COENOSES 10(1):17-28, 1995

CLUSTERING GEOBOTANICAL DATA WITH THE USE OF A GENETIC ALGORITHM

I. R. Moraczewski[1], W. Borkowski and A. Kierzek

Dept. of Plant Systematics and Geography, Warsaw University Al. Ujazdowskie 4, PL-00-478 Warsaw, Poland

Keywords: Classification, Cluster analysis, Genetic algorithm, Partitioning

Abstract: This paper considers the problem of clique-partitioning. Existing algorithms that optimally solve this problem employ a graph-theoretical perspective which is not suitable for geobotany. Instead, a GA-based procedure for clique-partitioning is presented, that can be applied to data of any kind. The fitness function involves a stress measure between dissimilarity matrix and the resulting classification. Emphasis is placed on examining the behavior of the proposed objective function with regard to its success in discovering the structure of data sets. Results of numerical experiments conducted on artificial and real-world examples are reported.

Introduction

Classification is one of the two most important tools of geobotanical data analysis. As the objects (communities, associations, etc.), in contrast to biological individuals, show tendency to form a *continuum* and do not interchange any genetic material in the course of evolution, a set of classes, i.e. a partition rather than a hierarchy is the appropriate means of representing relationships in data set (Goodall 1986). There are three basic ways leading from a (dis)-similarity matrix to non-hierarchical classifications:

1) hierarchical (agglomerative or divisive) clustering with stopping rules and splitting levels,

2) k-partitioning, i.e. division of the collection into k subsets,

3) clique-partitioning[2] (k not fixed).

Hierarchical clustering is the most commonly used classification method (Mucina & van der Maarel 1989), mainly because of its speed, ease of implementation, and availability. However, the transformation of hierarchical classification into a non-hierarchical one is very likely to be unsuccessful for two reasons. First, each step of the clustering procedure is performed deterministically which results in poor sampling of search space (Lucasius et al. 1993), hence the dendrogram obtained can be very distant from the optimal one. Second, having prepared a tree, the user has to make use of Occam's Razor which is a very unsafe thing (Dale 1988b).

Among methods of exploratory data analysis, k-partitioning has a somewhat strange status. It is concerned with the problem of revealing the hidden structure of given data set, but at the same time it demands from the user some non-trivial knowledge of the structure. Since k must be chosen *a priori,* the user is doomed to follow various rules of thumb (cf. Ross 1979) and intuition (Goodall 1973).

Clique-partitioning does not assume any previous knowledge of the number of clusters, therefore it seems to present a suitable perspective for vegetation data analysis. In this paper we apply a genetic algorithm to find the best clique-partitioning of the data collection. Our experiments suggest that the method presented in this paper is promising.

Clique-partitioning

Several attempts have been made to construct algorithms that can solve the clique-partitioning problem which is NP-complete (see Dorndorf & Pesch 1993). Grötschel & Wakabayashi (1989, 1990) proposed the branch and bound method; simulated annealing and tabu search were applied for clique-partitional clustering by Amorim et al. (1992); Dorndorf & Pesch (1993) provided a clique-partitioning procedure based on ejection-chain heuristics. All these algorithms, involving the graph-theoretical formal apparatus, start from a weighted complete graph (= primary data table) and cut it into subgraphs referred to as cliques. They perform

1 Fulbright scholar at Wichita State University, Dept. of Computer Science, Wichita, KS 67260-0083, USA

2 The term "clique-partitioning" in specialized mathematical papers is reserved for procedures of dividing complete graphs (or, more appropriately, sets of their vertices) into subgraphs (subsets of vertices), such that a "goodness" function of the resulting classification is maximized. In our work we consciously use this term, for lack of a better one, as the name for any algorithm leading to the optimal partition of a given set into non-empty, mutually exclusive subsets of unbounded size and number.

very well, that is are fast and lead to optimal solution, in most random and real-world problems examined in cited papers.

Nevertheless, as potential tools of geobotanical data analysis, they have some disadvantages. They try to maximize the similarity within the clusters, i.e.

$$\text{maximize} \sum_{1 \le i < j \le N} w_{ij} r_{ij}, \quad (1)$$

where N is the total number of objects, w_{ij} the edge weight between vertices i and j, and r_{ij} a binary variable taking 1 if i and j belong to the same clique, and 0 otherwise. Although such a simple objective function is very convenient for fast computations (given a partition, its neighborhood can be easily explored), it neglects distances between clusters, and therefore can lead to solutions that, although optimal according to (1), do *not* reflect the structure of the data. Moreover, the function discussed is designed for qualitative data only, which is clearly manifested in the definition of edge weights.

Edge weight, w_{ij}, is defined by $2s_{ij} - l$, where s_{ij} is the number of attributes possessed in common by i and j, and l denotes the total number of characteristics. In the case of boolean variables this formula becomes a + d - b - c, where a, b, c, and d are values of a 2x2 contingency table. Such a presence-absence resemblance coefficient has a limited importance to practice. Quantitative and mixed data sets can be processed only after a rough transformation of values for each attribute into a binary relation (equal vs. unequal) with the use of a threshold value which must be established *a priori*. For example, in Grötschel & Wakabayashi (1989) as well as in Dorndorf & Pesch (1993) the equality thres-hold value for micro computers data was fixed at 0.3. But why could not it be set to 0.65? The authors do not explain this.

It is evident that all the clique-partitioning methods involving the problem (1), though fast, do not satisfy the vegetation scientist's needs. We shall present an efficient GA-based clique-partitioning procedure that in a reasonable time provides good solutions and can be applied to dissimilarity matrices of any kind.

Genetic algorithms

Genetic algorithms (GA) developed by Holland (1975) have been applied to various optimization problems (Goldberg 1989). In contrast to traditional deterministic optimization procedures, they try to mimic the probabilistic evolutionary process. The GA usually starts with a random population of strings, each one representing a possible solution to the investigated problem. Next steps of a GA are repeated iteratively. Each iteration is referred to as a generation. In each generation, for each member of the population, each member's performance is computed according to a fitness (goodness) function, and then three "genetic" operators are executed:

1) *Duplication.* The new population is created from copies of strings of the old population, after considering their relative fitness. Let us assume the fitness function is to be maximized. Then strings with a high value of fitness function give offsprings, whereas the worst ones are rejected.

2) *Crossover.* It is a mixing of strings ("chromosomes") which insures innovation. In the simplest case the crossover operator splits two randomly chosen chromosomes at the same point and then crosswisely reassembles them.

