Diss/xx/32/195

M. Tech. (Computer Science) Dissertation Series

HURISTIC SEARCH TECHNIQUE FOR CLUSTERING

A dissertation submitted in partial fulfillment of the requirements for the M. Tech. (Computer Science) degree of the Indian Statistical Institute

By

Amitava Roy

Under the supervision of

Dr. K.S. Ray

INDIAN STATISTICAL INSTITUTE
203, B.T. Road, Cal -108

A C K N O W L E D G E M E N T

I would like to express my greatfulness to my guide Dr. K.S. Ray who not only provided me with various papers and other study materials extremely necessary for my dissertation but also helped me by continuous encouragement.

I would also like to thank Shri Tapas Das for typing my dissertation report and Shri Dilip Chatterjee for cyclostyling the same.

Abstract: The statistical problem we are concerned with is the well known clustering problem viz. to partition a set of n objects into m nonempty disjoint subsets called clusters. This clustering or grouping is done in such a manner that within clusters a certain criterion of homogeneity and between clusters a certain criterion of heterogeneity is maintained.

Though total enumeration of all possible clustering necessarily outputs a global optimum, for large n and m it is impossible to stick to this method. So, instead of total enumeration, the dynamic programming method for getting optimum clustering is examined which shows a considerable reduction in the number of inherent calculations. Later on we develop a heuristic function to make use of the method called dynamic programming method with reducing heuristic. The heuristic is developed in such a way that to solve the problem for small m and n, the amount of calculation taken is almost the same but for problems with larger dimensions (i.e. larger m and n) the reduction in intermediate calculation is spactecularly well.

Introduction:

The technique of partitioning n objects into m non-empty subsets or clusters, commonly khown in the literature as cluster analysis encompasses many situations in scientific and business investigation. We state the problem formally as follows.

Let the set $I = I_1, \ldots\ldots, I_n$ denote a set of n individuals from a conceptual population π_1. It is tacitly assumed that there exists a set of features or characteristic $c = (c_1, \ldots\ldots, c_p)'$ which are observable and are possessed by each individual in I. The term cobservable is used here to denote characteristics that yield both quantitative and qualitative data, although we will base most of the following discussions on quantitative data which we call measurements.

We denote the value of the measurement of the i-th characteristic of the individual I_j by x_{ij} and let $x_j = (x_{1j}, x_{2j}, \ldots\ldots x_{pj})'$ denote pxl vectpr of measurements of individual I_j.

Hence for the set of individuals I, there is available to the investigator a corresponding set $x = x_1, x_2, \ldots\ldots x_p$

of measurement vectors which in a certain manner describes the set I.

The Cluster Problem

Let m and n be natural numbers with m less than n. The cluster problem is to determine m clusters of individuals in I, say $\pi_1, \pi_2, \ldots, \pi_m$, such that I_j belongs to one and only one subset and individuals in the same cluster are similar in certain manner while individuals from different clusters are different in certain manner. These being determined on the basis of the observed set x.

A solution to the cluster problem is to partition n individuals into m subsets satisfying some optimality criterion. This may be determined in such a manner so as to reflect the level of desirability of the various partitioning.

As an example, suppose p = 1 characteristic is measured on each of n = 8 individuals resulting in the set x = 3,4,7,4,3,3,4,4 . The Within Group Sum of Squares or WGSS is given by

$$W = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2$$

where x_i is the measurement of the i th individual.

If we construct a single group of all the 8 persons, the WGSS becomes $\sum_{i=1}^{8} x_i^2 - \frac{1}{8}\left(\sum_{i=1}^{8}\right)^2 = 140 - 128 = 12$

If, instead, we partition then in 3 groups viz. $G_1 =$ 3,3,3 , $G_2 =$ 4,4,4,4 and $G_3 =$ 7 then the required WGSS is $W_1 + W_2 + W_3 = 0 + 0 + 0 = 0$, where W_i is the sum of squares for G_i. The optimal value, in this example is 0, if one desires three groups. In general one must consider both, the values of the objective function as well as no. of clusters desired.

By this time, it should be clear that to tackle this sort of clustering problem, one should clearly define the term s similarities and difference in a quantitative fashion. It is important to note here that each individual from P can be though as a point in the p-dimensional plane, E_p.

Distance Functions

Definition: A non-negative real-valued function $d(x_i, x_j)$ is said to be a distance function (metric) if

(i) $d(x_i, x_j) > 0$ for all x_i and x_j in E_p,

(ii) $d(x_i, x_j) = 0$ iff $x_i = x_j$,

(iii) $d(x_i, x_j) = d(x_j, x_i)$ and

(iv) $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$

where x_i, x_j, x_k are any three vectors in E_p. The value of $d(x_i, x_j)$ for specified x_i and x_j is said to be the distance between I_i and I_j with respect to the selected characteristcs $c = (c_1, \ldots\ldots, c_p)^T$. Some popular distance functions are given below.

