Crowdsourcing for Decision Making with Analytic Hierarchy Process

ISHWOR TIMILSINA, The University of Texas at Arlington
SUPERVISED BY: CHENGKAI LI, The University of Texas at Arlington

Analytic Hierarchy Process (AHP) is a Multiple-Criteria Decision-Making (MCDM) technique devised by Thomas L. Saaty. In AHP, all the pairwise comparisons between criteria and alternatives in terms of each criterion are used to calculate global rankings of the alternatives. In the classic AHP, the comparisons are provided collectively by a small group of decision makers. We have formulated a technique to incorporate crowd-sourced inputs into AHP. Instead of taking just one comparison for each pair of criteria or alternatives, multiple users are asked to provide inputs. As in AHP, our approach also supports consistency check of the comparison matrices. The key difference is, in our approach, we do not dismiss the inconsistent matrices or ask users to reevaluate the comparisons. We try to resolve the inconsistency by carefully examining which comparisons are causing the inconsistency the most and then getting more inputs by asking appropriately selected questions to the users. Our approach consists of collecting the data, creating initial pairwise comparison matrices, checking for inconsistencies in the matrices, try to resolve the matrices if inconsistency found and calculating final rankings of the alternatives.

1 INTRODUCTION

Decision making is an integral part of today's world, may it be governmental, medical, corporate, or just small group decisions. The stakes are higher than ever and decision makers want to make sharp, calculated decisions rather than hunches. Among all the decision-making tools and techniques available, Analytic Hierarchy Process (AHP)[9, 12] is one of the most popular ones. AHP, a multi-criteria decision making technique, was first introduced by Thomas L. Saaty[9, 12] in 1980. AHP helps make complex decision by breaking them down into smaller problems that are then solved by pairwise comparisons. AHP particularly stands out for qualitative judgments, although it works well for both subjective as well as objective comparisons. The comparisons are mostly human judgments and can vary between individuals. AHP is applicable to large set of problems, but it is most commonly used in industry-related applications[4].

In AHP, the decision makers are asked to compare pairs of criteria and pairs of alternatives/objects based on each of those criteria. All the pairwise comparisons (comparing item i with item j) are then used to obtain weights of criteria and relative weights of alternatives in terms of each criterion. The AHP additionally gives methods to check if the comparisons are consistent or not. This consistency check makes sure the comparisons provided are not random in nature and the end result obtained is trustworthy.

Countless entrepreneurs and corporations from search engine giants like Google, and social networking platforms like Facebook and Twitter to online businesses like Amazon and EBay have cashed in on the internet culture. This plethora of user participation has opened a doorway to crowdsourcing. Crowdsourcing (crowd + outsourcing) is a model that, as its name suggests, divides up the work and distributes it to an undefined public. This model has several advantages - diversity, costs, scalability, speed being some of them[4]. The 'Kickstarter' campaigns nowadays essentially mean crowdsourcing of funding. Wikipedia is arguably the best example of crowdsourcing. This made us think if and how crowdsourcing could be used for decision making purposes. This paper explores the use of crowd in decision making using the basic model of AHP.

While conventional AHP takes only one input per pairwise comparison, we allow each pairwise comparison to have multiple inputs. Our approach also allows for consistency check but it doesn't require dismissal or reevaluation of matrices if those matrices are found inconsistent. We have devised a method which keeps the already gained inputs intact yet still tries to resolve the inconsistency in the matrices by taking additional inputs from users for suitable pairs of comparisons.

2 BASICS OF AHP

AHP is a relative measurement[2] technique, where proportions between pairs are more important than exact measurements of entities. It uses a set of pairwise comparisons to calculate final ranking of the objects/alternatives. These pairwise comparisons may be between any two criteria and/or any two alternatives. First, AHP assigns score for each criterion per decision maker's pairwise comparisons of all the criteria. Then, for each criterion, AHP assigns score to alternatives based on the pairwise comparisons of all the alternatives based on that criterion. Finally, mathematical operations are done on the criteria weights and the alternatives weights in terms of all the criteria, to determine the global score of each alternative and subsequent rankings of the alternatives.

The Analytic Hierarchy Process (AHP) can be briefly described in below basic steps:

(1) Model the problem as a hierarchy - goal, criteria, and alternatives.
(2) Give the priorities to the criteria:
    (a) Give the scores for each pairwise comparison (j, k) of the criteria, thus creating an m x m matrix where m is the number of criteria (say C). Table 1 below shows the different scores for comparisons of two entities i and j (given by Saaty[9] himself):

Value	Meaning
1	j and k are equally important
3	j is slightly more important than k
5	j is much more important than k
7	j is strongly more important than k
9	j is absolutely more important than k
2,4,6,8	Intermediate values if needed
1/3,1/5,1/7,1/9	The reciprocal values show comparison of j to i. If (i, j) = 1/3, then (j, i) = 3.

Table 1. Different scores for comparisons in AHP

    (b) Calculate a priority weight vector for the criteria.
(3) Calculate the weights of different alternatives with regard to each criterion creating n x n matrix where n is number of alternatives (the process is like step 2)
(4) Check for the consistency of the intermediate results (after creation of matrices both in steps 2 and 3)
(5) Come to the final decision

A simple illustrative example of AHP is shown.
Goal: Select the best movie out of these alternatives
Criteria: Story, Direction, Acting, Music/Score
Alternatives: Schindler's List (SL), Silence of the Lambs (SOTL), American Beauty (AB), Lord of the Rings(LOTR)

2.1 Setting the hierarchy

The hierarchy of our problem becomes:

2.2 Assigning pairwise comparison scores and creating comparison matrix

Let's assign scores for each pairs of criteria. $C \epsilon R^{mxm}$ is a pairwise comparison matrix if it satisfies following three properties:

