistence.

### ***Unfilled Superdiagonal***

To be consistent, the superdiagonal must be either entirely empty or entirely filled (except for the last row). A CGC matrix that is comprised only of the diagonal, a filled superdiagonal and with no synergy blocks is considered consistent. An initial filling of the superdiagonal may be done simply by extracting it from the criteria list that is first loaded. This filling may be done at any time but should not be done if priority is neither known nor implied by criteria order. By deferring default filling of the superdiagonal until last, the process will be less constrained in resolving inconsistency.

If the criteria list order is meant to imply priority, then this information should be used to fill the superdiagonal with 1-values.

At the end of analysis, if the superdiagonal is partially filled, then the matrix is considered inconsistent. If it is partially filled, then some synergy values are cleared and some are not, resulting in some ambiguity about row priority or order per synergy values.

### ***Orphan Criteria***

Inconsistence is readily detected by the presence of any 1-values on a row after the first 0 is detected to the right of the diagonal during synergy calculation. This is because groups must be formed of successive, lower-priority criteria. Iterations that are outside of synergy blocks indicate non-successive (orphan) criteria.

## CONSISTENCE RESOLUTION

Consistence resolution as presented here involves, in general order of execution, resolving order, resolving orphan criteria and ensuring there are no synergy block overlaps. Once a matrix is consistent, priority and synergy may be unambiguously extracted. Synergy will be resolved and weights may be readily calculated.

A number of rules and corollaries are presented here with the intention that each be sequentially applied for resolution. This is not an exhaustive list of rules. These are also not the only possible rules that may be applied.

Note that when response is provided to the user, it is often described here in terms of row moves or cleared iterations. The user, however, will be presented with a query based on what rows and criteria represent. For example, instead of asking a user to cancel iteration of one row by another, a query might be more like, "Is it okay to remove consideration of Option A from the group?" The user need not be burdened with the technical jargon of the CGC matrix or of the underlying processes involved.

### Order Resolution

Order is consistent if the lower triangle of the CGC matrix is cleared of any iterations. That is the goal of this step of analysis. This is largely accomplished by moving rows in the CGC matrix.

**Rule 1:** A row in a consistent CGC matrix must be below any row that iterates it in the upper triangle.

**Corollary 1:** A row may not be moved above a row that iterates it in the upper triangle.

The primary method for clearing the lower triangle involves moving rows (and corresponding columns). Before scanning the matrix, any rows that contain only the diagonal and no iterations of other criteria should be moved to the end of the matrix. After that, row movement must be determined more carefully and iteratively during scanning. After any change to the matrix row order, scanning must be restarted to ensure order consistence.

The goal of moving rows at this point is to move all iterations to the upper triangle. The approach described here involves a scan of rows for iterations in the lower triangle. An appropriate move is determined and exercised. Then, the rows are scanned again. This continues until either all lower triangle iterations are cleared or until it is not possible to move rows to resolve the issue. In the latter case, the user may be prompted for resolution.

Per Rule 1, if a row has a value of 1 in the lower triangle and the iterated row iterates that row, then inconsistency is detected that requires the user reconsider iterations.

**Table 2:** Row 1, Row 3  
Inconsistence

The CGC matrix in Table 2 violates Rule 1. Its iterations indicate that the weight of row 1 should be greater than the weight of row 3 while also indicating that the weight of row 3 should be greater than the weight of row 1. Per user response, one of the iterations must be cleared for the array to be made consistent.

Criteria	1	2	3
1	1	0	1
2	0	1	0
3	1	0	1

Note that the inconsistency detected here is similar to an inconsistency that might be detected using AHP pairwise comparison. A difference is, of course, that the conflict is detected here through direct comparison of binary values as opposed to assigning integer or fractional values to comparisons and then comparing these values after much more sophisticated analysis of eigenvalues, eigenvectors and so forth.

**Rule 2:** Iteration in the lower triangle indicates the row should be moved above the row iterated.

**Corollary 2:** A row that iterates a row in the lower triangle may be moved above the iterated row.

When 1-value, lower triangle criteria are detected in a row, that row becomes an inconsistent row. If the iterated row satisfies Corollary 2, then the rows may be swapped.

For example, the move in Table 3 of row 2 above row 1 (which includes a corresponding column move) results in a consistent matrix per Table 4. Once the lower triangle is clear, order is consistent and attention may turn to resolving orphan criteria.

**Table 3:** Inconsistent Lower Triangle

Criteria	1	2	3
1	1	0	1
2	1	1	0
3	0	0	1

**Table 4:** Resolution After Move

Criteria	2	1	3
2	1	1	0
1	0	1	1
3	0	0	1

### Resolving Orphan Criteria

Though the lower triangle is cleared and order is consistent, row order may yet be allowed to vary along with other adjustments so that orphan criteria may be eliminated in the upper triangle. The goal in this step of analysis is to resolve orphan criteria and, as a side effect, form synergy blocks.