Name	Form
1. Euclidean	$d_2(x_i, x_j) = \left[\sum_{k=1}^{p} (x_{ki}-x_{kj})^2\right]^{\frac{1}{2}}$
2. L_1 norm	$d_1(x_i,x_j) = \left[\sum_{k=1}^{p} \lvert x_{ki} - x_{kj}\rvert\right]$
3. Sup-norm	$d(x_i,x_j) = \sup_{1 \leq k \leq p} \lvert x_{ki} - xkj\rvert$
4. lp, norm	$d_p(x_i,x_j) = \left[\sum_{k=1}^{p} \lvert x_{ki} - x_{kj}\rvert^p\right]^{1/p}$
5. Mahalanobis	$D^2(x_i,x_j) = (x_i-x_j)^T W^{-1}(x_i-x_j)$

Among the above distance functions, the Euclidean metric is possibly the most popular and very commonly uses one. Unless stated to the contrary, we will stick to this function in subsequent discussion.

Clustering by complete Enumeration

The principle is straight forward. First the all possible way of partitioning n objects into m clusters are determined. All possible such partitioning is given by the Stirling's number of second kind viz.

$$S(n, m) = \frac{1}{m!} \sum_{k=0}^{m} (-1)^{m-k} \binom{m}{k} k^n$$

where s (n, m) denotes the total number of such partitions.

Then we consider that clustering alternative which gives the minimum WGSS

Suppose there are n_k individuals in the cluster G_k of a given alternative for partitiong n entities into m clusters. In subsequent discussion, it will be assumed that

$$W = \sum_{k=1}^{m} T(g_k)$$

is the criterion measure of clustering homogeneity, where

$$T(G_k) = \frac{1}{n_k} \sum_{\substack{i<j \\ \text{in } G_k}} d_{ij}^2$$ is the transition cost of

cluster k.

Inefficiency of Total Enumeration

Though the underlying principle looks simple enough, in most cases, total enumeration of all feasible clustering alternative for the optimal solution is out of the question, even with the largest as well as fastest electronic computers at hand. Interestingly, the difficulty is usually one of computing speed, rather than rapid access storage. The number of feasible clustering alternative is astronomical for even moderate m and n.

As for example, for n = 25, m = 10

S (25, 10) = 1,203, 163,392,175,387,500

This inefficiency of total enumeration has led the evolution of several other practicable clustering methods which are computationally more efficient. But unlike total enumeration method, they merely search for the best solution among a small subset of clustering alternatives.

Dynamic Programming Approach

The dynamic programming algorithm (enunciated subsequently) ensures convergence to the global optimal solution using W minimization criterion, without having to enumerate all clustering alternatives.

From a computational stand point the algorithm eliminates many reduntant computations implicit in the total enumeration method.

Inefficiency of Dynamic Programming

Though it eliminates reduntant calculations considerably, unfortunately it also requires additional rapid access storage and like methods, may not be practical in very large problems because of added search time in auxiliary storage.

However for problems for certain dimensions, the dynamic programming method suggested here may be practical and yield substantial computer savings.

The Set-up of Dynamic Programming

The recursive formula for the dynamic programming formulation may be written as

$$W_k^*(G) = \begin{cases} 0 \text{ for } k = 0 \\ \min_y [\, T(G-y) + W_{k-1}(y) \,] \text{ for } k = 1, \ldots, m_o \end{cases} \quad \ldots(*)$$

m = number of disjoint and non-empty subsets into which n entities are to be partitioned.

k = index of stage variable

$m_o = m$ if $n \geq 2m$ and $n - m$ if $n < 2m$

G = state variable representing a given set of entities at stage k

y = state variable representing a given set of entities at stage $k - 1$

$G-y$ = set of all elements in G which are not in y.

$T(G-y) = \frac{1}{n_k} \sum_{\substack{i<j \\ \text{in} \\ (G-y)}} d_{ij}^2$, where n_k is the number of entities in $G-y$.

Values of $W_k^*(G)$ as defined in (*) represent the minimum W criterion value for the optimum way to partition entities

contained in G into k non-empty and mutually exclusive subsets.

Total Feasible Arcs and States in a Network

All the S(n, m) clustering alternatives can be classified according to various distribution forms of m clusters. For example, for n = 4, m = 2, the distribution forms are

3 , 1 and 2 , 2 . Distribution forms essentially stands for the size of the clusters. In the example above, the clusters corresponding to the distribution form 3 1 are given by.

(1, 2, 3), (4)
(1, 2, 4), (3)
(1, 3, 4), (2)
(2, 3, 4), (1)

For comtational convenience, the distribution forms are written in descending order. The number of distribution forms, in general is substantially smaller than the number of clusters itself.