- $C_{i,j} > 0$
- $C_{i,i} = 1$

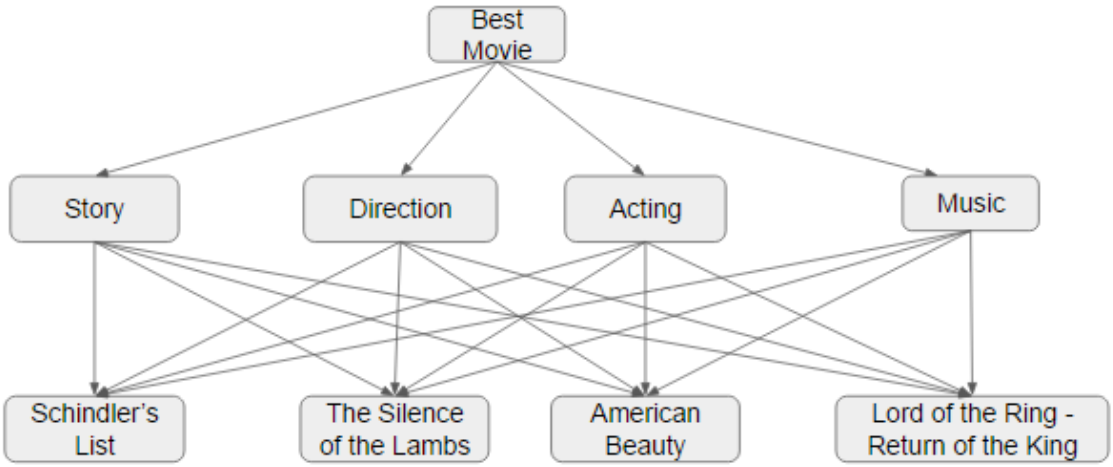

Fig. 1. Hierarchy for our goal, criteria and alternatives

- $C_{j,i} = 1/C_{i,j}$

Create an m x m matrix (C) where m is the number of criteria. In the following matrix (table 2), each individual cell is the ratio between the corresponding criteria. For instance, story-direction comparison, $C_{1,2} = 1/3$ means direction is slightly more important than story which is same as the $C_{2,1} = 3$.

	Story	Direction	Acting	Music
Story	1	1/3	5	7
Direction	3	1	7	9
Acting	1/5	1/7	1	3
Music	1/7	1/9	1/3	1

Table 2. Example - Criteria comparison matrix (C)

2.3 Calculating priority weights

Now we calculate the priority weights of all the criteria using the criteria matrix (C) from Table 2. There are three of the most widely used ways by which we can calculate the weights.

(1) Mean of normalized values
This is the oldest method and consists of following three basic steps: [1]
   (a) Sum of elements of column j
   (b) Normalization of column j
   (c) Mean of row i

(2) Geometric Mean approach (Crawford and Williams[3])
In this approach, the geometric means of all the rows results in the priority weight vector. This approach minimizes logarithmic errors[3], although Saaty[1, 10] criticizes it saying there's no conceptual justification for working with logarithmic scale. In this approach, we multiply all the elements of each row in the

matrix and then we calculate the n^{th} root of the product to get a vector. Once we normalize that vector, we get the priority weight vector for that matrix.

(3) Eigenvector approach
This approach was proposed by Saaty[9, 12] himself. This approach says, the principal eigen vector of the matrix is the desired weight vector. This is a very commonly used method for calculating the weights of criteria and alternatives. The weight vector is the normalized solution of:

$$Mw = \lambda_{max} w \tag{1}$$

where M is the matrix, w is the eigenvector of M, and λ_{max} is the maximal eigenvalue, λ_{max} >= m (Saaty [11]). In real world scenario with the participation of human on qualitative judgements, λ_{max} > m (Saaty[11]). According to the Perron-Frobenius Theorem, the eigenvalue λ_{max} is positive and real[7].
To calculate the principal eigenvector, we use a numerical method called 'power method'[6]. The power method is iterative process consisting of following steps:
(a) The matrix M is squared.
(b) All the elements in a row are then added together.
(c) The result is normalized to get approximate eigen vector.
(d) Repeat step a through c with the matrix we got from step a.
(e) Repeat step d until the row elements in two consecutive approximate eigen vectors are smaller than certain small threshold (eg: 0.0001).

Although many authors have complained about the transparency of the eigen value approach[5], the creator of AHP, Saaty, insists on the strength of this method and has justified it against inconsistent matrices[11].

We will use the Eigenvector approach for the demonstration of basics of AHP in this section, as well as in our approach later. After using the power method, we get the eigenvector of matrix C which will also the priority weight vector.

Story	0.2903
Direction	0.5824
Acting	0.085
Music	0.0423

Table 3. Priority weight vector (W) for matrix C from Table 2

Looking at this weight vector, we can easily determine that 'Direction 'is the most important/impactful criteria because it has the largest weight value.

2.4 Checking for Inconsistency

Finding the weight vector of a matrix is not enough. We need to make sure that the comparisons in the matrix are consistent enough to trust the end result they will yield. Human inconsistency is inevitable. Especially in the subjective matters where people compare based on their opinion rather than facts. To avoid the ill effect these inconsistent comparisons may create on the resulting ranking, it is better to check if the comparisons are consistent and if not, discard and/or redo them. AHP provides the mechanism for testing the consistency of a comparison matrix.

