Quaderni di Statistica
Vol. 3, 2001

A simulation study on clustering time series with metaheuristic methods

Roberto Baragona
Dipartimento di Sociologia, Università di Roma "La Sapienza"
e-mail: roberto.baragona@uniroma1.it

Summary: Given a set of time series, let $\mathbf{P}(\tau)$ be the cross-correlation matrix function which may be computed from prewhitened residual series. Then, a dissimilarity index is assumed between each pair (i, j) of time series which accounts for the cross-correlations but does not necessarily fulfills the Euclidean distance requirements. Metaheuristic methods are proposed to partition the set of time series into clusters in such a way that (i) the cross-correlation maximum absolute value between each pair of time series that belong to the same cluster is greater than some given threshold, and (ii) the k-min cluster criterion is minimized. The simulation experiment shows that suitably designed metaheuristic methods, and especially the tabu search algorithm, are able to solve this problem successfully, and produce results better than both the single linkage method, and, on the other hand, a pure random search algorithm.

Key words: Autoregressive moving average (ARMA) models; Cross-correlation function; k-mean cluster criterion; Genetic algorithms; Simulated annealing; Tabu search.

1. Introduction

Clustering time series may turn useful in several fields, such as business and economics, demography, environmental sciences, telecommunications, and medicine. The problem I will deal with in the present paper consists in finding a partition of a set of time series based on their cross-correlations. Motivation may be given such as identifying subsets of time series which may be jointly modeled, or seeking for subsets of time series correlated with one or more series of interest. This clustering device may possibly support further investigation about the existence of lead-lag relationships. Moreover, information on which time series turn strongly significantly correlated may find useful applications as far as the forecasting accuracy is concerned. Finally, data mining applications are becoming more and more important as well.

Other approaches I will not pursue here are present in the literature. See, for instance, Shumway and Unger (1974), Piccolo (1990), Bollobás, Das, Gunopulos and Mannila (1997), Corduas (2000), Maharaj (2000), and, in the multivariate framework, Kakizawa, Shumway and Taniguchi (1998), to mention but a few.

I will consider a cluster as a valid one in the present context, and let call it a group, if its time series fulfill a basic requirement as follows (Zani, 1983). Given a set of k stationary time series, a subset C which includes k' series $(k' < k)$ is said to form a group if all $k'(k'-1)/2$ cross-correlations $\rho_{i,j}(\tau)$ satisfy the condition

$$|\rho_{i,j}(\tau)| > c(\alpha) \tag{1}$$

for at least a lag τ between $-m$ and m, and $i, j \in C, i \neq j$.

The cross-correlations $\rho_{i,j}(\tau)$ are to be computed from the residuals of the models of the original time series (see, e.g. Brockwell and Davis, 1996, p. 232). If all time series have n as a common number of observations, then choosing the significance level $\alpha = 0.05$, say, gives the figure $c(\alpha) = 1.96/\sqrt{n}$ in (1). The previously stated definition does not exclude that a time series may belong to more than a single group. Then there are possibly several allowable partitions to consider, and their number may happen to turn very large.

General purpose heuristic methods which may deal with a vast class of problems are often called metaheuristic methods. This common denomination has been including simulated annealing, tabu search and the genetic algorithms. Using metaheuristic methods is worth the while in order to solve a particular problem either if the potential solutions are a largest number, or the requirements for employing otherwise powerful methods do not apply. Both such circumstances are present here, as a choice has to be made among so many admissible partitions which, in turn, form a large discrete set of elements where an objective function with the usual desirable properties, differentiability for instance, cannot be defined. Metaheuristics for clustering were proposed based on simulated annealing by Selim and Al-Sultan (1991), on tabu search by Al-Sultan (1995), and Sung and Jin (2000), and on genetic algorithms by Maulik and Bandyopadhyay (2000), and Tseng and Yang (2001). The availability of some Euclidean distance was generally assumed.

The problem considered in the present paper refers to the cross-correlation matrix function $\mathbf{P}(\tau)$, with elements $\rho_{i,j}(\tau)$. Several dissimilarity indexes were proposed, based on $\mathbf{P}(\tau)$, that do not necessarily meet the requirements for the Euclidean distance assumptions be satisfied. I will use the dissimilarity index (Bohte, Čepar and Košmelj, 1980)

$$d_{i,j} = \sqrt{\{1 - \rho_{i,j}^2(0)\} / \sum_{\tau=1}^{m} \rho_{i,j}^2(\tau)} \tag{2}$$

which is likely to be able to provide a reliable recovery of the clustering structure. The similarity counterpart of (2) will be defined

$$d_{i,j}^{+} = \exp(-d_{i,j}). \tag{3}$$

The index (2) is suitable for the k-min cluster criterion (Sahni and Gonzalez, 1976) which, according to the formulation of Adorf and Murtagh (1988), consists in minimizing the objective function

$$f(C_1, C_2, \ldots, C_g) = \sum_{\omega=1}^{g} \sum_{i,j \in C_\omega, i \neq j} d_{i,j}. \tag{4}$$

In this case, the number of clusters g must be supplied for, otherwise, any algorithm which uses (4) as objective function is likely to assign each time series a separate cluster, by letting $g = k$ at the end. On the other hand, if g were specified as the maximum allowable number of clusters, then the solution will display exactly g clusters.

In order to let the procedure itself determine the number of clusters g, I will use the following objective function to be maximized

$$f^{+}(C_1, C_2, \ldots, C_g; g) = \sum_{\omega=1}^{g} \sum_{i,j \in C_\omega, i \neq j} d_{i,j}^{+}, \tag{5}$$

where g is unknown, and the similarity index (3) is included. When using (5), it is crucial that each cluster be a group, according to (1), for, otherwise, any algorithm, unless prematurely ended, will put together all time series into a single cluster.