The clustering alternatives are first classified according to their distribution forms.

At the first stage, the objective function for each cluster corresponding to the first distribution form component is evaluated and saved. At the second stage the

objective function for the clusters corresponding to the first 2 components of the distribution forms is evaluated using all information from the first stage, i.e. the WGSS is not recomputed for any cluster but carried over from the first stage.

We will subsequently work out a numerical example after developing relevant theories.

Theory:

The maximum number of entities max (k), at stage k is equal to the maximum sum of the distribution form components from stage 1 to stage k inclusively.

$$\therefore \quad \max(k) = n - m + k$$

similarly, if n is an even multiple of m, we have min (k) $= k \cdot [\frac{n}{m}]$.

otherwise, when n is not a even multiple m,

$$\min(k) = \begin{cases} ([\frac{n}{m}]+1)k & \text{for } 1 \leq k \leq n - m[\frac{n}{m}] \\ n - (m-k)[\frac{n}{m}] & \text{for } n - m[\frac{n}{m}] < k \leq m \end{cases}$$

where $[\frac{n}{m}]$ denotes the integer part of $\frac{n}{m}$.

At stage k of the dynamic programing process, the number of states that can be formulated is

$$NS(k) = \begin{cases} 1 & \text{for } k = o \text{ (one dummy state is assumed to exist)} \\ \sum_{l=\min(k)}^{\max(k)} (n)_l & \text{for } k = 1, \ldots, m_o \end{cases}$$

Between successive stages, states are connected by arcs. As a necessary condition, feasible arcs cannot exist between a state in stage k and that in stage k+1 if the former is not the subset of the later.

Let TFA denote the total number of feasible arcs in the entire network, where

$$TFA = NS(1) + \sum_{k=o}^{m_o - 1} TA(k)$$

There are NS(1) arcs connecting the dummy origin with the NS(1) states in stage I.

The value TA (k), representing the total number of feasible arcs between stage k and stage k + 1, is given for $k = 1, \ldots, m_o$ by

$$TA(k) = \sum_{l=\min(k)}^{\max(k)} \sum_{j=1}^{\max(k+1) - \min(k+1)} FA(l,j)$$

where $FA(l,j) = \begin{cases} (n)_l (n-l)_j, & \text{if } \min(k+1) \leq l+j \leq \max(k+1) \\ 0 & \text{otherwise} \end{cases}$

note that m_o, as defined earlier is the final stage of the dynamic programming problem. $m_o = m$ could always be possible. However in simpler problems where $n < 2m$, there must always be $n - m$ clusters with more than one entity. For the remaining single-entity clusters, the WGSS is zero which does not contribute effectively to the total clustering sum of squares.

Reduced Network Formulation

Although the dynamic programming formulation given .. can be recursively applied to a network of TFA feasible arcs, that must be evaluated because various symmetries create redundancies.

We now indicate how redundant feasible arcs can be eliminated quite simply. Let NAT represent the number of arcs remaining after such elimination i.e. NAT should be viewed as the maximum number of feasible arcs required to solve the problem). For our purposes, let NAT =

$$NS(1) + \sum_{k=o}^{m_o - 1} NA(k)$$

The value NA (k), representing the number of arcs between stage k and stage k + 1, is given by

$$NA(k) = \sum_{l=\min(k)}^{\max(k)} \sum_{j=1}^{\max(k+1)-\min(k+1)} A(l,j)$$

$$\text{where } A(l,j) = \begin{cases} \binom{n}{l}\binom{n-l}{j} & \text{if } l \neq j \\ \frac{1}{2}\binom{n}{l}\binom{n-l}{j} & \text{if } l = j \end{cases}$$

and $\min(k+1) < l + j$

$\text{and} < \max(k+1),$

$(m-k)\, j + l > n$

We proceed by way of illustration. Suppose N = 7 entities are to be partitioned into M = 3 clusters. There are 4 possible distribution forms as given below:

5	1	1
4	2	1
3	3	1
3	2	2

In this case $N = 7 > 2.M = 6$ so $M_o = M = 3$

max (1) = 5 max (2) = 6, max (3) = 7

min (1) = 3 min (2) = 5, min (3) = 7

NS (1) = 91, NS (2) = 28, NS(3) = 1

TA (1) = 602, TA (2) = 28,

NA (1) = 427, NA (2) = 28

Because there are 7 entities to be grouped by taking 3, 4, 5 at a time, the total number of states in stage 1 is $\binom{7}{3} + \binom{7}{4} + \binom{7}{5} = 91$

And for k = 1, we have the following feasible arcs:

FA (3, 1) = o, FA (3, 2) = $\binom{7}{3}\binom{4}{2}$ = 210

FA (3, 3) = $\binom{7}{3}\binom{4}{3}$ = 140 FA (4,1) = $\binom{7}{4}\binom{4}{1}$ = 105

FA(4,3) = 0, FA (5,1) = $\binom{7}{5}\binom{2}{1}$ = 42 etc.etc.