3) *Mutation.* It is a fitness-independent operator. It alters a randomly chosen bit in a randomly chosen chromosome. As a Monte Carlo component of GA, it gives a possibility to avoid the search to be "trapped" by local fitness function optima.

Here is the GA scheme in a C-like notation:

```
Population design();
For each generation
{
    For each chromosome
        Calculation of fitness();
    Duplication();
    Crossover();
    Mutation();
}
```

GA-based clustering

Genetic algorithms for clustering problems have only recently attracted the attention of investigators. Rhagavan & Agarwal (1987) first used the reproductive plan for a special clustering task, namely for optimal determination of user-oriented clusters in an information retrieval system. Bhuyan et al. (1991) presented a GA-based method for k-partitioning that performed as good as or better than the greedy algorithms ABF and AFB developed by Ismail & Kamel (1989). Lucasius et al. (1993) applied a genetic algorithm to the problem of k-medoid clustering (a variant of k-partitioning), which proved to accomplish a better sampling of the combinatorial search space than CLARA did - yet another method for k-medoid clustering. Mühlenbein et al. (1988) provided an evolution k-partitioning algorithm involving a function that maximized "the intra-partition traffic".

As it can be seen, the problem of k-partitioning attracted investigators trying to cope with it in a Darwinian manner. GA-based clique-partitioning, however, is a task different enough to require the elaboration of specific string representation and a special fitness function.

Representation

The partition is a fairly sophisticated structure which poses some difficulties in encoding. Ideally, a good representation should meet two requirements: it should permit the encoding of *each* element of the domain and be such as to enable the application of standard exploration operators. Unfortunately, these two demands are conflicting. An exact, verbatim and natural encoding usually rules out the application of the well-known mutation and crossover operators. In turn, the traditional representation (as binary strings) results either in confinement of the search space and/or leads to high time complexity associated with genetic operations on these

representations. Further details on the latter issue can be found in Bhuyan et al. (1991).

The representation used in our paper takes its main idea from works on k-medoid clustering (see, for example, Lucasius et al. 1993). According to this clustering framework, a partition of N-element set (N > 2) into k blocks is defined by a collection of k objects being representatives for these clusters. The remaining elements of the set are assigned to nearest representatives using a distance measure. The more central the location of an object within a cluster, the better representative it is for that cluster. For this reason the optimal representative objects are called medoids. Analogously, in our algorithm, a classification takes the form of a string consisting of 0s and 1s. If an object is 'set to 1', it is treated as a representative of a (possibly one-element) cluster, else it is assigned to a cluster defined by the representative closest to it according to a chosen dissimilarity measure (Fig. 1). Contrary to the k-medoid model, the number of clusters is unbounded and is subject to changes during the program's execution. It is assumed, however, that "pure" strings, i.e. consisting only of zeros or ones, or including only one '1', are not allowed. Such chromosomes being the counterparts of trivial classifications (k = 1 or k = N) are treated as "dead" and rejected.

The advantage of the adopted representation is its simplicity which enables the application of traditional genetic operators that are already quite well researched and understood. However, due to this representation, the domain is restricted to no more than $2^N - N - 2$ partitions. Moreover, some classifications can have more than one representation (Fig. 1), while others cannot be represented at all in the way we propose (Fig. 2). In particular, this relates to classifications composed of oblong, tangled and/or overlapping clusters. But such clusters are of rather limited importance for cluster analysis, with the goal to determine *clouds* of points in multidimensional character space. Our algorithm, similarly to the k-medoid model, deals with such partitions that consist of more or less hyperspherical and mutually separated clusters. This kind of clusters is one favored by the fitness function.

In contrast to the k-medoid framework, where the objective function is the sum of the distances between medoids and all other objects of the same group, in our approach the representatives, although define clusters, do not play any role in computation of fitness.

Fitness function

Let A be the set of objects. The cardinality of A will be denoted by N. Suppose N > 2. Let $\mathbf{D} = [d_{ij}]$, i, j = 1, ..., N, be a dissimilarity matrix which meets two requirements:

1) $d_{ij} = 0 \Leftrightarrow i = j$,

2) $d_{ij} = d_{ji}$.

Let $P = \{C_1,...,C_k\}$ be a partition of A into k classes whose cardinalities will be denoted by $N_{C_1},...,N_{C_k}$. We bear in mind that k is initially unbounded. Let us define the fitness function F assigning a value ranging over the [0,1] interval to each partition P of the set A:

$$F(P) = \begin{cases} 0, & \text{if } k=1 \text{ or } k=N \\ 1/(1+s(D,\Delta)), & \text{otherwise} \end{cases} \qquad (2)$$

where $S(\mathbf{D},\Delta)$ is a stress that measures how well a matrix $\Delta = [\delta_{ij}]$ corresponding to the partition P matches the dissimilarity matrix **D**. It is assumed that $S(\mathbf{D},\Delta) \geq 0$. The smaller $S(\mathbf{D},\Delta)$, the better the approximation of **D** by Δ. In the simplest case one would assume Δ to be equal to the complement of equivalence relation induced by P. If one went a

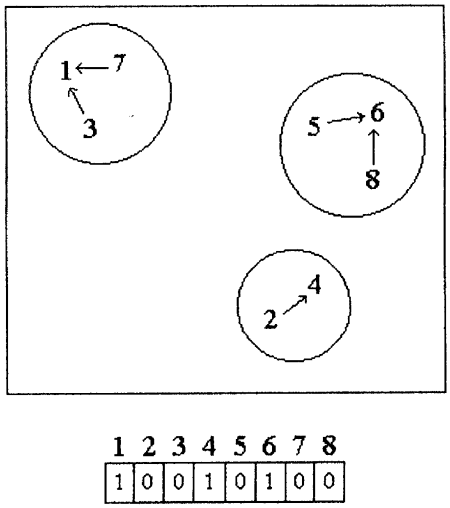

Figure 1. Partition of a two-dimensional data set and its string representation. Euclidean distance is assumed. Note that there exist several alternative strings for this classification (01100001, for example).