It looks convenient to code any valid time series partition in permutation form. Each time series is labeled with a positive integer number between 1 and k. Then, let $(i_1, i_2, \ldots, i_k)$ be a permutation of $(1, 2, \ldots, k)$. Given the significance level α, the permutation will be given its proper meaning as follows.

[1] The first time series, labeled i_1, is taken as the first element of the first cluster.

[2] Let i_2 be considered. If the maximum absolute value cross-correlation between the time series i_1 and i_2, computed after prewhitening them, is greater than $c(\alpha)$, then i_2 joins i_1 into the first cluster. Otherwise, the time series i_2 is to become the first element of the second cluster.

[3] The i_j-th time series adds to an existing cluster if (1) turns true for all pairs belonging to it. If such a circumstance applies for more than one cluster, then the cluster ω is chosen for which $\sum_{i \in C_\omega} d_{i,i_j}^{+}$ is greatest.

[4] The decoding procedure ends as soon as each time series belongs to a cluster.

The choice criterion included into step [3] may look somewhat arbitrary, but it proved necessary, for if, for instance, the time series were assigned so as to maximize the overall criterion, then some undesirable penalization of small clusters would be introduced.

The plan of the paper is as follows. In the next section the simulated annealing, tabu search, and genetic algorithms to be used for solving the time series partitioning problem, as previously stated, will be presented. The plan of the simulation experiment will be illustrated in section 3. Comparison of the results that I obtained from applying the algorithms to the simulated time series sets will be accounted for in section 4. I will draw some concluding remarks in the last section.

2. Metaheuristic methods

The simulated annealing, tabu search and genetic algorithms were implemented as follows, where the definition of group based on (1), the similarity index (3), and the objective function (5) to be maximized were adopted.

Simulated Annealing

Let a permutation $(i_1, i_2, \ldots, i_k)$ be generated at random, and assume it as initial tentative solution. Then, let the associate partition be decoded by applying the steps [1-4], as illustrated in the preceding section. Compute the objective function f^+. An initial temperature T_0 has to be chosen, and let $T = T_0$. The following two steps are executed for a pre-specified number of times n_t.

[1] A new permutation is generated from the current one, and let f^+_{new} denote its objective function value. The difference

$$\Delta = -f^+_{\text{new}} + f^+$$

is computed. The new permutation is taken to replace the current one if either $\Delta < 0$, or $U(0,1) < \exp(-\Delta/T)$, where $U(0,1)$ is a uniform random number in the interval $(0,1)$. This search is repeated for a pre-specified number of times N.

[2] Let $T = T \times \rho$, where ρ is a given pre-specified real number between

0 and 1, 0.95 for instance. This step is referred to as lowering the temperature.

As far as step [1] is concerned, I adopted the device of swapping two labels selected at random in the current permutation to obtain the new one.

Every time a value $\Delta < 0$ was found through the iterations, it was recorded as the best solution to date. The value returned by the algorithm when iterations were complete was assumed as the final solution.

The code has been developed based on the algorithm AS298 (Brooks, 1995).

$Tabu\ Search$

The tabu search procedure adapted to the present clustering problem is taken from Al-Sultan (1995). Differences are due to the use of a dissimilarity index instead of the Euclidean distance, and the number of clusters does not need be supplied from the user, but is determined by the procedure itself.

Let A_c denote a random permutation $(i_1, i_2, \ldots, i_k)$ of the integers from 1 through k, and let $A_c(j) = i_j$. A_c is assumed as the current solution, from which both the group membership and the objective function (5) may be computed as before. A tabu list, empty at the beginning, is prepared to temporarily record the solutions, $mtls$ at most, which will be accepted. Given a pre-specified number of iterations, N say, the following steps are executed N times.

[1] Starting from A_c, nts permutations are randomly generated, being nts some pre-specified positive integer. Let A_t be any of these nts permutations, and P denote a pre-specified threshold probability. Then, for $i = 1, 2, \ldots, k$, $A_c(i)$ is copied into $A_t(i)$ if $U(0, 1) < P$, otherwise a positive integer $j, j \neq i$, is randomly selected, and $A_c(i)$ and $A_c(j)$ exchange.

[2] The nts trial solutions are examined and possibly one of them is taken

to replace A_c. Firstly they are ordered according to their objective function values, the best solution given the first place. Then, three cases are considered. (*i*) The best trial solution is not tabu: A_t replaces A_c. (*ii*) The best trial solution is better than A_c: A_t replaces A_c. (*iii*) If neither (*i*) nor (*ii*) occurs, then both (*i*) and (*ii*) are repeated, up to a pre-specified $itmax$ allowable number of times, from the second through the nts-th best trial solution.

[3] The new current solution is inserted at the end of the tabu list. If the tabu list present size exceeds the pre-specified positive integer $mtls$, the first item in the list is deleted.

The algorithm provides as final solution the permutation which, at the end of the N iterations, is the current one.

Genetic Algorithm

The algorithm that I implemented here was developed essentially along the guidelines provided by Jones and Beltramo (1991), as the coding adopted makes the objective function to depend only on the ordering information.

Given a pre-specified positive integer $size$, a set of permutations $(i_1, i_2, \ldots, i_k)$ is generated at random to form an initial population with $size$ elements. Unlike the common statistical practice, the term population is used here to denote a set of potential solutions. The initial population is assumed as the current one, and, through a pre-specified number of iterations N, the following steps are executed.

[1] Selection. The objective function, that must be positive real valued, is computed for each of the $size$ permutations. Then, for $size$ times, a permutation is selected at random with probability proportional to its fitness. The $size$ selected permutations form the new population, and are ready to be processed further according to the next two steps.