As an improvement and faster method, the heuristic dynamic programming search is introduced below.

Heuristic Search

In order to solve many hard problems efficiently, often it is necessary to compromise the requirements of mobility and systemicity and to construct a control structure that is no longer guaranteed to find the best answer but that will almost always find a very good solution. A heuristic is a technique that improves the efficiency of a search process, possibly by sacrificing the claims of completeness. Using heuristics, we can hope to get good (even if non-optimal) solutions to hard problems, such as travelling salesman, in less than exponential time.

Heuristic Search in Dynamic Programming

When we were talking about stage-wise solution of the clustering method, with most emphasise on arcs from one level to that immediately higher, we can, as well have viewed the solution method, that of a graph search, the nodes of the graph being the states of the different stages and arcs are that between successive stages.

That graph-search comes dynamic programming method necessarily converge to the globally optimal solution. Now to improve that method, we propose the introduction of heuristic function.

The set up is as follows.

Let the evaluation function f(n) at any node n estimates the sum of the cost of the minimal cost path from the start node s to node n plus the cost of a minimal cost path from node n to h (in this specific set up, to the) goal node.

i.e. $f(n) = g(n) + h(n)$

where

g(n) is the first cost component which is actually calculated, where as h(n) is an estimate of the 2nd cost component. If h*(n) is the actual value of the 2-nd cost component then we call h(n) is an heuristic function corresponding to h*(n).

Result: If $h(n) \leq h^*(n)$ for all nodes n, then this heuristic dynamic programming search (supposed to be ad-hoc) guarantees the convergence to the globally optimal solution. Basing on the above result, we choose our heuristic to exploit and guarantee the above criterion of optimality. Specifically, h is choosen asfollows.

If n is a node in k-th stage then

$h(n) =$ min T(m), minimum being taken over all nodes m in stage (k+1) s.t. there isan arc between n and m.

Where, we should recall, for a node (set) g_k at stage k,

$$T(g_k) = \sum_{\substack{i<j \\ \text{in } g_k}} d_{ij}^2$$

It is sufficient to prove that h is a reducing heuristic i.e. $h \leq h^*$.

Pf: The optimal path from node n (at stage k) to the goal node must pass along one of the arcs from n to the immediately next stage node m (say) since we are choosing minimum cost among these arcs and there will always be (at least) non-negative cost component from node m to the goal node, we will always have $h(n) \leq h^*(n)$ for all n. So, by the result stated earlier, we will indeed converge to th global optimal solution by this heuristic function.

Discussion and Results

We will now see through numerical examples how the dynamic programming and its subsequent heuristic improvement finds an optimal solution in an workable fashion.

Let n = 5, m = 3. Let the elements to be clustered be names as 1,2,3,4,5 The arcs and the associated redundant arcs are exhibited later.

Here n $\neq$ 2m so $m_o = n-m = 2$

max (1) = 5-3+1 = 3, max (2) = 5-3+2 = 4

min (1) = $\left(\frac{5}{3} + 1\right) \cdot 1 = 2$, min(2) = $\left(\frac{5}{3} + 1\right) \cdot 2 = 4$

NS(o) = 1, NS(1) = $\binom{5}{2} + \binom{5}{3} = 20$, NS(2) = $\binom{5}{4} = 5$

No. of arcs from stage 0 to stage 1 = NS(1) = 20

No. of arcs from stage 1 to stage 2 is

A (2,2)= $\frac{1}{2}$ $\binom{5}{2}$ $\binom{3}{2} = 15$ and A (3,1) = $\binom{5}{3}$ $\binom{2}{1} = 20$

i.e. total NA (1) = 20 + 15 = 35

So the total no. of arcs in the networkm is given by 35+20 =

Let p=2 or 2 characteristics of the observation is

(1 3 5 4 1)

(1 4 5 4 2)

$d_{12}^2 = 13$, $d_{13}^2 = 32$, $d_{14}^2 = 18$, $d_{15}^2 = 1$,

$d_{23}^2 = 5$, $d_{24}^2 = 1$, $d_{25}^2 = 8$, $d_{34}^2 = 2$, $d_{35}^2 = 25$,

$d_{45}^2 = 13$

Now we compute the relevant transition costs as shown in table 1.

Stage o. $W^*(o) = o$

Stage 1. In table II.

Stage 2. In table III.