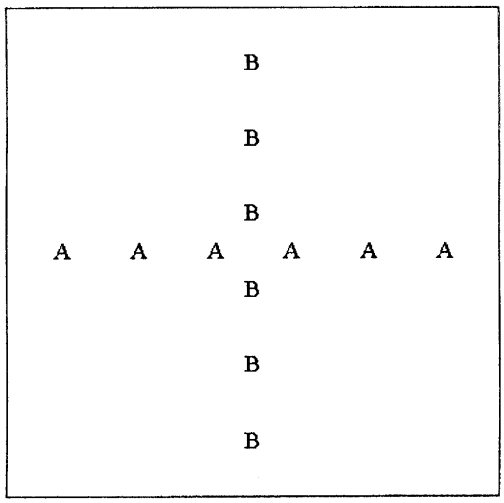

Figure 2. Example classification of a 2-dimensional artificial data set that cannot be represented by any binary string of the kind proposed for our algorithm. Euclidean distance is assumed, cluster membership is denoted by arbitrarily chosen characters.

step further and replaced **D** by a binary dissimilarity (incompatibility) relation, one would arrive at Zahn's (1964) problem which is reducible to (1) (Amorim et al. 1992). These transformations would clearly result in a loss of information conveyed by the distances and would lead to an oversimplified version of our problem, what in turn could affect the quality of obtained solution. To avoid this we suggest such a definition of the reflexive and symmetrical matrix Δ that yields, loosely speaking, a quantitative description of P based on **D**:

$$\delta_{ij} = \begin{cases} \bar{d}_C, & \text{if objects i and j belong the same cluster C,} \\ \bar{d}_{EXT}, & \text{otherwise} \end{cases}$$

where $\bar{d}_C$ is the mean distance within the cluster C, and $\bar{d}_{EXT}$ stands for the mean distance between objects from different clusters.

While searching for the stress formula that would be the best for our purposes we have analyzed the stress originally introduced by Kruskal (1964) as well as its various modifications (Young & Null 1978, Aivazyan et al. 1989). Finally, we have chosen the following variant:

$$S(\mathbf{D},\Delta) = \sqrt{\sum_{1\le i<j\le N} \frac{(d_{ij} - \delta_{ij})^2}{d_{ij}^2}} \qquad (3)$$

that proved to fare well regardless of data type, number of clusters, etc. The others were biased in favor of distinguishing too few groups. They also showed a tendency not to distinguish cluster boundaries.

Turning now to fitness, we see that to maximize the objective function (2) is the same as to minimize (3), which is achieved by execution of the genetic algorithm sketched above.

Exploration operators

In our implementation of the genetic algorithm we utilize single point crossover procedure (Fig. 3) and simple mutation operator (Fig. 4). The selection of individuals for a new population is performed by means of tournaments. The randomly selected specimens are compared with respect to their fitness and the worse of the two is replaced by the better one with the probability P_R. Note that the greater P_R, the greater the selective pressure. If $P_R = 0.5$ there is no selective pressure at all. In addition to that, we copy the fittest individual into the new population.

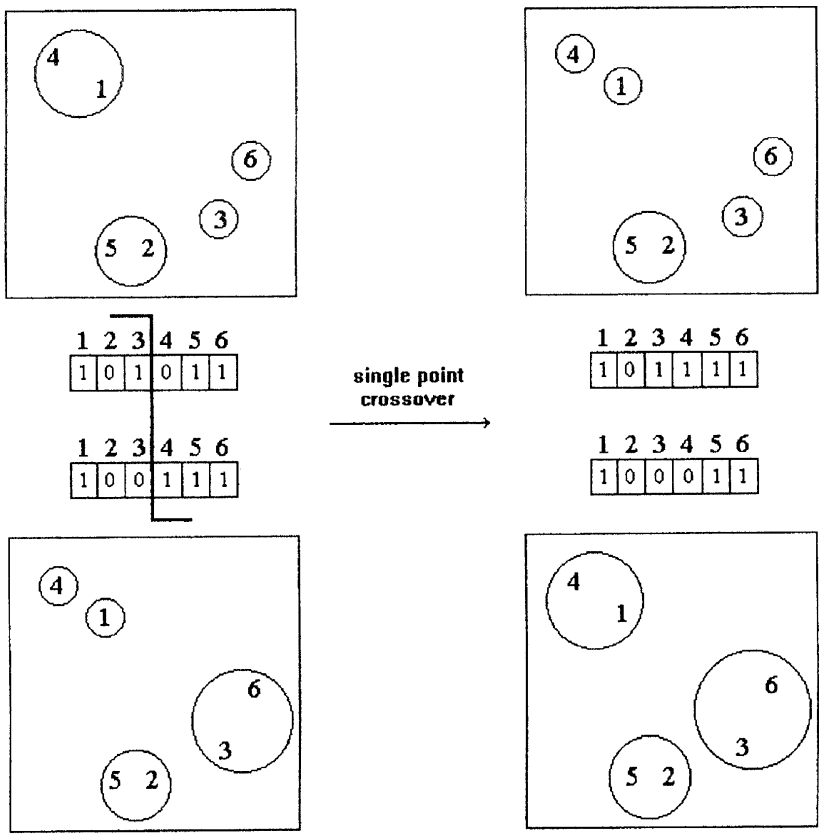

Figure 3. Single point crossover scheme.

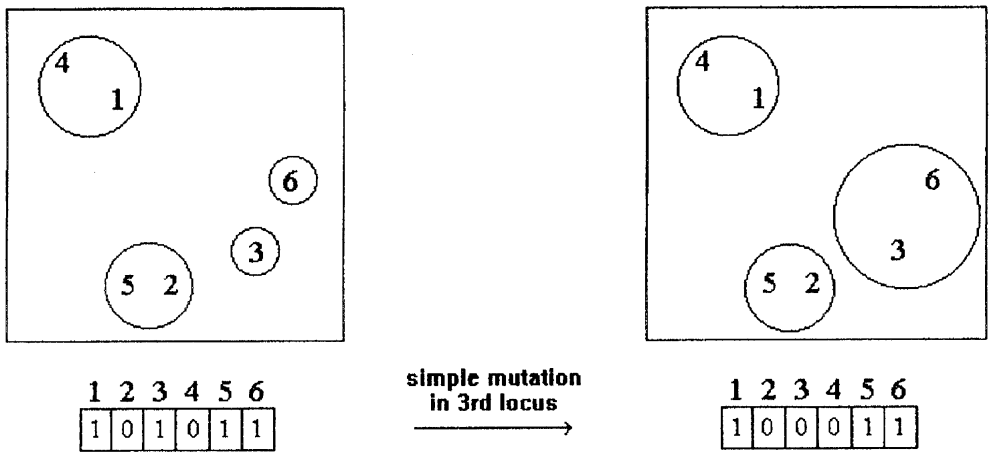

Figure 4. Simple mutation scheme.