[2] Crossover. The partially matched crossover is adopted, and some im-

plementation details follow. Firstly, $\lfloor size/2 \rfloor$ couples of permutations are formed at random. The symbol $\lfloor . \rfloor$ denotes the floor function, and, if $size$ is odd, the permutation which is not paired does not undergo the crossover. Secondly, each pair is examined in turn. If $U(0,1) < p_c$, where p_c is a pre-specified probability value, then the pair undergoes the crossover according to the following steps. (*i*) Two different positive integers a and b, say, are chosen at random in the range from 1 through k, and let $a \leq b$. (*ii*) The labels which, in the two permutations, stay in places $a, a+1, \ldots, b$ are exchanged from a permutation into the other one. (*iii*) Such exchange is to define a map which turns useful to solve problems that may possibly arise as far as the validity of the resulting permutations is concerned.

[3] Mutation. Each label of each permutation is taken into account in turn. Let i_j be the label of the permutation under consideration. If $U(0,1) < p_m$, where p_m is a pre-specified mutation probability, then an integer, ℓ say, different from j is selected at random in the interval $(1,k)$, and the labels i_j and i_ℓ are exchanged.

The elitist strategy is adopted, i.e. the best permutation, according to the objective function value, is always maintained into the current population, possibly by discarding the poorest one, so as to keep $size$ unchanged. This case apart, at the end of each iteration the new population, generated by applying the three steps illustrated above, entirely replaces the past one, and is assumed as the current population. The best permutation found into the population at the N-th iteration is taken as final solution.

In addition to the three algorithms described above, a pure random search procedure was implemented for comparison. For a pre-specified number of iterations, a permutation is generated at random, and the objective function value is checked if it exceeds the best found to date. If so, the best value is updated, and the new permutation is recorded. Obviously, such a procedure is expected to yield results worse than that provided by the other stochastic search algorithms, as these latter are believed to be able to search the solution space much more efficiently.

Moreover, for each of the three metaheuristics, a linear code version has been developed. According to this type of encoding device, time series are arbitrarily numbered, so that each integer number in the sequence from 1 through k corresponds uniquely to a time series. Then, the group membership vector is built up, where each entry is the label, preferably an integer number from 1 through g, of the cluster to which the time series belongs (see, e.g. the aforementioned papers by Al-Sultan and Jones and Beltramo). This coding requires the number of clusters g be supplied, and the objective function (4) had better be employed. In particular, as the genetic algorithm assumes only positive objective function, the negative exponential of (4) is computed in this case.

The linear code versions of the algorithms develop in a manner similar to that illustrated in case of the permutation code, and, for tabu search, the implementation closely follows the original Al-Sultan's scheme. The only difference that deserves a special mention is concerned with the genetic algorithm. It resides in that the single-point crossover was used instead of the partially matched one, as this latter applies only to the permutation code.

Finally, the grouping genetic algorithm (Falkenauer, 1998) was tried, because its formulation offers a promising alternative way to handle properly the partitioning time series problem as stated before. A peculiar implementation of the original Falkenauer's algorithm was only needed for the crossover operator. The same bin packing crossover was employed, but the re-assignment procedure was assumed the step [3] of the decoding algorithm displayed in the preceding section.

3. A simulation experiment

Ten sets of time series were generated from either univariate models of the type

$$z_t = x_{t-u} + y_{t-v} + e_t, \tag{6}$$

or vector ARMA models of orders $p = 1$ and $q = 1$. For each time series 300 observations were generated. The first 50 were discarded, as I assumed that the initial values, all set to zero, had afterwards negligible impact on the data. Then, discarding the last 50 observations allowed any shift u and v, either positive or negative, to apply. This way, all time series resulted to have $n = 200$ observations. The algorithm AS183 (Wichmann and Hill, 1982) was used to provide the uniform random numbers in (0,1). The normal standard unit random numbers were provided by the algorithm AS241 (Wichura, 1988). The Cholesky factorization method was used to yield the vector white noise with assigned variance-covariance matrix $\mathbf{\Sigma}$.

For each set, the experiment was replicated 100 times. The sets were partitioned by several procedures, that have been illustrated in the preceding section, and, for further comparison, by using the single linkage method (Gower and Ross, 1969) and the neural network approach (Adorf and Murtagh, 1988).

Each run of each algorithm was programmed in such a way that its stop occurred only when all iterations up to the maximum allowable prespecified number were complete.

The results were evaluated by means of some indicators, for each of which the average over 100 replications was computed. I assumed as such indicators the estimated number of clusters $\hat{g}$, the number of iterations really needed to obtain the final result, $\hat{N}$ say, the corrected Rand index (Hubert and Arabie, 1985), $\hat{R}$ say, as an external criterion to evaluate the adherence of the estimated partition to the true one, and the Pearson correlation $\hat{r}$ between the final objective function value and $\hat{R}$, which is concerned with the validity of the objective function as internal criterion.

The sets of artificial time series were constructed so as to offer a wide range of sources of cross-correlation amongst the series belonging to the same set.

The first set, set 1, included 60 time series forming 6 clusters of 10 series each. Time series were generated according to (6), where both x_t and y_t were low order ARMA univariate models. The white noise was assumed zero-mean standard unit normal, whilst the series e_t was zero-mean normal white noise with variance 2. For each cluster, a specification of (6) was employed, as displayed in Table 1. Each of the 10 time series belonging to the same cluster were shifted by assigning u and v random values from the integer triangular distribution centered on pre-specified values $\bar{u}$ and $\bar{v}$ respectively, and range $(\bar{u} \pm 2)$ and $(\bar{v} \pm 2)$.