At stage 2, the process is terminated. At this point, the minimum $W^*(2)$ value is found to be 1 corresponding to the optimal clustering policy of (1,5), (2,4) and (3)

Comments In a very small problem as this, dynamic programming (with or without heuristic) is no more efficient than total enumeration. But as the problem increases in dimension for larger values of m and n, it will be clearer that these 2 methods suggested here are far stronger.

TABLE I

Transition Costs

$T(1) = o$	$T(1,2) = \frac{1}{2} d_{12}^2 = 6.5$
$T(2) = 2$	$T(1,3) = \frac{1}{2} d_{13}^2 = 16.o$

$T(3) = 0$ $\quad T(1,4) = \frac{1}{2} d_{14}^2 = 9.0$

$T(4) = 0$ $\quad T(1,5) = \frac{1}{2} d_{15}^2 = 0.5$

$T(5) = 0$ $\quad T(2,3) = \frac{1}{2} d_{23}^2 = 2.5$

$T(2,4) = \frac{1}{2} d_{24}^2 = 0.5$

$T(2,5) = \frac{1}{2} d_{25}^2 = 4.0$

$T(3,4) = \frac{1}{2} d_{34}^2 = 1.0$

$T(3,5) = \frac{1}{2} d_{35}^2 = 12.5$

$T(4,5) = \frac{1}{2} d_{45}^2 = 6.5$

$T(1,2,3) = \frac{1}{3} (d_{12}^2 + d_{13}^2 + d_{23}^2) = 16.67$

$T(1,2,4) = \frac{1}{3} (d_{12}^2 + d_{14}^2 + d_{24}^2) = 10.67$

$T(1,2,5) = \frac{1}{3} (d_{12}^2 + d_{15}^2 + d_{25}^2) = 7.33$

$T(1,3,4) = \frac{1}{3} (d_{13}^2 + d_{14}^2 + d_{34}^2) = 17.33$

$T(1,3,5) = \frac{1}{3} (d_{13}^2 + d_{15}^2 + d_{35}^2) = 13.00$

$T(1,4,5) = \frac{1}{3} (d_{14}^2 + d_{15}^2 + d_{45}^2) = 10.67$

$T\ (2,3,4) = \frac{1}{3}\left(d_{23}^2 + d_{24}^2 + d_{34}^2 \right) = 2.67$

$T\ (2,3,5) = \frac{1}{3}\left(d_{23}^2 + d_{25}^2 + d_{35}^2 \right) = 32.67$

$T\ (2,4,5) = \frac{1}{3}\left(d_{24}^2 + d_{25}^2 + d_{45}^2 \right) = 7.33$

$T\ (3,4,5) = \frac{1}{3}\left(d_{34}^2 + d_{35}^2 + d_{45}^2 \right) = 13.33$

TABLE II

Values for Stage I

$W_1^*\ (1,2,3)$	= 16.67
$W_1^*\ (1,2,4)$	= 10.67
$W_1^*\ (1,2,5)$	= 7.33
$W_1^*\ (1,3,4)$	= 17.33
$W_1^*\ (1,3,5)$	= 13.00
$W_1^*\ (1,4,5)$	= 10.67
$W_1^*\ (2,3,4)$	= 2.67
$W_1^*\ (2,3,5)$	= 32.67
$W_1^*\ (2,4,5)$	= 7.33
$W_1^*\ (3,4,5)$	= 13.33

W_1^* (1,2) = 6.5

W_1^* (1,3) = 16.0

W_1^* (1,4) = 9.0

W_1^* (1,5) = 0.5

W_1^* (2,3) = 2.5

W_1^* (2,4) = 0.5

W_1^* (2,5) = 4.0

W_1^* (3,4) = 1.0

W_1^* (3,5) = 12.5

W_1^* (4,5) = 6.5

TABLE III

Values for Stage 2

$W_2^*(1,2,3,4) = \min\ T(4) + W_1^*(1,2,3),\ T(3) + W_1^*(1,2,4),\ T(2) + W_1^*(1,3,4),\ T(1) + W_1^*(2,3,4),\ T(2,3) + W_1^*(1,4),\ T(2,4) + W_1^*(1,3),\ T(3,4) + W_1^*(1,2) = T(1) + W_1^*(1,3,4) = 2.67$

$W_2^*(1,2,3,5) = \min\ T(5) + W_1^*(1,2,3),\ T(3) + W_1^*(1,2,5),\ T(2) + W_1^*(1,3,5),\ T(1) + W_1^*(2,3,5),\ T(1,3) + W_1^*(2,5),\ T(2,3) + W_1^*(1,5),\ T(3,5) + W_1^*(1,2) = T(2,3) + W_1^*(1,5) = 3.00$

$W_2^*(1,2,4,5) = \min\ T(5) + W_1^*(1,2,4),\ T(4) + W_1^*(1,2,5),\ T(2) + W_1^*(1,4,5),\ T(1) + W_1^*(2,4,5),\ T(1,2) W_1^*(4,5),\ T(2,4) + W_1^*(1,5),\ T(2,5) + W_1^*(1,4) = T(2,4) + W_1^*(1,5) = 1.00$