First, we need a temporary vector that we will get by multiplying the original criteria matrix by its weight vector.

$$W_T = \{C\}\{W\} \tag{2}$$

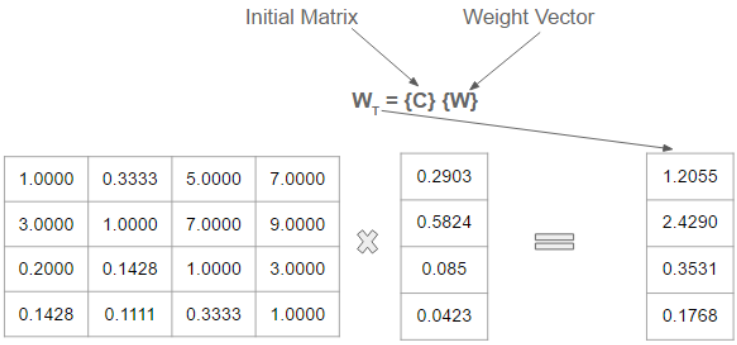

Fig. 2. Multiplying initial matrix by its weight vector to get W_T

Now, divide each element of W_T by its corresponding element of W.

$$W_{cons} = W_T.1/W \tag{3}$$

For example, W_{T1}/W_1 = 1.2055/0.2903 = 4.1526. Doing this for all elements, we get,

$$W_{cons} = \{4.1526, 4.1707, 4.1541, 4.1797\}$$

If we take the average of all these elements in W_{cons}, we get the maximal eigenvalue λ_{max} = 4.1643.

Next step is to calculate consistency index (CI). For m number of criteria,

$$CI = \frac{(\lambda_{max} - m)}{(m-1)} \tag{4}$$

Thus,

$$CI = \frac{4.1643 - 4}{4 - 1} = 0.0547$$

The matrix would be perfectly consistent if CI = 0 i.e., λ_{max} = m. But we cannot expect it to be so with human judgments. For this purpose, we need to calculate consistency ratio (CR).

$$CR = \frac{ConsistencyIndex(CI)}{RandomIndex(RI)} \tag{5}$$

Random Indices have already been calculated by multiple sources including Saaty[12] to almost same results.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.000	0.000	0.525	0.880	1.109	1.248	1.342	1.406	1.450	1.485	1.514	1.537	1.555	1.571	1.584

Table 4. Random Indices for different numbers of criteria/alternatives

If CR >= 0.1, then the pairwise comparisons should be reevaluated. Otherwise, the matrix is considered near-consistent. In our example,

$$CR = \frac{0.0547}{0.880} = 0.0621$$

Our CR is less than 0.1 so the matrix is acceptable.
Now that the criteria comparison matrix is created and verified to be near-consistent, it is time to create alternative comparison matrices in terms of each criterion.

	SL	SOTL	AB	LOTR
SL	1	3	5	1
SOTL	1/3	1	3	1/3
AB	1/5	1/3	1	1/5
LOTR	1	3	5	1

Table 5. Alternatives comparison matrix in terms of Story (CR=0.016)

	Story
SL	0.3909
SOTL	0.1509
AB	0.0674
LOTR	0.3909

Table 6. Priority weight vector for Table 5

	SL	SOTL	AB	LOTR
SL	1	1/3	1/3	3
SOTL	3	1	3	5
AB	3	1/3	1	5
LOTR	1/3	1/5	1/5	1

Table 7. Alternatives comparison matrix in terms of Direction (CR=0.073)

	Direction
SL	0.1465
SOTL	0.4995
AB	0.2883
LOTR	0.0655

Table 8. Priority weight vector for Table 7

	SL	SOTL	AB	LOTR
SL	1	1/5	1/3	5
SOTL	5	1	3	7
AB	3	1/3	1	5
LOTR	1/5	1/7	1/5	1

Table 9. Alternatives comparison matrix in terms of Acting (CR=0.0887)

	Acting
SL	0.1326
SOTL	0.1326
AB	0.2609
LOTR	0.048

Table 10. Priority weight vector for Table 9

	SL	SOTL	AB	LOTR
SL	1	1/3	3	1/7
SOTL	3	1	5	1/5
AB	1/3	1/5	1	1/9
LOTR	7	5	9	1

Table 11. Alternatives comparison matrix in terms of Music (CR=0.0631)

	Music
SL	0.0955
SOTL	0.2044
AB	0.0456
LOTR	0.6545

Table 12. Priority weight vector for Table 11

2.5 Calculating Final Rankings of the alternatives

After we have all the near-consistent matrices and their respective weight vectors, we are ready to calculate the final rankings of the alternatives. For that, we combine all the alternatives'weight vectors with each vector as a row to form a matrix. Then, we multiply this formed matrix by the criteria weight vector (Table 3) as shown in Fig 3. The result will yield the final rankings. Finally, we get the global rankings of the alternatives.

	Story	Direction	Acting	Music/Score
Schindler's List	0.3909	0.1465	0.1326	0.0955
The Silence of the Lambs	0.1509	0.4995	0.5585	0.2044
American Beauty	0.0674	0.2883	0.2609	0.0456
LOTR - Return of the King	0.3909	0.0655	0.048	0.6545

Story	0.2903
Direction	0.5824
Acting	0.085
Music	0.0423

Fig. 3. Multiplying matrix formed by combining all alternatives weight vectors by criteria weight vector

Schindler's List	0.2141
Silence of the Lambs	0.3908
American Beauty	0.2116
LOTR - Return of the King	0.1834

Table 13. Final global rankings of all the alternatives

As we can infer from in Table 13, Silence of the Lambs appears to the best movie among these four alternatives followed by Schindler's List, American Beauty, and Lord of the Rings - Return of the King.

3 OUR APPROACH

We had to make some modifications and additions to the AHP model to accommodate our objective, i.e., allowing many users (crowd) to participate in the decision making process.