Experimental results and evaluation

Clustering methods can be evaluated according to the range of their application, quality of end solutions, and computation time required. Other possible criteria, such as the amount of computer memory required, ease of developing and use, seem to be of lesser importance.

Our program of GA-based clique-partitioning accepts as input a distance matrix. Thus, it can deal with data irrespective of the application domain and variable types. The only essential constraint imposed on the input matrix is that it cannot contain any zero outside the diagonal. If there are two or more identical objects in the data set, they must be joined up prior to the analysis. They can be easily identified afterwards using a multiple label.

Before we answer the questions regarding what solutions our method provides and how fast the program performs, we shall discuss the problem of its configuration.

Configuration

For a genetic algorithm to be effective it is necessary to define parameters measuring various aspects of its performance: the population size and the probabilities of applying genetic operators. In general, there is no satisfactory universal method that will determine the tradeoff among these values and decide their optimal combination. This can be ascribed to the high complexity of genetic algorithms. One of the most successful strategies to optimize the configuration is to make educated estimates stemming from the practice and based on experts' intuition (Lucasius et al. 1993). Other approaches, like the fractional factorial design (Lucasius et al. 1993) or application of meta-genetic algorithms (Grefenstette 1986) are not feasible, because they are computationally very costly.

Following the guidelines obtained from the literature on the subject, we have established a realistic range for each essential parameter in GA, i.e. the population size S, crossover, mutation and reproduction (duplication) probabilities, P_C, P_M, and P_R, respectively, and conducted several numerical experiments to examine the influence of values selected from within these ranges on our program's performance. For the experiments we used a randomly generated two-dimensional `c_test` example (Fig. 5, see Table 1 for coordinates) in which clusters are relatively difficult to distinguish.

There are at least two ways of determining the end of evolution. According to the first one, which is commonly used in a vast majority of GA-based applications, the user simply defines the number of generations. This approach, however, is not appropriate when calibrating GA, since it makes it difficult to compare the effectiveness of the al-

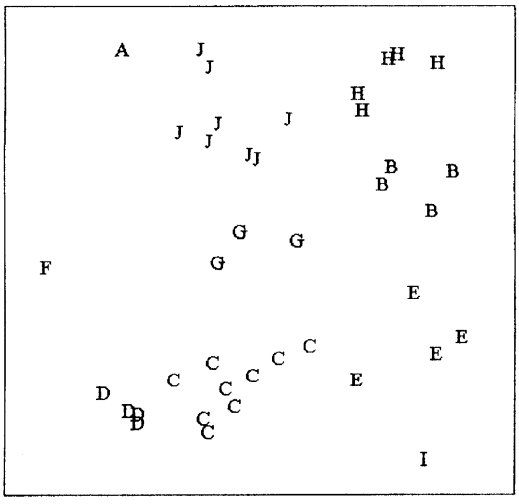

Figure 5. Best partition found for `c_test` artificial data set.

Table 1. Two-dimensional test problems.

Data file		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
c_test	x	32	83	20	82	47	92	59	91	63	39	50	44	56	34	74	74	80	41	40	41	16	50	87	3	41	60
	y	80	5	4	31	46	7	20	41	72	89	28	82	75	23	14	80	35	8	92	76	83	79	60	54	25	48
	x	22	24	24	98	75	96	46	52	81	90	43	42	92	39												
	y	87	90	88	70	18	32	86	29	6	98	21	53	74	4												
random1	x	93	86	54	54	10	62	65	23	22	98	68	14	73	93	63											
	y	7	86	9	77	2	10	28	69	4	92	43	65	19	69	74											
random2	x	30	76	79	92	53	59	94	74	50	66	7	60	45	54	51											
	y	32	43	39	82	0	75	60	93	73	68	68	38	65	41	36											
concentr	x	3	4	5	6	7	8	8	8	8	8	8	7	6	5	4	3	3	3	3	3	5	6	6	5		
	y	3	3	3	3	3	3	4	5	6	7	8	8	8	8	8	8	7	6	5	4	5	5	6	6		
latch	x	4	3	3	6	6	1	2	6	4	2	2	4	5	1	5	5	6	6	4	1	3	6	1	1	3	1
	y	3	3	5	5	3	5	7	7	1	1	3	5	1	3	5	7	4	2	7	6	7	1	4	7	1	1
cross	x	15	15	15	15	15	15	15	15	14	16	13	17	18	12	19											
	y	14	13	12	11	10	9	8	7	10	10	10	10	10	10	10											
2circles	x	5	5	5	10	10	10	10	10	15	15	15	15	15	20	20	20	20	20	25	25	25	25	25	29	29	30
	y	20	15	10	25	20	15	10	5	25	20	15	10	5	25	20	15	10	5	24	20	15	10	6	20	10	15
	x	34	35	35	39	39	39	39	39	44	44	44	44	44	49	49	49	49	49	54	54	54	54	54	59	59	59
	y	15	20	10	24	20	15	10	6	25	20	15	10	5	25	20	15	10	5	25	20	15	10	5	20	15	10

gorithm running under various configurations. Consider, for instance, an increment of P_C. Perhaps it will improve the solution found by the last generation, but it will certainly diminish the number of generations per time unit (fitness of the new specimen must be computed!), and lenghten the whole experiment. Therefore, it is reasonable to apply the termination criterion based on the allocated running time.

The program, written in C, was run on a Challenge L computer (MIPS R4400 processor, 150 Mhz). The running time of each experiment was set to 10 s. A hundred runs for each configuration were performed.

We used the following measures for the configuration's evaluation:

- the percentage of runs in which the best obtained (and, most probably, optimal) solution was found,

- the harmonic mean of number of chromosomes eliminated during the evolution. We named this parameter the cost of evolution. Since this measure proved to be reversely proportional to the first one, in what follows we will report our results referring only to the percentage of successful runs.

Our main numerical experiment was 3-dimensional. Population size (S) was fixed at 50, duplication ratio (PR) was set to 0.7. Mutation and crossover probabilities were changed from 0.0 to 0.14 and from 0.0 to 1.0, respectively. The results are displayed in Fig. 6. In agreement with expectations, too low and too high mutation rates result in poor program's performance. The optimal value for P_M lies between 0.07 and 0.09. In turn, crossover seems to be the operator that hardly influences the program's performance. For $P_C = 0.6$ the results are as good as those obtained with no crossing-over. With other P_C values greater than zero our program fares a little bit worse. Possible explanations of this fact are provided in the Discussion.