Table 1. Univariate time series models for cluster generation

x_t	coeff.s	y_t	coeff.s	z_t	coeff.s
AR(1)	$\phi = -.5$	AR(1)	$\phi = .7$	ARMA(2,2)	$\phi_1 = .2$ $\phi_2 = .35$ $\theta_1 = .1$ $\theta_2 = .143$
AR(1)	$\phi = .5$	AR(1)	$\phi = -.7$	ARMA(2,2)	$\phi_1 = -.2$ $\phi_2 = .35$ $\theta_1 = -.1$ $\theta_2 = .143$
MA(2)	$\theta_1 = .7$ $\theta_2 = -.7$	MA(1)	$\theta = -.5$	ARMA(0,2)	$\theta_1 = .1$ $\theta_2 = -.14$
MA(2)	$\theta_1 = -.7$ $\theta_2 = -.7$	MA(1)	$\theta = .5$	ARMA(0,2)	$\theta_1 = -.1$ $\theta_2 = -.14$
AR(1)	$\phi = -.7$	MA(1)	$\theta = -.7$	ARMA(1,2)	$\phi = -.7$ $\theta_1 = -.6$ $\theta_2 = -.09$
AR(1)	$\phi = .7$	MA(1)	$\theta = .7$	ARMA(1,2)	$\phi = .7$ $\theta_1 = .6$ $\theta_2 = -.09$

The AR parameters are denoted by ϕ , and by θ the MA ones. The standard normal white noise was used to generate both x_t and y_t, whilst the sum model z_t includes the white noise e_t with mean equal to zero and variance equal to 2.

The 10 pairs $(\bar{u}, \bar{v})$ were given the values $(0,0)$, $(10,0)$, $(0,10)$, $(5,5)$, $(5,0)$, $(0,5)$, $(-5,5)$, $(-5,-5)$, $(-10,0)$, and $(0,-10)$. The component time series were chosen in (6) so as the one had large spectral density at low frequencies, and the other at the high ones. So, the spectral density of the sum time series z_t had two local maxima, at 0 and π. As a consequence, the different time shifts, u for x_t, and v for y_t, determined the time series z_t be shifted in different frequency bands. For models in rows

1, 3, and 5, the time series with large spectral density at high frequencies is listed first, and next that with large spectral density at low frequencies. The opposite is true as far as models in rows 2, 4, and 6 are concerned.

The two following sets of time series allow for comparison between the case of clusters all of equal size, and that where a single cluster contains about 60% time series while the remaining ones divide equally amongst the others (see Milligan, 1985). The first of these two sets, that was denoted 1a/e, contains 60 time series and 6 clusters each with 10 time series. Clusters were generated from the 6 models displayed in Table 1. Then, in each cluster, half of the series belonging to it were shifted with shifts centered at $\bar{u} = 0$ and $\bar{v} = 0$, whilst the remaining ones were shifted with $\bar{u} = -5$ and $\bar{v} = 5$. The other set, that was denoted 1a/u, differs only in that the first cluster, generated by the first model in Table 1, includes 40 time series, whilst the other 5 clusters include only 4 series each.

The fourth and fifth sets of time series were structured as well to allow comparison between clusters of equal and unequal size. The first one, that was denoted 2a/e, contains 9 clusters, generated as follows. Models x_t and y_t in lines 1, 2 and 3 of Table 1 were considered, and, from each pair, 21 sum series were generated. Of these, 7 were shifted with $\bar{u} = 0$ and $\bar{v} = 0$, 7 with $\bar{u} = -5$ and $\bar{v} = 5$, and 7 with $\bar{u} = 5$ and $\bar{v} = -5$. A vector white noise $\mathbf{e}_t$ with mean zero and variance-covariance matrix $\mathbf{\Sigma}$ was added to the 7 time series in each cluster. The matrix $\mathbf{\Sigma}$ had all variances equal to 2 on its diagonal, whilst the covariance between any pair of component noise series was taken equal to 1.

This choice ensures that $\mathbf{\Sigma}$ is positive definite and its determinant far away from zero, so that bad estimates are very unlikely to occur. In fact, $\mathbf{\Sigma}$ is a special case of a circulant matrix (see, e.g., Lütkepohl, 1996, p. 113). Its eigenvalues $\lambda_1, \lambda_2, \dots \lambda_{k'}$ are easily computed as

$$\lambda_1 = a + (k' - 1)b, \qquad \lambda_i = a - b, i = 2, \dots k', \tag{7}$$

where I let a denote the common value of the diagonal entries, b the off-diagonal ones, and k' the matrix dimension.

The set 2b/u had the same structure than 2a/e, but its first cluster, based on the first model in Table 1, included 31 time series, and the other 8 clusters included 4 series each. The vector white noises were dimensioned accordingly. In this case too, the variance-covariance matrices had common diagonal entry equal to 2, and all off-diagonal elements equal to 1.

The next time series set, that I called set 2b/e, was given a cluster structure very similar to set 2a/e, but a greater number of smaller clusters was assumed. The number of clusters was 12, each with 5 time series, so that the whole set contained 60 time series. The first 4 models in Table 1 were used, and the same 3 pairs of shifts as before. Each combination between a model and a couple of shifts yielded the basic structure of a univariate time series, that was replicated 5 times. A vector white noise $\mathbf{e}_t$ with variance-covariance matrix $\mathbf{\Sigma}$ was added to the time series belonging to the same cluster.