3.1 Minimal Sample Size

First we want to get minimal number of inputs for all the comparisons but still want the samples to give us the means that are representative enough of the whole population. Since we do not know the actual population size, minimal sample size for estimating the population mean is given by following formula [8],

$$n = \frac{1}{\frac{d^2}{z^2_{\alpha/2}.\sigma^2} + \frac{1}{N}} \tag{6}$$

where,
d = margin of error (moe)
$z_{\alpha/2}$ = z-score for certain confidence level
σ = standard deviation
N = total estimated population

We do not know the population size, but we know that the population will be potentially large (crowdsourcing). For this reason, we assign some arbitrary large value to N (100000 should be fine).
We also do not have an exact standard deviation, so we also need to estimate the standard deviation of the population. To estimate the standard deviation, we use the 'Range Rule' which says,

$$\sigma = \frac{range}{4} \tag{7}$$

3.2 Use of confidence interval as threshold during initial inputs

We wanted to make the samples of inputs as useful as possible. Since, after calculating the minimal sample size, we ask users to provide the inputs, we do not know how variant their inputs are going to be. The sample should have certain confidence for it to be useful.
To check this,

(1) calculate confidence interval of the previously created minimal sample at a certain confidence level (CL) (e.g. 95%)
   (a) calculate degree of freedom(df), df = sample size (n) - 1
   (b) calculate the mean(μ) and standard deviation(σ) of the sample
   (c) subtract the confidence level from 1, then divide by two

$$\alpha = \frac{1 - CL}{2} \tag{8}$$

   (d) look up t-score in t-distribution table for the degree of freedom and α
   (e) calculate standard error (SE),

$$StandardError(SE) = \frac{\sigma}{\sqrt{n}} \tag{9}$$

   (f) multiply t-score by SE
   (g) confidence interval = $\mu \pm$ step f
(2) check if result of step 1(f) (<=) certain threshold (Threshold_1)
(3) Take more inputs until the resulting sample satisfies step 2.

3.3 Changing scales for the inputs

The scale used in traditional AHP contains value between 1/9 to 9, where 1/9, 1/7, 1/5, 1/3 are reciprocal and signify practically opposite meanings of 9, 7, 5, 3. For example, if 3 means a is slightly more important than b, then 1/3 means a is slightly less important than b or b is slightly more important than a. We found this scale unhelpful in few stages in our methodology. The reason for that is, as we will find out later, we use thresholds on confidence intervals (see sections 3.3 and 3.5) to limit the number of inputs taken for each unique pair. While setting the threshold, the same threshold needs to work for all the pairs. The AHP scale does not have uniform intervals. Example: 1 to 3 has the same interval as 3 to 5 or 5 to 7, but it does not have the same interval as 1 to 1/3 or 1/3 to 1/2. So, while taking user inputs, we map the inputs into our scale. Our scale has uniform intervals throughout its range. The new scale is equivalent to AHP's scale in following manner (Table 14):

3.4 Initial Matrices Creation

After we get enough inputs from the users, we want to create one matrix for criteria and each one for alternatives in terms of each individual criterion. Furthermore, the matrices need to be fully formed i.e., they should not contain any null or zero values to be functional. To populate the matrix, we take the mean value from a sample of the corresponding pairwise comparison. For m number of criteria/alternatives, the matrix will have m(m-1)/2 unique comparisons cells with each having their reciprocal values and m diagonal cells having value of 1. We

AHP's input scale	Our Input Scale
1/9	-4
1/7	-3
1/5	-2
1/3	-1
1	0
3	1
5	2
7	3
9	4

Table 14. Mapping between our scale and AHP's scale

calculate the mean of samples for all the pairs. Since our data is the range [-4, 4] but AHP operates in the range [1/9, 9]. We convert all these means to the AHP scale for further operations like priority weight and consistency ratio calculations.

In simplest of words: for pair (A, B), say there are 6 inputs from users [a1, a2, a3, a4, a5, a6], then in Table 6, cell (A, B), a = (a1 + a2 + a3 + a4 + a5 + a6)/6. Once we have the value for cell (A, B) in the matrix, the value at cell (B, A) is simply the reciprocal of the value at (A, B) that is 1/3 in this example. Similarly, for pair (A, C), if there are 5 inputs from users, b = (b1 + b2 + b3 + b4 + b5)/5.

	A	B	C	D
A	1	a	b	c
B	1/a	1	d	e
C	1/b	1/d	1	f
D	1/c	1/e	1/f	1

Table 15. Populating a simple pairwise comparison matrix

3.5 Checking for inconsistency in the matrices

To check for inconsistency, we follow the same steps from conventional AHP. Once we create matrices by populating the cells with the mean values of corresponding samples of each pairwise comparison inputs, the matrices can be practically treated as the matrices that we use in traditional AHP.

If CR < 0.1, then the matrix is near-consistent and the weight vector generated from the matrix is considered useful for final ranks generation. But if CR >= 0.1, then the matrix is inconsistent and it needs to be resolved.

3.6 Resolving Inconsistency in a matrix (If any)

In normal AHP, the decision makers are asked to reevaluate the comparisons in the matrix. But in our case, that is not the most practical approach. We try to resolve the inconsistent matrix by changing the values in the cells by getting more inputs from the users. For few reasons, we don't want to take inputs for all the comparisons. First, it would be costly since it is more likely that a huge number of new inputs would be required for a fair problem. Second, we wouldn't even know if it would solve the problem. So, the best approach is

to select a cell whose change is most likely to make the matrix near-consistent. Then we ask for new input for that pair/cell. To find out which question to ask or which pair to ask the input for, we have devised a technique.

Finding the ripest cell to change in the matrix

To find the cell, in the inconsistent matrix, that has the best promise to resolve the matrix if changed, we follow these following steps:

(1) Start from the first cell of the matrix.
(2) See what the confidence interval of the sample that generated the value used in that cell is.
(3) Keep the values as they are in all the other cells (that is, the mean value of the samples that we used to populated the matrix).
(4) Substitute the value in the current cell with left range of the confidence interval.
(5) Calculate the new CR for this new matrix.
(6) Now, substitute the value in the current cell with the right range of the confidence interval.
(7) Calculate the new CR for this new matrix.
(8) Record these two new CRs of the matrix and corresponding cell index.
(9) Iterate through all remaining cells of the original matrix one by one and repeat step 2 through step 7.
(10) Peruse through all these new calculated CRs and see which CR improves the original CR of the matrix. By 'improve', we mean the CR that is the smallest but is still less than the original CR of the inconsistent matrix.
(11) If no such CR is found, we conclude that the matrix cannot be resolved.