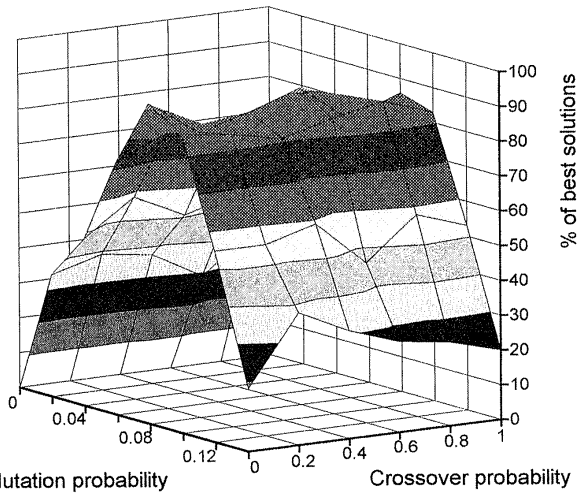

Figure 6. Percentage of successful runs as a function of cross-over and mutation probabilities. Calculations were carried out for `c_test` 2-dimensional example within 10 s running time. Population size (S) was taken to be 50, reproduction probability (P_R) was fixed at 0.7.

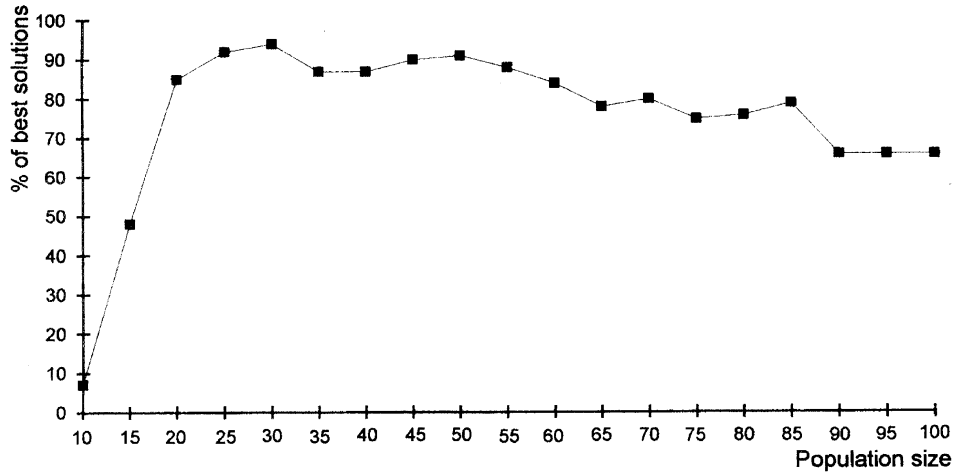

Figure 7. Percentage of successful runs vs. population size. The experiment was conducted on `c_test` example, $P_M = 0.08$, $P_R = 0.7$, $P_C = 0.6$.

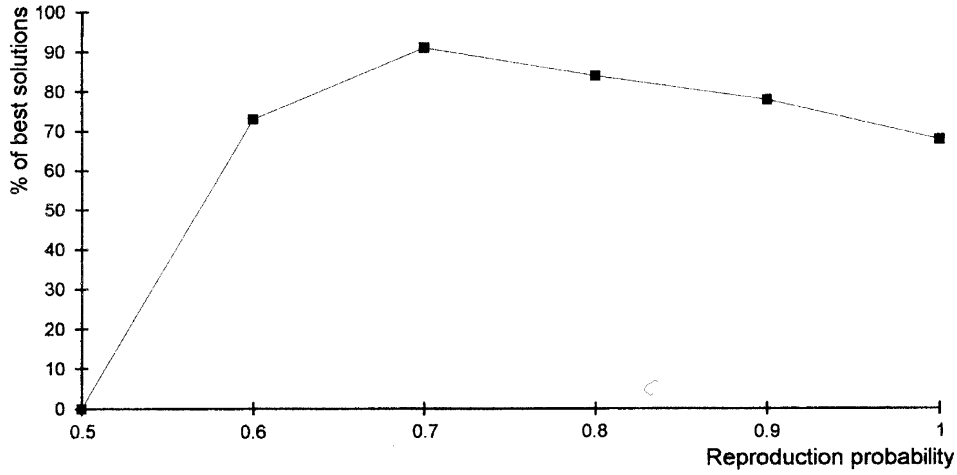

Figure 8. Percentage of successful runs vs. reproduction probability. The experiment was conducted on `c_test` example, $P_M = 0.08$, $P_C = 0.6$, $S = 50$.

In order to observe the influence of population size and reproduction rate on our program's performance, we designed 2-dimensional experiments in which constant crossover and mutation rates were maintained ($P_M = 0.08$ and $P_C = 0.6$). Results are presented in Figs 7 and 8. It can be seen that very small populations do not yield good results. Maximum performance was obtained for 25 to 60 strings in population. The reproduction rate $P_R = 0.7$ was found to be the best among the values tested.

The minimum requirement for a genetic algorithm is that it should perform better than a random search (i.e., a search with no selective pressure). It can be seen in Figs 6 and 8 that GA passed such a test successfully.

Test problems

Experimental evaluation of the proposed algorithm was conducted on several data sets. First, to get a feeling for how the fitness function works, we applied our algorithm to 2-dimensional examples which usually have only a few local optima and therefore are rather easy to solve. To evaluate our program on real-world problems and against known results, we ran it on some geobotanical examples drawn from

Table 2. Results obtained by the GA-based clique-partitioning. $P_M = 0.08$, $P_R = 0.7$, $P_C = 0.6$, $S = 50$.