$W_2^*(1,3,4,5) = \min\ T(5) + W_1^*(1,3,4),\ T(4) + W_1^*(1,3,5),\ T(3) + W_1^*(1,4,5),\ T(1) + W_1^*(3,4,5),\ T(1,4) + W_1^*(3,5),\ T(4,5) + W_1^*(1,3),\ T(1,5) + W_1^*(3,4) = T(1,5) + W_1^*(3,4) = 1.50$

$W_2^*(2,3,4,5) = \min\ T(5) + W_1^*(2,3,4),\ T(4) + W_1^*(2,3,5),\ T(2) + W_1^*(3,4,5),\ T(3) + W_1^*(2,4,5),\ T(3,4) + W_1^*(2,5),\ T(3,5) + W_1^*(2,4),\ T(4,5) + W_1^*(2,3) = T(5) + W_1^*(2,3,4) = 2.67$

Now the above calculations reveal that minimum is attained (in table III) is $W_2^*(1,2,4,5) = T(2,4) + W_1^*(1,5) = 1.00$

Hence the optimum clustering is 1,5 , 2, 4 and 3.

Heuristic Search for the same problem

We would often refer to the tables I and II of dynamic programming solution.

For the start node, g (n) = 0 (the actual cost). By the heuristic method described earlier, h (start node) = min over all states from stage 1 which is (1,5) or (2,4) because for both of them h (n) = T (n) = 0.50 (note nodes are states or set of clustering alternatives). We break the ties arbitrarily i.e. say in favour of (1,5).

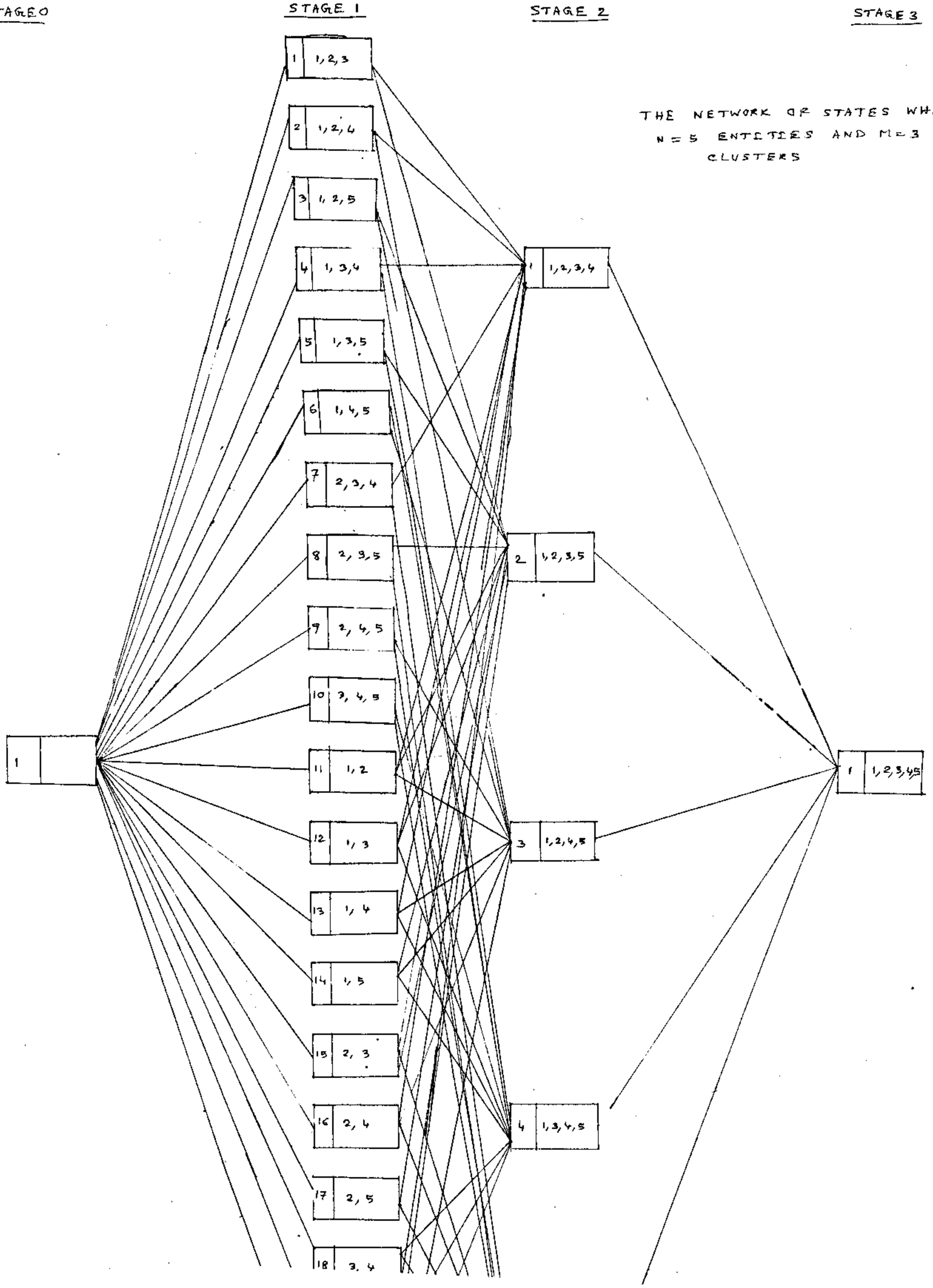
Again from table I, h (1,5) = 0.5 where as