Algorithm 1 Selecting the best cell to take new input on

```
function FindCell(orignalMatrix, conf_int)  ▷ Where conf_int is the set of confidence intervals for all the cells in originalMatrix
    original_cr = calculateConsistency(originalMatrix)
    tempMatrix = originalMatrix
    crDict = { }                                                    ▷ crDict is a hash map
    for i = 1 to number_of_rows do
        for j = i to number_of_columns do
            left, right = conf_int[i][j][0], conf_int[i][j][1]
            tempMatrix[i][j] = left
            tempMatrix[j][i] = 1/left
            crDict['i_j_left'] = calculateConsistency(tempMatrix)
            tempMatrix[i][j] = right
            tempMatrix[j][i] = 1/right
            crDict['i_j_right'] = calculateConsistency(tempMatrix)
            tempMatrix = originalMatrix
        end for
    end for
    if minimum(crDict) < originalCR then
        return i,j
    end if
end function
```

	SL	SOTL	AB	LOTR
SL	1	0.1958	0.2911	4.8874
SOTL	5.1072	1	3.0878	6.9576
AB	3.4352	0.3238	1	4.806
LOTR	0.2046	0.1437	0.2081	1

Table 16. Example alternatives matrix after initial matrix creation phase

Original Consistency Ratio of the matrix = 0.10467. That means, the matrix is inconsistent and we have to resolve it. To do that we follow the Algorithm 1.

Cell	Value	Confidence Interval	i_j_left: CR	i_j_right: CR
1_2	-2.0532	[-3.1728, -0.9336]	1_2_left: 0.12232	1_2_right: 0.10561
1_3	-1.2175	[-1.933 , -0.5019]	1_3_left: 0.14556	1_3_right: 0.06666
1_4	1.9437	[0.8254, 3.062]	1_4_left: 0.04687	1_4_right: 0.16077
2_3	1.0439	[-0.0379, 2.1256]	2_3_left: 0.088	2_3_right: 0.15788
2_4	2.9788	[1.9093, 4.0482]	2_4_left: 0.14928	2_4_right: 0.08211
3_4	1.903	[1.5351, 2.2709]	3_4_left: 0.11047	3_4_right: 0.10203

Table 17. List of all the CRs when replacing each cell by lower and higher range of confidence interval

As we can see in Table 17, the least CR is given by 1_4_left, which is also smaller than the original consistency of the matrix. Hence, we take a new input for 1_4 or SL vs SOTL.

Creating new question to ask and getting new user input

Once we find out which cell, if changed, improves the consistency of the matrix the most, we won't change it directly of course. We ask for new input from the users for that cell. Since we have already taken the confidence interval of the sample for that cell while creating the matrix and when finding the ripeness of the cell into account, we hope the new user input to be in the range. Formulating the question is easy. Since each cell in the matrix signifies a comparison between a unique pair, we ask the users to compare that very pair.

Once we get a new input from a user, we again check for the inconsistency in the matrix. If the matrix is found to be inconsistent again, we try to resolve the matrix by following the same process explained above. We do this each time we receive a new input from the users. We stop as soon as the matrix turns near-consistent (CR < 0.1).

If the step 10 doesn't yield any such cell, we conclude the matrix cannot be resolved and thus return the same matrix (and its weight vector) for final ranks generation.

Preventing same question from being asked indefinitely

Yes, this is possible. Our algorithm may find the same cell to be the ripest for change multiple times. We may need to force the program to stop taking new inputs for the cell/comparison.

We use the concept of confidence intervals for this case too. Each time a new input is taken for a cell/comparison, the whole sample of the inputs for that comparison is examined. We calculate the confidence interval of the sample with certain confidence level. Then we test if the range of the confidence interval satisfies a certain threshold. This threshold is smaller than the one we used during creation of initial matrices. If the threshold is

met, the cell is archived to not ask for new inputs again. Keeping this threshold lower than the one during initial creation matrices gives us more assurance that it won't harm us to not take any more inputs for a cell.

3.7 Calculating priority weight vectors

Once we have the matrices ready for further processing, we need their weight vectors to calculate the final rankings. We have already discussed the process of calculating the weight vectors in section 2.3 (Basics of AHP). The matrices are treated the same way they are treated in normal AHP. We too use Eigenvector approach to calculate the priority weight vectors of all the matrices.

3.8 Calculating global rankings of the alternatives

The final ranking calculation is done the normal AHP way. The process is exactly the same as described in section 2.5.

4 SIMULATION

Although we have devised our algorithm to work with real time crowd data, we could not test it on real data because of time, cost, and feasibility constraints. For the testing purposes, we used simulated data.

Before simulating the data, we first manually prepared a reference data (similar to what is shown as example in section 2). We assigned comparison values (in AHP's scale) for all the pairs in the problem hierarchy. Then we converted those manually assigned values into our scale [-4, 4] and use those individual values as the means to generate samples of size 100 for respective pairs of comparisons. We then stored the generated data in a local database and designed the program and schema so that it retrieves the data from the database as it would from the real world users. We have generated the data with normal distribution. While generated the data, we used different standard deviations for testing purposes. Furthermore, we created both consistent and inconsistent matrices. We will observe the results in the coming section.

5 EXPERIMENTS

For testing, we created problem to rank 10 movies based on 5 criteria. We manually prepared all the required pairwise comparison matrices and used those individual values to generate samples of size 100 for each pair. We generated the samples using two different variations in the distribution:

- σ = 0.7 (small variance)
- σ = 1.5 (large variance)

We made a few sensible assumptions to calculate the minimal sample size (section 3.2). We take,
margin of error (d) = 2
$z_{\alpha/2}$ = 1.96 (for confidence level of 95%)
total estimated population (N) = 100000 (some large arbitrary value)
To calculate standard deviation, we use equation 7 with range = 8 getting σ = 2
Substituting all of these values in equation 6, we get minimal sample size (n) = 3.84 ≡ 4.