**Rule 3:** Consistently iterated criteria in a row of the upper triangle must all be grouped together immediately after the diagonal and superdiagonal, potentially forming a synergy block.

**Corollary 3:** A non-iterated criterion may be moved past the last iterated criterion in a row so long as order consistence is preserved.

**Table 5:** Orphan Inconsistence **Table 6:** Orphan Resolution

Table 5 shows a CGC matrix with readily resolvable orphan inconsistency. This inconsistency may be resolved while maintaining consistent order by moving non-iterated row 3 past row 4 as shown in Table 6.

Criteria	1	2	3	4
1	1	1	0	1
2		1	0	0
3			1	0
4	0			1

Criteria	1	2	4	3
1	1	1	1	0
2		1	0	0
4			1	0
3	0			1

Rows with no iterations are moved to the end of the matrix during Order Resolution. This helps to avoid trivial orphan criteria instances.

Unresolvable orphan inconsistency requires user response to force row moves or remove iterations.

### Resolving Synergy Block Overlap

At this point of processing, the lower triangle is cleared and orphan criteria have been eliminated in the upper triangle. The goal in this step of analysis is to resolve synergy block overlaps.

At this point, the order should not vary. So, it is appropriate to begin by filling the superdiagonal so that order is set. In the final result, most synergy values within the matrix will be 1 except for the rows at the top of each synergy block and the lowest priority row (set to 0). At this point, synergy can be calculated from the CGC matrix and further processing done directly with the resulting synergy vector.

In general, synergy block overlaps are only resolved through user response as they are detected, usually resulting in synergy block truncation or removal.

### Detecting Synergy Block Overlap

Synergy blocks are defined by criteria that have synergy values greater than 1. Thus, a synergy vector can be calculated directly from the synergy blocks of a CGC matrix.

**Rule 4:** A synergy vector entry that has value greater than 1 determines the start of a synergy block as well as how many of the consecutive criteria that follow are included in the synergy block.

**Corollary 4:** A synergy block overlap is detected if any of the criteria within a synergy block have synergy greater than 1, aside from the last criterion.

Per Corollary 4, the synergy vector can be scanned sequentially, detecting synergy blocks and then counting off sequential criteria between synergy blocks. Any synergy value greater than 1 within a synergy block, aside from the last criterion, results in synergy block overlap detection, which is an inconsistency.

**Synergy Block Overlap Resolution**

Options for resolving a synergy overlap (such as in the synergy vector of Table 7 with synergy 3 overlapping synergy 2) include lower synergy block removal and upper synergy block truncation. For lower synergy block removal, synergy may be set to 1 (Table 8, criterion 2). For upper synergy block truncation, the upper synergy block may be modified such that the last criterion of the upper synergy block is also the starting criterion of the lower synergy block (Table 9, criterion 1). During truncation, if the lower synergy block immediately follows the upper synergy block, then the upper synergy block will be effectively removed.

**Table 7:** Synergy Block Overlap

Criteria	Synergy	Weight
1	3	8
2	2	4
3	1	2
4	0	1

**Table 8:** Block Removal

Criteria	Synergy	Weight
1	3	7
2	1	3
3	1	2
4	0	1

**Table 9:** Block Truncation

Criteria	Synergy	Weight
1	1	5
2	2	4
3	1	2
4	1	1

**CONCLUSION**

This paper has introduced CGC matrix analysis which can serve as part of a decision making application to resolve order and weight. This analysis includes resolving inconsistency in the matrix which may include improper order, orphan criteria and overlapping synergy blocks. Where resolution cannot be readily determined per given iterations, user response can be used to resolve inconsistency. Once consistent, order is determined and the resulting synergies and prioritized weights can be calculated.

**REFERENCES**

[1] Dyer, J.S., Fishburn, P.C., Steuer, R.E., Wallenius, J., and Zionts, S. (1992). "Multiple Criteria Decision Making, Multiattribute Utility Theory: The Next Ten Years," *Management Science*, V38, N5, May, pp. 645-654.

[2] Saaty, T.L. (1980). *The Analytic Hierarchy Process*, New York: McGraw-Hill, 1980.

[3] Wallin, P, Fröberg, J., Axelsson, J. (2007). "Making Decisions in Integration of Automotive Software and Electronics: A Method Based on ATAM and AHP," Fourth International Workshop on Software Engineering for Automotive Systems (SEAS'07)

[4] Ra, J. W. "Chainwise paired comparison", *Decision Sciences*, 30(2):581-599, 1999.

[5] McCaffrey, James (June, 2005). "Test Run: The Analytic Hierarchy Process". *MSDN Magazine*.