g (1,5) = f ( ) = 0.5 (refers to dummy first stage single set element).

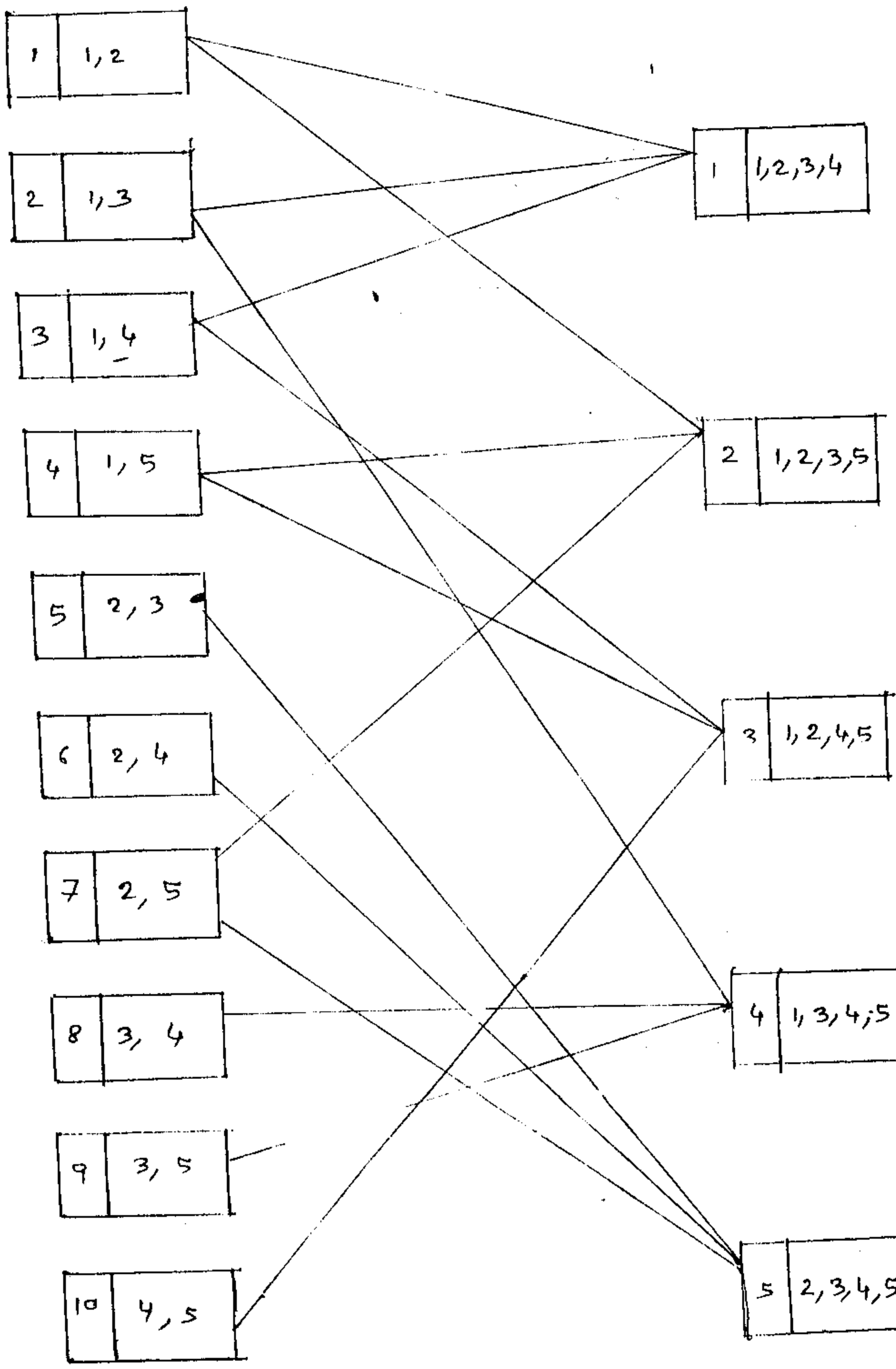
∴ f (1,5) = g (1,5) + h (1,5) = 0.5 + 0.5 = 1.00

i.e. this is the optimum clustering alternative for heuristic search method i.e. 1,5 , 2, 4 and 3.