As we described earlier, we enforce thresholds in two places - 'Threshold 1' while taking initial inputs (section 3.3) and 'Threshold 2' while taking new inputs to resolve inconsistency in a matrix (section 3.6).
Both thresholds being absolute difference between mean of input sample and left range of confidence interval of the sample. The confidence level used is 90% for all thresholds and data.

	Threshold 1	Threshold 2
1	<= 1.5	<= 0.3
2	<= 1.5	<= 0.2
3	<= 1.2	<= 0.3
4	<= 1.2	<= 0.2
5	<= 1.0	<= 0.3
6	<= 1.0	<= 0.2
7	<= 0.8	<= 0.3
8	<= 0.8	<= 0.2

Table 18. Meaning of columns 1-8 in coming tables for data with small variation (σ=0.7)

	Threshold 1	Threshold 2
1	<= 1.5	<= 0.4
2	<= 1.5	<= 0.3
3	<= 1.2	<= 0.4
4	<= 1.2	<= 0.3
5	<= 1.0	<= 0.4
6	<= 1.0	<= 0.3
7	<= 0.8	<= 0.4
8	<= 0.8	<= 0.3

Table 19. Meaning of columns 1-8 in coming tables for data with large variation (σ=1.5)

Threshold_1_Threshold_2								
	1.5_0.3	1.5_0.2	1.2_0.3	1.2_0.2	1.0_0.3	1.0_0.2	0.8_0.3	0.8_0.2
Criteria matrix	69	69	192	192	531	531	813	813
Alternative matrix 1	251	251	704	704	1846	1846	3734	3734
Alternative matrix 2	244	244	834	834	1919	1919	3067	3225
Alternative matrix 3	456	938	630	1157	1760	2179	3130	3314
Alternative matrix 4	495	771	1005	1394	2228	2225	3510	3534
Alternative matrix 5	526	1172	898	1472	1811	2247	2609	2944
Total	2041	3445	4263	5753	10095	10947	16863	17564

Table 20. Number of inputs required for each matrix with different threshold combinations shown in Table 18 (data with small variation)

Threshold_1_Threshold_2								
	1.5_0.4	1.5_0.3	1.2_0.4	1.2_0.3	1.0_0.4	1.0_0.3	0.8_0.4	0.8_0.3
Criteria matrix	809	809	809	809	809	809	809	809
Alternative matrix 1	3743	3743	4116	4116	4212	4212	4404	4404
Alternative matrix 2	3787	3889	4193	4242	4193	4242	4410	4410
Alternative matrix 3	3693	3805	3792	3818	4129	4134	4415	4420
Alternative matrix 4	3976	3972	4223	4223	4220	4220	4317	4317
Alternative matrix 5	3365	3365	4116	4116	4117	4117	4281	4364
Total	19373	19583	21249	21324	21680	21734	22636	22724

Table 21. Number of inputs required for each matrix with different threshold combinations shown in Table 19 (data with large variation)

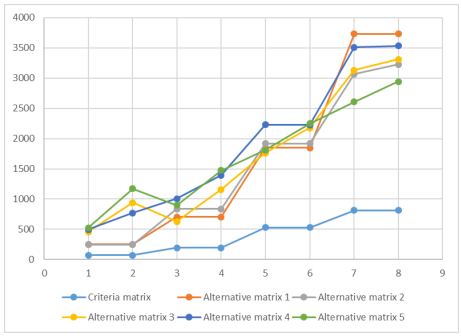

Fig. 4. Number of inputs taken to create each matrix until final rank calculation with different threshold combinations from table 18 (data with small variation)

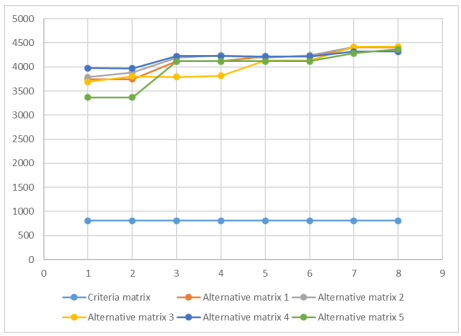

Fig. 5. Number of inputs taken to create each matrix until final rank calculation with different threshold combinations from table 19 (data with large variation)

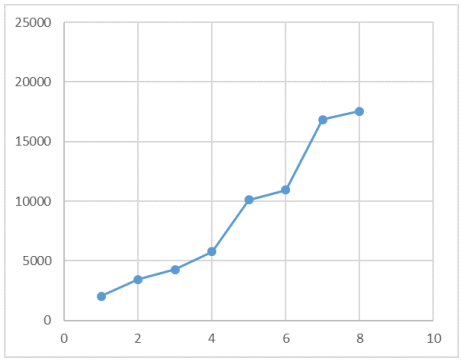

Fig. 6. Total combined number of inputs taken until final rank calculation with different threshold combinations from table 18 (data with small variation)

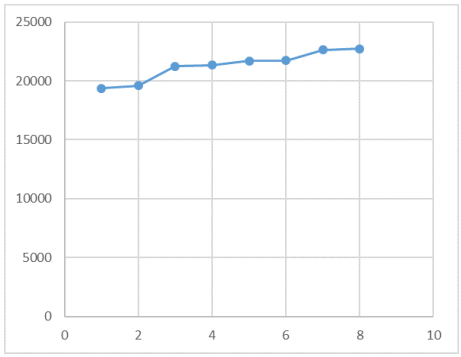

Fig. 7. Total combined number of inputs taken until final rank calculation with different threshold combinations from table 19 (data with large variation)