Example	Source	*N*	Dissimilarity measure	Best solution found	Fitness of the best partition	Average time [s] (harmonic mean)	Number of best solutions in 100 runs
c_test	(original)	40	Euclidean	(1,9-10,12-13,19-20,22,33) (2,6,15,31,35) (3) (4,8,17,32) (5,26,38) (7,11,14,18,25, 34,37,40) (16,23,30,39) (24) (36) (21,27-29)	0.046027	9.7	41
random1	(original)	15	Euclidean	(1,3,6-7,13) (2,10,14) (4,15) (5,9) (8,12)	0.172848	0.0	100
random2	(original)	15	Euclidean	(1,5,11)(2,3) (4,7-8) (6,9-10,13) (12,14,15)	0.175771	0.2	55
concentr	(original)	24	Euclidean	(1-3,19-20) (4-8) (9-13) (14-18) (21-24)	0.100337	1.2	82
latch	(original)	26	Euclidean	(1-2,11,14,23) (3-4,12,15,17) (5,9,13,18,22) (6-7,20-21,24) (8,16,19) (10,25-26)	0.087405	1.4	99
cross	(original)	15	Euclidean	(1-3) (4-6,9-10) (7,8) (11,14) (12-13,15)	0.138318	0.1	100
2circles	(original)	52	Euclidean	(1-26) (27-52)	0.03809	7.2	84
gertowns	Späth 1980	22	Euclidean	(1,5,10,12,20,22) (2,8,14-17,19) (3-4,7,11,21) (6,9,13,18)	0.108626	0.4	100
soil_data	Lagonegro and Feoli 1985 (taken from Marsili-Libelli 1989)	20	Euclidean	(1-2,4-5) (3,19-20) (6-18)	0.126986	0.2	100
adelaide	Bowman and Wilson 1986 (drawn from Dale 1988a)	41	Canberra	(1-5,7-10,13-17, 21,25) (6,11-12, 18,39) (19,23) (20) (22,27-28, 35,38,40-41) (24,32,36-37) (26,29-31,33-34)	0.107416	14.6	19
nicheovlp4	Yu and Orlóci 1990	35	measure of niche dissimilarity (see source)	(1-10,12,14-17, 21,26) (11) (13,28) (18) (19,31) (20) (22) (23,35) (24, 33) (25,32) (27,29) (34)	0.39191	75.4	5

the literature. For all the experiments we used the following probabilities: $P_M = 0.08$, $P_R = 0.7$, and $P_C = 0.6$. The population size was taken to be 50. All runs were performed 100 times within 10 s running time on INDY PC computer (MIPS R4600 processor, 100 Mhz). The results are compiled in Table 2.

Fig. 9 portrays artificial 2-dimensional examples (see Table 1 for detailed description) together with obtained partitions. In general, the resulting classifications are very intuitive. Although our algorithm cannot capture very sophisticated distribution patterns (cf. `concentr` and `latch` examples, Fig. 9) it is to some extent sensitive to the form of clusters. In particular, in `2circles` example it can be seen that, in contrast to many clustering methods, a minimum distance may not be a within-group one. The algorithm can grasp global forms, disregarding some local (dis)similarities. Our program also fares well in its ability to determine the appropriate number of clusters.

The next example is Späth's (1980) data set which is a collection of Cartesian coordinates of 22 German towns derived from a map. The partition produced by our GA (Fig. 10) turned out to be equal to that obtained by application of Späth's heuristic procedure for k-partitioning in case k = 4 with the advantage of our algorithm that the number of clusters need not be provided by the user.

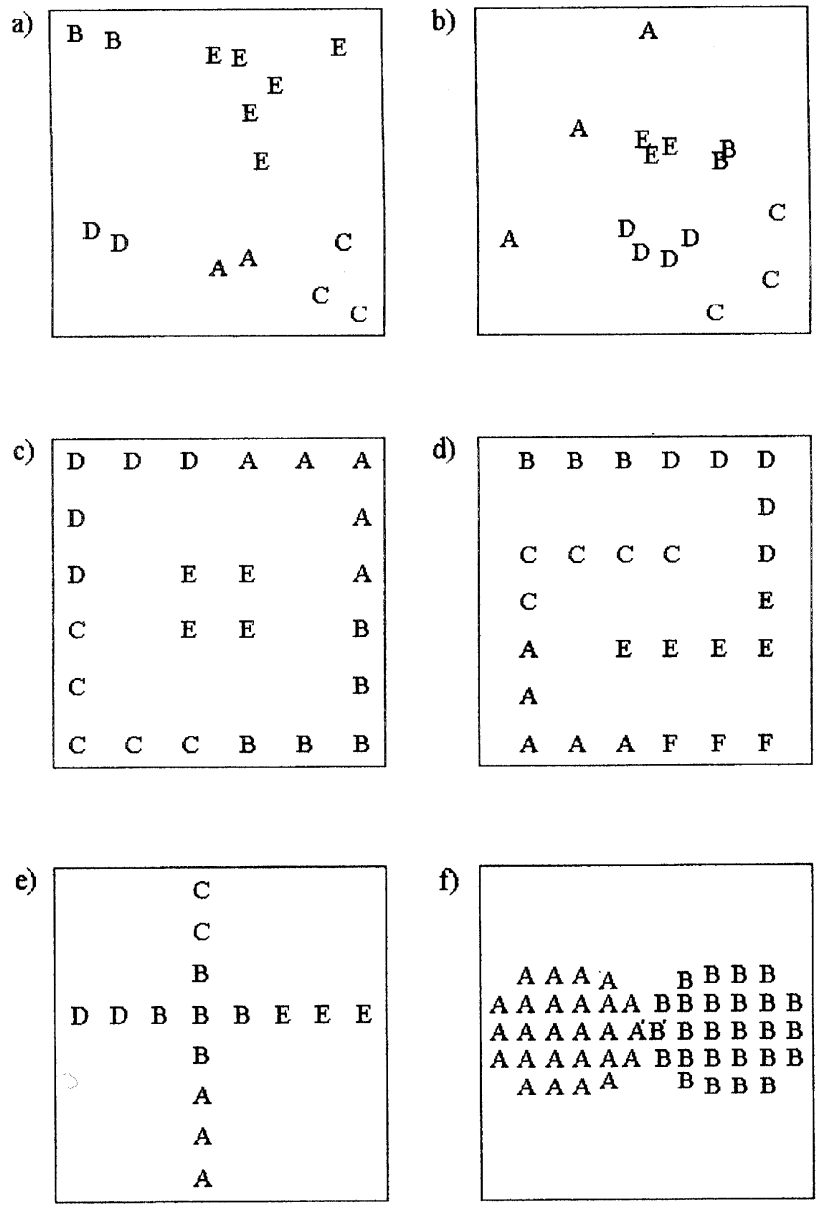

Figure 9. Partitions found for six artificial data sets. a - `random1`, b - `random2`, c - `concentr`, d - `latch`, e - `cross`, f - `2circles`. In the latest example d(A',B') is the minimum distance.