The seventh set of time series, that was denoted set 3, was formed by allowing the clusters to include series generated by different models. For each cluster, 6 time series were generated, one for each of the 6 models in Table 1, and the vector white noise $\mathbf{e}_t$ with variance-covariance matrix $\mathbf{\Sigma}$ was added as before. Then, each cluster was replicated 9 times, so yielding a set of 54 time series. Shifts were not applied here, as the source of variability within each cluster was taken instead the difference amongst the models. For comparison purpose, a similar set was generated by adding uncorrelated white noise. This latter set 3a was considered as a random set. The algorithms were expected to produce a partition anyway, but in the presence of objective function values worse than that obtained for sets characterized by a genuine cluster structure.

The last two sets of time series were generated by some vector ARMA(1,1) models, one for each cluster. Let k' be the dimension of the model

$$(\mathbf{I} - \mathbf{\Phi}B)\mathbf{z}_t = (\mathbf{I} - \mathbf{\Theta}B)\mathbf{e}_t, \tag{8}$$

where $\mathbf{e}_t$ is a vector white noise with variance-covariance matrix $\mathbf{\Sigma}$.

The first set, that I called set 4a/e, had 12 clusters with 6 time series each. So, the set 4a/e contained 72 time series. The diagonal elements of the matrix $\mathbf{\Sigma}$ were set equal to 2, and to 1 the off-diagonal ones. The matrices $\mathbf{\Phi}$ and $\mathbf{\Theta}$ were given the same special circulant structure, so that the stationarity and invertibility conditions could be easily checked by using (7). For model (8) to be stationary, it suffices that all eigenvalues of

$\mathbf{\Phi}$ be inside the unit circle, and so those of $\mathbf{\Theta}$ for invertibility. The matrix parameters in model (8) are displayed in the left hand side of Table 2 as far as set 4a/e is concerned. The cross-correlation between any pair of time series that belong to the same cluster was yielded by the covariance structure of the white noise $\mathbf{e}_t$ for 5 out of 12 clusters, where the off-diagonal elements of both $\mathbf{\Phi}$ and $\mathbf{\Theta}$ were zero. The time series within the remaining clusters were correlated due to both the white noise covariance structure and the off-diagonal model parameters.

Table 2. Vector ARMA time series models for cluster generation

set	4a/e				4b/e			
	AR		MA		AR		MA	
model	ϕ_d	ϕ_o	θ_d	θ_o	ϕ_d	ϕ_o	θ_d	θ_o
1	0.00	0.00	0.00	0.00	-0.10	0.02	0.10	-0.02
2	0.70	0.00	-0.50	0.00	0.10	-0.02	-0.10	0.02
3	0.50	0.00	-0.70	0.00	0.70	-0.01	-0.50	0.01
4	-0.70	0.00	0.50	-0.00	0.50	-0.01	-0.70	0.01
5	-0.50	0.00	0.70	-0.00	-0.70	0.01	0.50	-0.01
6	0.70	-0.10	-0.70	0.10	-0.50	0.01	0.70	-0.01
7	-0.70	0.10	0.70	-0.10	0.70	-0.10	-0.70	0.10
8	-0.50	0.10	0.50	-0.10	-0.70	0.10	0.70	-0.10
9	0.50	-0.10	-0.70	0.10	0.50	-0.10	-0.70	0.10
10	0.70	-0.10	-0.50	0.10	0.70	-0.10	-0.50	0.10
11	-0.50	0.10	0.70	-0.10	-0.50	0.10	0.70	-0.10
12	-0.70	0.10	0.50	-0.10	-0.70	0.10	0.50	-0.10

The AR parameters in the matrix $\mathbf{\Phi}$ are denoted ϕ_d if on the diagonal, and ϕ_o otherwise. The MA parameters in the matrix $\mathbf{\Theta}$ are denoted θ_d if on the diagonal, and θ_o otherwise.

The last set, that I called 4b/e, had 72 time series and 12 clusters with 6 series each. The time series in each cluster were generated by assigning model (8) the matrix parameters as displayed in the right hand side of Table 2. Unlike the set 4a/e, the variance-covariance matrix $\mathbf{\Sigma}$ was assumed diagonal, with entries equal to 2. So, the cross-correlation between any pair of time series that belong to the same cluster is entirely due to the relationships, defined by the model (8), that the univariate models used for prewhitening cannot account for. The off-diagonal elements in Table 2, in this case, are always different from zero.

4. Experimental results

All 10 time series sets were processed by the algorithms illustrated in section 2, and computation was replicated 100 times. Many parameters were needed to be specified, concerned with the prewhitening and the assessment of the dissimilarity matrix. On the other hand, quite a few parameters were needed as well for each of the algorithms could provide solutions to the partitioning problem.

For prewhitening, an AR model of high order h was estimated for each time series. According to the theoretical AR parameter values, $h = 5$ was selected in all cases. The dissimilarity (or similarity) matrix for each set of time series was computed from the cross-correlation matrix function of the estimated AR(h) models residual series. The maximum lag m for cross-correlation computation was chosen according to the values of the shifts that were applied to the time series in each set. Figures were $m = 12$ for set 1, $m = 7$ for the sets 1a/e, 1a/u, and for all sets of the type 2, and $m = 4$ for the remaining ones. The significance level was chosen $\alpha = 0.05$, and, as a consequence, the threshold value for the cross-correlations turned equal to 0.1386.

The choice of the algorithm parameters was made by following the suggestions found in the aforementioned papers. For simulated annealing, the initial temperature T_0 was 10, the reduction factor was set $\rho = 0.9$, the number n_t of temperature reductions was 10, and the iterations for each given temperature were set to $N = 500$. As far as the tabu search algorithm is concerned, the choice was $mtls = 20$ for the maximum size of the tabu list, $P = 0.95$ for the threshold probability, $nts = 20$ for the number of trial solutions, and the number of iterations was set to 250. The parameter $itmax$, i.e. the maximum number of allowable trials for choosing the current partition in any given iteration, was set to 20, though in the simulation experiment the best trial solution seldom happened to be rejected. For the genetic algorithm, the number of candidate solutions considered in each iteration was $size = 50$, the crossover probability $p_c = 0.6$, the mutation probability $p_m = 0.001$, and the number of iterations was 100. The choice of the number of iterations for the three algorithms was such that, for each replication, approximately 5,000 calls to the objective function evaluation routine were performed in all cases.