Threshold_1_Threshold_2								
	1.5_0.3	1.5_0.2	1.2_0.3	1.2_0.2	1.0_0.3	1.0_0.2	0.8_0.3	0.8_0.2
Criteria matrix	0.09742	0.09742	0.09825	0.09825	0.09972	0.09972	0.09716	0.09716
Alternative matrix 1	0.09604	0.09604	0.09398	0.09398	0.09007	0.09007	0.08669	0.08669
Alternative matrix 2	0.094	0.094	0.09205	0.09205	0.08797	0.08797	0.1034	0.10551
Alternative matrix 3	0.11186	0.1114	0.11543	0.12169	0.12208	0.12792	0.12319	0.12602
Alternative matrix 4	0.11514	0.09991	0.11537	0.10138	0.10728	0.09987	0.10809	0.09995
Alternative matrix 5	0.14414	0.11686	0.14559	0.11687	0.14646	0.11868	0.1434	0.11764

Table 22. Final Consistency Ratios (CR) of all the comparison matrices with different threshold combinations from table 18 (data with small variation)

Threshold_1_Threshold_2								
	1.5_0.4	1.5_0.3	1.2_0.4	1.2_0.3	1.0_0.4	1.0_0.3	0.8_0.4	0.8_0.3
Criteria matrix	0.09512	0.09512	0.09512	0.09512	0.09512	0.09512	0.09512	0.09512
Alternative matrix 1	0.09786	0.09786	0.09232	0.09232	0.09173	0.09173	0.09215	0.09215
Alternative matrix 2	0.11031	0.10923	0.11015	0.11086	0.11015	0.11086	0.11121	0.11121
Alternative matrix 3	0.11994	0.11487	0.12051	0.1185	0.12071	0.11868	0.12507	0.12314
Alternative matrix 4	0.10198	0.09972	0.09979	0.09979	0.09979	0.09979	0.09964	0.09964
Alternative matrix 5	0.09719	0.09719	0.09086	0.09086	0.0943	0.0943	0.10949	0.10872

Table 23. Final Consistency Ratios (CR) of all the comparison matrices with different threshold combinations from table 19 (data with large variation)

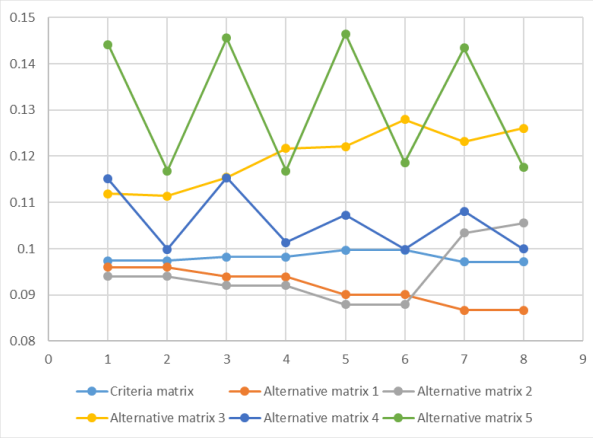

Fig. 8. Final Consistency Ratios (CR) of all the matrices with different threshold combinations from table 18 (data with small variation))

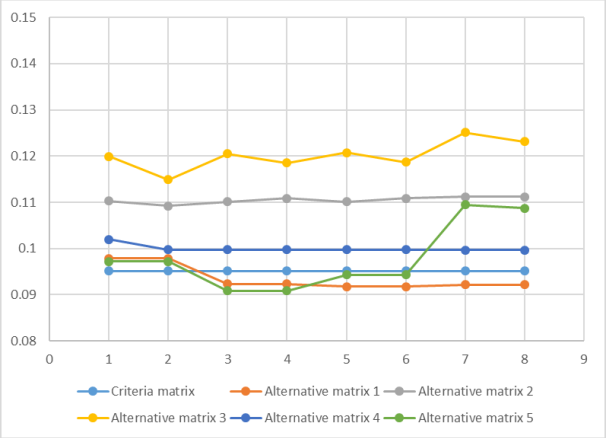

Fig. 9. Final Consistency Ratios (CR) of all the matrices with different threshold combinations from table 19 (data with large variation)

Threshold_1_Threshold_2								
	1.5_0.3	1.5_0.2	1.2_0.3	1.2_0.2	1.0_0.3	1.0_0.2	0.8_0.3	0.8_0.2
LOTR	0.2072	0.2072	0.2022	0.2025	0.1993	0.1995	0.2027	0.2022
TDK	0.1745	0.1756	0.1727	0.1739	0.1737	0.1745	0.1667	0.1719
SL	0.1612	0.1611	0.1694	0.1693	0.1664	0.1662	0.1652	0.1647
LL	0.128	0.1278	0.1281	0.1283	0.1319	0.1322	0.1327	0.1312
HR	0.1003	0.1011	0.0983	0.0982	0.0998	0.0999	0.1007	0.1001
SP	0.0834	0.0825	0.0831	0.0823	0.0832	0.0825	0.0841	0.0833
MM	0.0606	0.0607	0.0611	0.0611	0.0604	0.0607	0.0594	0.0587
JW	0.0366	0.0361	0.0372	0.0368	0.0375	0.0373	0.0397	0.0396
ESN	0.0363	0.0361	0.0361	0.0357	0.0358	0.0354	0.0371	0.0364
TR	0.0119	0.0119	0.0119	0.0119	0.0119	0.0119	0.0118	0.0118

Table 24. Final Rankings of all the alternatives with different threshold combinations from table 18 (data with small variation)

Threshold_1_Threshold_2								
	1.5_0.4	1.5_0.3	1.2_0.4	1.2_0.3	1.0_0.4	1.0_0.3	0.8_0.4	0.8_0.3
LOTR	0.203	0.2017	0.2033	0.2027	0.2034	0.2028	0.2021	0.2021
TDK	0.172	0.172	0.1755	0.1753	0.1745	0.1744	0.174	0.174
SL	0.1662	0.1661	0.1617	0.1617	0.1619	0.1619	0.1615	0.1615
LL	0.1234	0.1246	0.1242	0.1241	0.1248	0.1247	0.124	0.124
HR	0.1102	0.1101	0.1094	0.1104	0.1096	0.1106	0.1108	0.1108
SP	0.0834	0.0835	0.0832	0.0832	0.0828	0.0829	0.0846	0.0846
MM	0.0562	0.0562	0.0563	0.0563	0.0566	0.0566	0.0566	0.0566
JW	0.0392	0.0392	0.0395	0.0395	0.0396	0.0395	0.0394	0.0392
ESN	0.0348	0.0348	0.035	0.035	0.0351	0.0351	0.0354	0.0354
TR	0.0117	0.0117	0.0117	0.0117	0.0117	0.0117	0.0117	0.0118