THE NETWORK OF STATES WHEN N = 5 ENTITIES AND M = 3 CLUSTERS

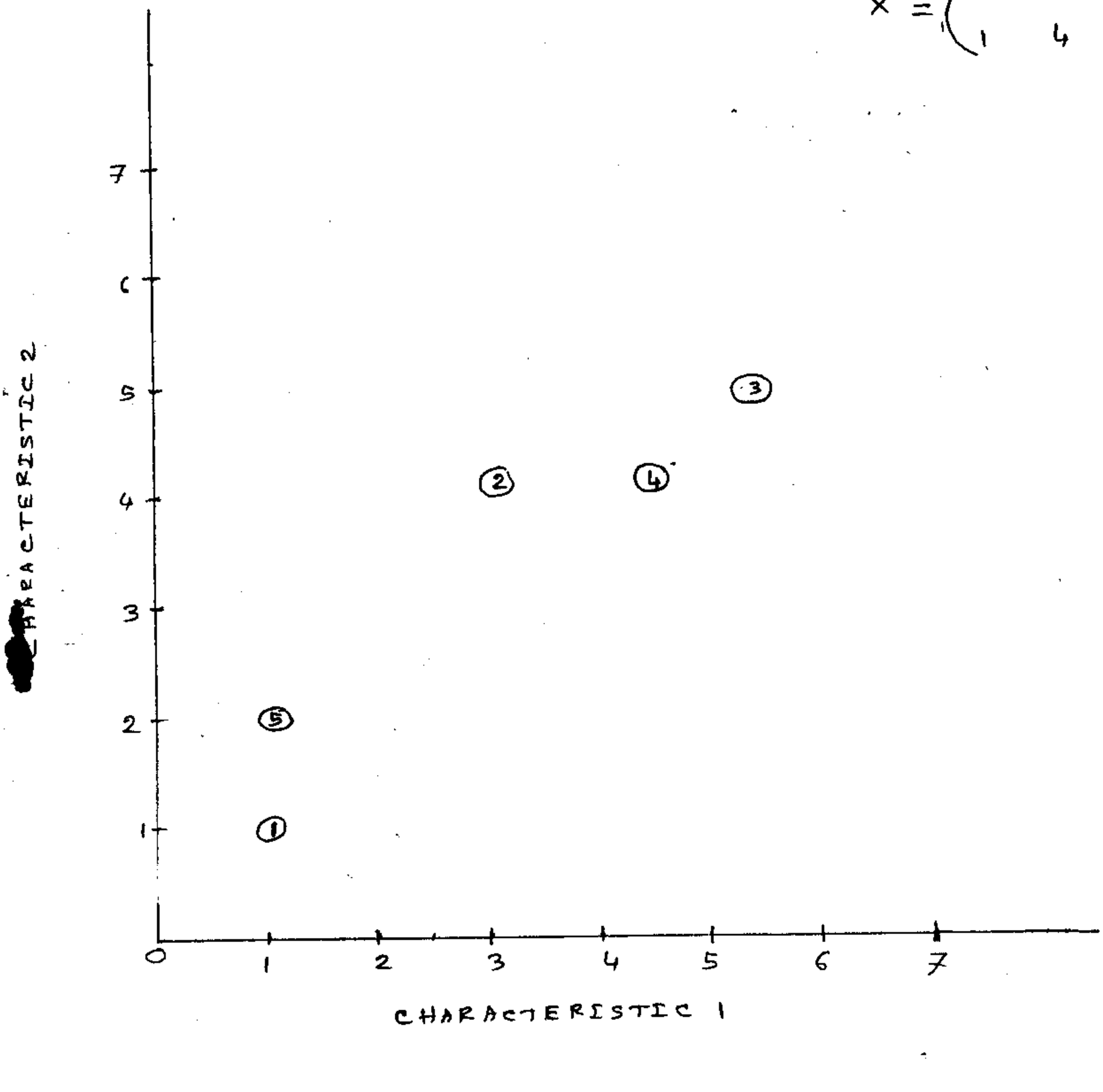


STAGE 1 | ARCS ELIMINATED | STAGE 2



NOTE THE 5×2 X matrix was

$$X = \begin{pmatrix} 1 & 3 & 5 & 4 & 1 \\ 1 & 4 & 5 & 4 & 2 \end{pmatrix}$$



GRAPH OF N = 5 ENTITIES