The results for sets 1, 1a/e, and 1a/u are reported in Table 3. The number of clusters turns out slightly overestimated for set 1, and the corrected Rand index is smaller for this set than for the other two. As a matter of fact, the set 1 is expected to exhibit more difficulties because each series in the same cluster was given markedly different pairs of shifts. On the contrary, only two kinds of shifts were applied to the time series in the sets 1a/e and 1a/u, and the recovering of the cluster structure is almost perfect, with respect both to the number of clusters and the corrected Rand index.

According to the stochastic nature of the algorithms, a great variability is displayed as far as the number of iterations is concerned. In spite of this, the standard deviation of the other estimated indicators are very small, and the results turn very similar over the replications. The final objective function values depend on the number of cross-correlation absolute values greater than the threshold $c(\alpha)$, and on the presence of a cluster with large size. The cross-correlation matrix function may differ considerably from a replication to another, so that the correlation between the external criterion and the internal one may turn rather small. However, what really matters is the sign of the relationship, which turns out correctly positive as expected.

As far as the comparison between the performance of the algorithms in case of clusters of equal and unequal size is concerned, the results displayed in Table 3 show that the presence of a large cluster does not affect the procedures substantially, being only the figures of the corrected Rand index slightly worse in the latter case. The objective function values are much greater than that from the equal size case, but only because, if a cluster is large, much more similarity terms are summed.

Table 3. Partitioning results for the time series sets of the type 1

1	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	6.7 (.53)	.62 (.10)	41.12 (2.46)	0.47	3,050 (1379)	3,050
t.s.	6.1 (.31)	.87 (.13)	46.13 (1.74)	0.29	144 (69)	2,880
g.a.	6.5 (.56)	.71 (.12)	43.30 (2.52)	0.45	81 (20)	4,050
1a/e	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	6.1 (.31)	.99 (.02)	44.95 (1.53)	0.44	2,812 (1371)	2,812
t.s.	6.1 (.27)	.997 (.01)	45.30 (1.45)	0.35	52 (50)	1,040
g.a.	6.1 (.27)	.99 (.03)	44.90 (1.66)	0.19	41 (26)	2,050
1a/u	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	5.9 (.49)	.98 (.03)	130.2 (9.0)	0.53	2,089 (1467)	2,089
t.s.	5.9 (.44)	.98 (.02)	130.7 (8.96)	0.39	98 (68)	1,960
g.a.	6.0 (.47)	.98 (.02)	131.25 (9.25)	0.35	52 (30)	2,600

The algorithms are all based on the permutation code, s.a. stands for simulated annealing, t.s. for tabu search, and g.a. for genetic algorithm. The average estimated number of clusters over 100 replications is $\hat{g}$, the corrected Rand index $\hat{R}$, the final objective function $\hat{f}^+$, and $\hat{N}$ the iterations actually needed to obtain the final result. The standard deviation of the estimates is enclosed in parentheses. The Pearson correlation between $\hat{R}$ and $\hat{f}^+$ is $\hat{r}$, and N_{call} the average calls to f^+.

Comparison amongst the methods favors the tabu search algorithm, no matter what indicator is taken into account.

In Table 4 the sets of the type 2 are considered. In this case, the cross-correlations turn out to be smaller than for the time series of the previous sets. Evidence of this circumstance is given by the fact that the objective function values, which are directly related to the cross-correlation absolute values, are much smaller than that reported in Table 3. So, the recovery of the true cluster structure is rather difficult, because most estimated cross-correlations are only slightly above the pre-specified threshold $c(\alpha)$.

Table 4. Partitioning results for the time series sets of the type 2

2a/e	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	11.3 (.66)	.28 (.08)	8.7 (.49)	0.48	2,337 (1,362)	2,337
t.s.	10.5 (.73)	.42 (.13)	9.99 (.78)	0.61	184 (56)	3,680
g.a.	11.2 (.74)	.31 (.09)	8.93 (.66)	0.54	71 (25)	3,550
2a/u	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	9.6 (.84)	.49 (.12)	20.0 (5.13)	0.95	2,766 (1,336)	2,766
t.s.	8.9 (.78)	.52 (.13)	21.4 (5.8)	0.95	165 (58)	3,300
g.a.	9.5 (.81)	.50 (.11)	20.0 (5.18)	0.95	73 (21)	3,650
2b/e	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	11.2 (.58)	.15 (.06)	7.2 (.31)	0.13	2,953 (1,415)	2,953
t.s.	10.3 (.72)	.22 (.07)	8.0 (.41)	0.19	176 (60)	3,520
g.a.	11.3 (.63)	.17 (.05)	7.3 (.40)	0.25	72 (24)	3,600

Moreover, confusion may arise as some estimated cross-correlations may turn by chance greater than the threshold, whilst others correspond to time series that actually belong to the same cluster. This circumstance reflects through the poor accordance between the estimated partition and the true one. Nonetheless, though the number of clusters looks overestimated in case of the set 2a/e, and, on the contrary, underestimated in case of the set 2b/e, the overall results may be considered rather satisfying. The cluster structure appear to be correctly recovered for most time series. For set 2a/e, the estimated cross-correlations are possibly too low, so that aggregation cannot take place because the condition that defines a group, based on (1), is not fulfilled. In the set 2b/e, the estimated cross-correlations appear not so low, but most values are close to the threshold $c(\alpha)$. As a consequence, more time series than expected are put together in the same cluster.