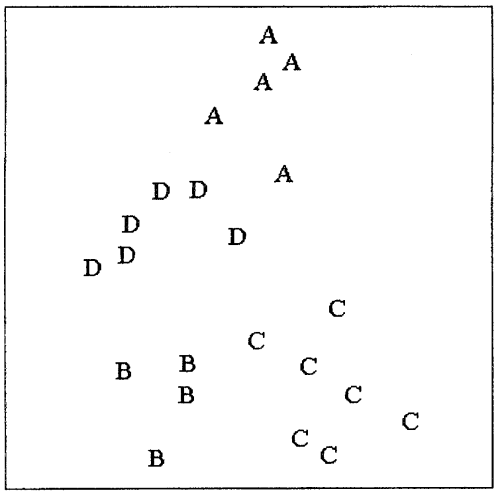

Figure 10. Partition of `gertowns` data set (test problem taken from Späth 1980).

`Soil_data` processed by Lagonegro and Feoli (1985) in a "conventional" way was also analyzed by Marsili-Libelli (1989) within a fuzzy framework. This gives us an opportunity to compare GA-based clique-partitioning with two dendrograms, a crisp and a fuzzy one. The best clique-partitioning found (see Table 2) is fully consistent with the UPGMA conventional dendrogram: there exists an a-cut that will split the tree in three classes corresponding to groups obtained by GA (Fig. 11). It is otherwise if we consider the fuzzy dendrogram. Before leaving this example we want to point out that the compatibility of UPGMA clustering with GA-based partitioning is not surprising. UPGMA is a technique known to yield ultrametrics that is very close to the input dissimilarity matrix (see Sneath & Sokal 1973), and the idea underlying our fitness function is just minimizing stress between the matrix and the resulting partition. Note, however, that some classifications obtained by the GA may not be compatible with the UPGMA dendrogram (see the `2circles` example).

Let us consider the `adelaide` data set consisting of 41 stands from the Adelaide River flood plain in Northern Australia, compiled by Bowman & Wilson (1986). Having applied some standard classification methods to this example (Canberra metrics and flexible sorting), Dale (1988a) obtained a seven-class partition. The classification produced by GA-based method (Table 2) also suggests occurrence of seven clusters, but "cleavage sites" are located in somewhat different places. It is noteworthy that stand number 20, and sites 19 and 23, which were recognized by Dale as aberrant by means of Wong & Liu's (1975) technique, form separate clusters in the GA-based partition. The best solution for the `adelaide` example was reached in 19% of runs which indicates that this data set poses some problems for our algorithm configured as we described earlier.

Compared to problems considered above, the `nicheovlp4` example taken from Yu & Orlóci (1990) is much more difficult. The best partition (supposed to be the optimal one) was reached only in 5 percent of runs (Table 2). After increasing the time scope to 20 s, the best solution was found in 7 percent of runs. To find out the reasons of such poor performance we carried out a more detailed study of this example.

The dendrogram presented by Yu & Orlóci (1990) was apparently computed according to Ward method. On the basis of this tree the authors recognized ten guilds of species. However, other well-known agglomerative methods that we applied (UPGMA, single link, complete link) turned out to give dendrograms completely unlike Ward dendrogram and quite different from one another. In terms of indeterministic approach we would say that the `nicheovlp4` data set contains many local optima, corresponding to various classifications, and each deterministic method reaches one of them, if any. Thus, in such cases each "conventional method" loses its usefulness, since taken separately it yields a result that is totally unreliable. In contrast, an indeterministic approach with a clearly stated objective function seems to be the only way of handling such awkward cases, unless we want to admit that a given data set cannot be successfully classified at all, because a partition itself is not suitable as a means of representing the relationships between the analyzed objects. Great number of local optima decreases the effectiveness of GA-based clustering, which is manifested in low percentage of successful runs, but it by no means impairs the purposefulness of the method. We must bear in mind that using any GA we have no guarantee neither to arrive at the global optimum nor to get high repeatability of obtained solutions. The percentage of runs in which the best solution was found can serve as a measure of applicability of classification as a tool of summarizing data. Low repeatability of best solutions means there are many classifications that are almost

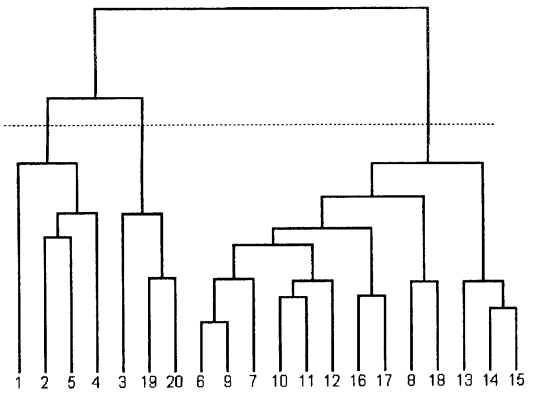

Figure 11. Dendrogram of Lagonegro's and Feoli's (1985) `soil_data` (Euclidean distance, UPGMA). Broken line cuts the tree into 3 classes equal to groups obtained by means of the GA-based partitioning.

equally good, though quite different from one another. The interpretation given above is not intended as an argument that there is no need for further improvement of the presented algorithm. Two possible ways of doing it will be mentioned in the next section.

Discussion

The GA-based clique-partitioning algorithm we presented is likely to be particularly useful for small and medium-size clustering tasks involving the determination of optimal number of clusters as one of the goals. The procedure demonstrated its ability to reproduce intuitive classifications, yielding clearly bad results only for the most sophisticated distribution patterns. It is doubtful, however, if there would exist a general method that will deal with all the awkward cases.

Our algorithm accepts a dissimilarity matrix as input, and therefore, it is applicable to essentially all types of data. This creates an advantage over the existing clique-partitioning algorithms that are too specific in their requirements, and therefore local in scope. It would be an interesting investigation on its own to compare various indeterministic search methods (simulated annealing, tabu search, GAs) utilizing the same general objective function of the kind proposed in this paper. This task, together with possible enhancements of the presented GA, such as local tuning (Bhuyan et al. 1991) and sharing functions (Debb & Goldberg 1989), must be left for future research.

We would like to conclude the paper by making some remarks about the reason why the simple crossing-over operator utilized in our algorithm did not affect significantly the program's performance.