Table 25. Final Rankings of all the alternatives with different threshold combinations from table 19 (data with large variation)

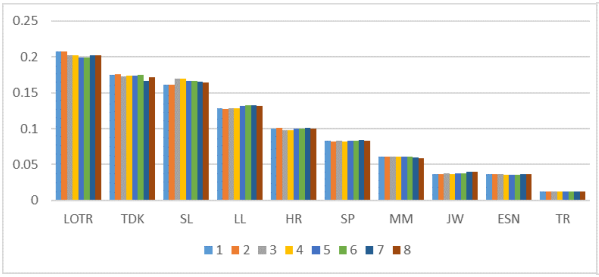

Fig. 10. Final rankings of all alternatives with different threshold combinations from table 18 (data with small variation)

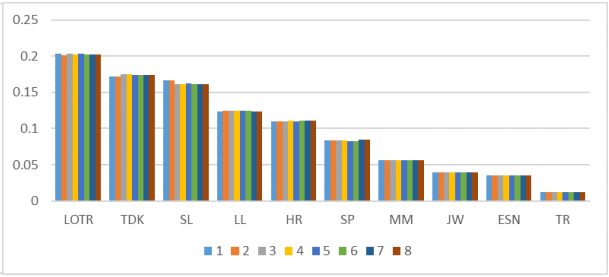

Fig. 11. Final rankings of all alternatives with different threshold combinations from table 19 (data with large variation)

6 RESULTS

By analyzing above tables and graphs, we can make following interpretations:

- If the underlying data has high variation, then the thresholds we implement do not have much effect on the total inputs taken from the users for each comparison matrix.
- Smaller the variation in the underlying data, higher are the chances of thresholds having more impact on the number of inputs taken
- For data with small underlying variation, the number of inputs taken from the users generally increases as the threshold becomes strict.
- There seems to be not much difference in final consistency ratio of matrices despite any thresholds, if the data used has large underlying variation.
- For data with smaller variation, the changes in the final consistency ratios are a bit unpredictable.
- No matter the variation in the data or the thresholds enforced, the final rankings don't seem to care. We saw no change in the actual final rankings, and little to negligible changes in the final global weights of the alternatives.

7 CONCLUSION

In this work, we attempted to use AHP's decision making model in crowd sourcing environment instead of the closed small group of people making each comparison together. As we have seen in this paper, multiple users are allowed to participate and contribute in the decision making process. The application supports any ranking problems as long as the hierarchy (goal, set of criteria, set of alternatives) has been properly set. Although, we tested and showed the results using simulated data, we expect this technique to work in the real world too. The inputs taken from the users until the generation of final rankings seem reasonable depending upon how strict we want the thresholds to be. There might be some threshold that could be the optimal one for most of the problems out there. We realize this work needs refinement and thorough testing on real world environment. Our work is mostly helpful for certain kinds of problems where uncertainty is not a big issue and where huge user participation is needed. This model could be a fit for problems where exact measurements are not important but the final rankings are. In this age of social networking and abundance of mobile devices users in the world, we see a lot of scope for our technique.

ACKNOWLEDGMENTS

I would like to thank Dr. Chengkai Li for his invaluable guidance, suggestions, and supervision. I am also grateful to the whole IDIR Lab and its members for their feedbacks, help and encouragement along the way.

REFERENCES

[1] Markus Lusti Alessio Ishizaka. How to derive priorities in AHP: a comparative study. (????).
[2] Matteo Brunelli. 2015. *Introduction to the Analytic Hierarchy Process.* Cambridge University Press. DOI: https://doi.org/10.1007/978-3-319-12502-2
[3] Williams C. Crawford G. 1985. A Note on the Analysis of Subjective Judgement Matrices. *Journal of Mathematical Psychology* 29, 387-405 (1985).
[4] Araz Taeihagh John Prpic and James Melton. 2015. The Fundamentals of Policy Crowdsourcing. *Policy and Internet* 7, 3 (2015). DOI: https://doi.org/10.1002/poi3.102
[5] Wang T.Y. Johnson C.R., Beine W.B. 1979. Right-left asymmetry in an eigenvector ranking procedure. *Journal of Mathematical Psychology* 18, 61-64 (1979).
[6] Markus Lusti. 2002. *Data Warehousing und Data Mining.* Vol. 2. Springer-Verlag, Berlin.
[7] C.R. Johnson R.A. Horn. 1985. *Matrix Analysis.* Cambridge University Press.
[8] R. Lyman Ott Richard L. Scheaffer, William Mendenhall III. 2012. *Elementary Survey Sampling.* Number 76-117. Thompson Books/Cole, Berlin. DOI: https://doi.org/0-534-41805-8

[9] Thomas L. Saaty. 1977. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology* 15, 234âĂŞ281 (1977).
[10] Thomas L. Saaty. 1984. Comparison of Eigenvalue, Logarithmic Least Squares and Least Squares Methods in Estimating Ratios. *Mathematical Modelling* 5, 309-324 (1984).
[11] Thomas L. Saaty. 2003. Decision-making with the AHP: Why is the Principal Eigenvector necessary? *European Journal of Operational Research* 145, 85-91 (2003).
[12] L.Saaty Thomas. 1980. *The Analytical Hierarchy Process.* Tata McGraw Hill, New York.