As for the type 1 time series, in this case too the standard deviation of the number of iterations actually needed to obtain the final result is large, but all remaining indicators exhibit very small variability over the replications. As far as the comparison between clusters of equal and unequal size is concerned, these latter are even better recovered than the former ones. This may happen because the indicators are influenced by quite a few correct assignments of time series to the first cluster, which is the largest one. As noted for the type 1 series, the objective function values for the set 2a/u are greater than those for the set 2a/e, due to the peculiar formulation of the objective function.

According to all indicators, the overall performance of the tabu search algorithm encompasses that of both the simulated annealing and genetic algorithm. Unlike the type 1 series, however, for the time series of the type 2 more iterations are needed for tabu search to obtain the final result.

The results concerned with the set 3 of time series are displayed in Table 5. No shifts are applied in this case, and the variability within the same cluster is due to the different models from which the time series are generated.

Table 5. Partitioning results for the time series set 3

3	$\hat{g}$	$\hat{R}$	$\hat{f}^{+}$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	13.3 (.80)	.49 (.10)	3.75 (.39)	0.65	2,600 (1,329)	2,600
t.s.	12.4 (.79)	.62 (.13)	4.27 (.52)	0.73	172 (56)	3,440
g.a.	12.5 (.94)	.54 (.12)	4.00 (.53)	0.82	76 (20)	3,800

This kind of perturbation seems to produce less severe difficulties to the clustering procedures. It looks that cross-correlations above the threshold between pair of time series belonging to different clusters are very unlikely to occur. Nevertheless, the estimated cross-correlations between pairs of time series belonging to the same clusters happen often to be very low. As a consequence, many clusters appear split in two. This circumstance may explain the overestimation of the number of clusters, between 13 and 15 on the average, instead of 9. On the other hand, the clusters recovered by the algorithms look small and approximately of equal size, so that the objective function values are small too. However, the figures of the final objective function values are approximately twice those computed for the random set 3a. This ensures that the estimated partition reflects the genuine cluster structure.

In Table 6 results concerned with the time series of the type 4 are displayed. The same large variability of the number of iterations needed to obtain the final solution is observed as before, whilst the other indicators show small standard deviation. The clustering procedures yield different performances for the two sets 4a/e and 4b/e. Whilst the cluster structure of the former set looks easily recovered, that of the latter set suffers of the same drawback than set 3. It happens often, in case of the set 4b/e, that a pair of time series, that belong to the same cluster, yet have low cross-correlation absolute values. As a consequence, the number of clusters is overestimated. Nonetheless, most time series are correctly joined, though into clusters smaller than expected. For this reason, the final objective function values computed for set 4b/e are smaller than those computed for the set 4a/e.

Table 6. Partitioning results for the time series sets of the type 4

4a/e	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	13.3 (.81)	.77 (.06)	31.4 (2.17)	0.74	2,887 (1,370)	2,887
t.s.	12.0 (.31)	.98 (.03)	38.2 (1.22)	0.35	136 (59)	2,720
g.a.	12.7 (.72)	.87 (.07)	34.6 (2.21)	0.80	80 (17)	4,000
4b/e	$\hat{g}$	$\hat{R}$	$\hat{f}^+$	$\hat{r}$	$\hat{N}$	N_{call}
s.a.	18.0 (.99)	.36 (.05)	5.8 (.47)	0.53	2,778 (1,491)	2,778
t.s.	16.6 (.88)	.52 (.05)	7.3 (.52)	0.37	193 (46)	3,860
g.a.	17.4 (.99)	.44 (.06)	6.6 (.65)	0.66	85 (14)	4,250

It has to be noted that the incorrect recover of the true number of clusters in set 4b/e does not seem to depend on some inadequacy of the algorithms. In fact, the tabu search algorithm, for instance, even if allowed to run 1,000 iterations, and $nts = 50$, still provides 16.3 as the average estimated number of clusters, while the corrected Rand index only increases from .52 to .54. The genetic algorithm too exhibits a similar behavior. If the maximum allowable number of iterations is set to 1,000, then the average estimated number of groups over 100 replications decreases only to 16.53, while the average corrected Rand index increases to 0.52. Moreover, the single linkage method, within 1,720 iterations, is able to provide a solution with 12 clusters. Nevertheless, the corrected Rand index turns equal to 0.20, a figure less than that obtained from the same single linkage method in 1,000 iterations. This seems to indicate that too many estimated cross-correlations do not fulfill (1), and the expected number of clusters may be obtained only by overruling such inequality. The group membership, however, may be expected to turn often incorrect.

Each one of the artificial sets of time series was processed as well by the pure stochastic search clustering algorithm, as illustrated in section 2. Results were found not better, though very close indeed in some cases, than that obtained by the one which, amongst the metaheuristic algorithms, yielded the poorest performance. This seems to indicate that the permutation code, and the decoding procedure, turned useful to drive the stochastic procedure towards the correct solution. Significant improvement was further gained by applying the searching devices peculiar to each of the three algorithms based on metaheuristic methods.