The explanation of somewhat strange behavior of our program may be given on the basis of Holland's (1975) schemata hypothesis. Schemata are templates describing classes of strings sharing some bits, e.g. 1****, or *0*1*. Each schema defines a subset of possible solutions to the problem. The main goal of the crossover operator is to join together different schemata, that is, to link two possible solution's classes and drive the search towards their common part. The difference between this classical genetic mechanism and our procedure is that in our representation any proper subset of fixed bits does not define any exact subset of possible solutions (given a schema there will always exist partitions that cannot be unequivocally said to be defined by the schema or not), because the meaning of any single bit depends on every other bit's setting. Thus, schemata as defined by Holland do not exist. That is probably the reason why the crossover operator does not influence significantly our algorithm's performance.

One should notice, however, that if crossover did not generate any useful solutions, then the experiments with high crossover rate would yield definitely poor results because of high computational cost of new specimen's evaluation. Almost equal results for any crossover probability tested implies that some good solutions are generated by the crossover operator in spite of absence of stringent schemata. It is probably due to the fact that each proper subset of fixed bits, although it does not define any exact subset of solution space, does restrict the space (some solutions are excluded). We expect that in case of larger problems with much longer strings the crossover operator should play a more important role in the GA-based clique-partitioning. Further analysis is necessary.

To elucidate the behavior of our algorithm we can also compare it with the natural evolution. In nature we observe species which effectively evolve without using crossing-over, for instance bacteria and viruses. In the case of these organisms the only engine of their development is a mutation-selection mechanism. Bacterial and viral genomes are very simple compared to eucaryotic ones, but genomes of our "organisms" are even simpler. The number of possible solutions for 40-element example is 240, whereas the "solution space" of simplest virus' genome is about 43500. If viral genomes are able to evolve without crossing over, our "organisms" can do this all the more.

Acknowledgments: We would like to express our gratitude to the Polish EMB-net node at Institute of Biochemistry and Biophysics in Warsaw for making available their computer resources. Special thanks are due to Mrs. Elizabeth Spytkowski, M.A., for her help in the preparation of the manuscript.

References

Aivazyan, S.A., V.M. Buchstaber, I.S. Yenyukov & L.D. Meshalkin. 1989. Applied Statistics. Classification and Reduction of Dimensionality. (In Russian). Finansy i statistika, Moscow.

Amorim de, S.G., J.-P. Barthélemy & C.C. Ribeiro. 1992. Clustering and Clique Partitioning: Simulated Annealing and Tabu Search Approaches. Journal of Classification 9: 17-41.

Bowman, D.M.J.S. & B.A. Wilson. 1986. Wetland vegetation pattern on the Adelaide River flood plain, Northern Territory, Australia. Proc. Roy. Soc.Old. 97: 69-77.

Bhuyan, J.N., V.V. Raghavan & V.K. Elayavalli. 1991. Genetic algorithm for clustering with an ordered representation. In: Belew, R.K. and L.B. Booker (eds). Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, 13.-16. July 1991. pp. 408-415. Morgan Kaufmann Publishers.

Dale, M.B. 1988a. Some Fuzzy Approaches to Phytosociology: Ideals and Instances. Folia Geobotanica et Phytotaxonomica 23: 239-274.

Dale, M.B. 1988b. Knowing when to stop: cluster concept - concept cluster. Coenoses 3: 11-32.

Debb, K. & Goldberg, D. 1989. An investigation of niche and species formation in genetic function optimization. Proceedings of the Third International Conference on Genetic Algorithms, Fairfax, VA, pp. 42-50.

Dorndorf, U. & E. Pesch. 1993. Fast clustering algorithms. ORSA Journal on computing, in press.

Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Goodall, D.W. 1973. Numerical classification. In: Whittaker, R.H. (ed.). Handbook of Vegetation Science. Vol. 5, Ordination and classification of communities. pp. 575-615. Junk, The Haque.

Goodall, D.W. 1986. Classification and ordination: their nature and role in taxonomy and community studies. Coenoses 1: 3-9.

Grefenstette, J.J. 1986. Optimization of control parameters of genetic algorithms. IEEE Trans. Syst., Man & Cybern., SMC-16 1: 122-128.

Grötschel, M. &. Wakabayashi. 1989. A cutting plane algorithm for a clustering problem. Mathematical Programming 45: 59-96.

Grötschel, M. & Y. Wakabayashi. 1990. Facets of the clique partitioning polytope. Mathematical Programming 47: 367-387.

Holland, J.H. 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

Ismail, M.A. & M.S. Kamel. 1989. Multidimensional data clustering utilizing hybrid search strategies. Pattern Recognition 22: 75-89.

Kruskal, J.B. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29: 1-27.

Lagonegro, M. & E. Feoli. 1985. Multivariate Data Analysis. Libreria Goliardica editrice, Trieste (in Italian).

Lucasius, C.B., A.D. Dane & G. Kateman. 1993. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. Analytica Chimica Acta 282: 647-669.

Marsili-Libelli, S. 1989. Fuzzy clustering of ecological data. Coenoses 4: 95-106.

Mucina, L. & E. van der Maarel. 1989. Twenty years of numerical syntaxonomy. Vegetatio 81: 1-15.

Mühlenbein, H., M. Gorges-Schleuter & O. Krämer. 1988. Evolution algorithms in combinatorial optimization. Parallel Computing 7: 65-85.

Raghavan, V.V. & B. Agarwal. 1987. Optimal determination of user-oriented clusters: an application for the reproductive plan. In: Grefenstette, J.J. (ed.). Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, MIT, Cambridge, MA, 28.-31. July 1987. pp. 241-245. Lawrence Erlbaum Associates: Hillsdale, New Jersey.

Ross, D.R. 1979. TAXON Users Manual, ed. P3. CSIRO, Division Computing Research, Canberra, A.C.T.

Sneath, P.H.A. & R.R. Sokal. 1973. Numerical taxonomy. W.H. Freeman, San Francisco.

Späth, H. 1980. Cluster Analysis Algorithms. John Wiley & Sons, New York.

Wong, A.K.C. & T.S. Liu. 1975. Typicality, diversity and feature pattern of an ensemble. IEEE. Trans. Comput. C-24: 158-181.

Young, F.W. & C.H. Null. 1978. Multidimensional scaling of nominal data: the recovery of metric information with ALSCAL. Psychometrika 43: 367-379.

Yu, S.X. & L. Orlóci. 1990. On niche overlap and its measurement. Coenoses 5: 159-165.

Zahn, C.T. 1964. Approximating Symmetric Relations by Equivalence Relations. SIAM Journal on Applied Mathematics 12: 840-847.

Manuscript received April, 1995