The linear code versions of the simulated annealing, tabu search and genetic algorithm were not able to provide results comparable with those displayed in the tables 3-6. Their objective function evaluation routine is faster than permutation code. Nonetheless, increasing the number of iterations, even beyond the execution time needed to the permutation code methods to perform 5,000 calls to the objective function evaluation routine, did not seem to produce any substantial advantage. For instance, the linear code tabu search algorithm, if allowed to take 50,000 calls to the objective function evaluation routine, produced 0.72 as best average corrected Rand index for set 1a/e, whilst the corresponding figure provided by the permutation code tabu search algorithm was 0.997. Moreover, running the 100 replications of such experiment with the linear code and 50,000 calls took 7 minutes and 20 seconds, whilst running the same 100 replications with the permutation code and 5,000 calls took 3 minutes and 59 seconds. Such elapsed time figures refer to a CPU speed 600 MHz, and total physical memory 256 MB. Comparisons between linear and permutation code algorithms yielded similar results for the other time series sets as well.

Let me now consider the single linkage method and the neural network approach. In both cases, for the time series sets 1, 1a/e, 1a/u, and 4a/e, where the clusters are well separated, performances were found comparable, and even slightly better than that obtained from the metaheuristic methods and permutation code. In other cases, the neural network approach could take advantage of the fact that it was supplied with the true number of clusters. This circumstance was particularly apparent in the presence of a large number of clusters, as for the set 2b/e. Otherwise, the performances of these two procedures were not as good as those provided by the metaheuristic methods.

Finally, the grouping genetic algorithm performed as good as the linear code versions of the simulated annealing, tabu search and genetic algorithm methods. Moreover, with respect to the latter one, it yielded better results, no matter what time series set was considered. When compared to the metaheuristic methods, however, even increasing the number of iterations, up to 10,000 for instance, yet the estimated indicator figures turned not so close to the true ones. The clustering problem, indeed, is possibly stated in such a way that, as observed already, the permutation code and the decoding procedure play an important role as far as the correct cluster recovery is concerned.

5. Concluding remarks

I considered the problem of finding the best partition of a set of time series, according to the cross-correlations estimated from the prewhitened series. The dissimilarity matrix entries were computed to include the cross-correlations up to a pre-specified lag. The k-min cluster criterion was a natural choice, as such indexes were found not to meet the requirements of the Euclidean distance. Furthermore, the clusters were constrained to include only time series such that their pairwise cross-correlation absolute values exceeded a given threshold. This latter was taken to depend on the selection of a significance level.

As solutions were expected to form a large discrete space, I introduced three algorithms based on metaheuristic methods and permutation code with a special decoding procedure. A simulation experiment on ten sets of artificial time series proved that the metaheuristic algorithms were able to yield the best results, compared to alternative methods such as the pure random search, the single linkage method, the neural network approach, the linear code versions of the same metaheuristics, and the grouping genetic algorithm. The series were generated from low order univariate and vector ARMA models. Their parameters were chosen so as to result well inside the stationarity and invertibility regions, to avoid, as far as possible, the occurrence of estimates which fell by chance outside such limits.

Amongst the metaheuristic methods, i.e. simulated annealing, tabu search, and the genetic algorithm, the tabu search procedure was found to perform better, according to the estimated number of clusters, the corrected Rand index, and the final value of the objective function.

In some cases a marked difference between the true and the estimated number of clusters was found. As a matter of fact, this seemed strictly depend on the constraint that series had to be correlated more than a given threshold to be allowed to belong to the same cluster. If the cross-correlation structure were weak, then the algorithms could not be expected to indicate the correct number of clusters. Any of the procedures would point instead at a number greater than that used for generating the time series sets. This circumstance does not appear a too severe drawback, however, as this kind of error may be amended, if needed, by deeper examining the estimated clusters, and merging some of them, possibly overruling the threshold condition. On the other hand, there is not a close connection between the way by which the time series are generated and the cross-correlation absolute values. Once the significance level has been selected, then the partition takes place at that level, irrespective whether the simulation procedure be actually in accordance with it.

Acknowledgments: This paper is supported by MURST 1998: ”Statistical Models for Time Series Analysis.” An earlier version of this article was presented at the Meeting of the MURST 1998 research group, October 27, 2000, Dipartimento di Scienze Statistiche, Universita’ degli Studi di Napoli ”Federico II.” Comments of the Meeting participants, particularly Francesco Battaglia, Marcella Corduas and Emma Sarno were very helpful. However, responsibility for any errors, or errant interpretation of other works, is solely my own.

References

Adorf, H.-M., Murtagh, F. (1988) Clustering based on neural network processing, *Compstat 1988*, Physica-Verlag, 239-244.

Al-Sultan, K. S. (1995) A tabu search approach to the clustering problem, *Pattern Recognition*, 28, 1443-1451.

Bohte, Z., Čepar, D., Košmelj, K. (1980) Clustering of time series, *Compstat 1980*, Physica-Verlag, 587-593.

Bollobás, B., Das, G., Gunopulos, D., Mannila, H. (1997) Time series similarity problems and well-separated geometric sets, *13th Annual ACM Symposium on Computational Geometry*, 454-456.
http://www.cs.helsinki.fi/∼mannila/postscripts/compgeom.ps

Brockwell, P. J., Davis, R. A. (1996) *Introduction to Time Series and Forecasting*, Springer.

Brooks, S. P. (1995) A hybrid optimization algorithm, *Applied Statistics*, 44, 530-552.

Corduas, M. (2000) La metrica Autoregressiva tra modelli ARIMA: una procedura in linguaggio GAUSS, *Quaderni di Statistica*, 2, 1-37.

Falkenauer, E. (1998) *Genetic Algorithms and Grouping Problems*, Wiley.

Gower, J. C., Ross, G. J. S. (1969) Minimum spanning trees and single linkage cluster analysis, *Applied Statistics*, 18, 54-64.

Hubert, L., Arabie, P. (1985) Comparing partitions, *Journal of Classification*, 2, 193-218.

Jones, D. R., Beltramo, M. A. (1991) Solving partitioning problems with genetic algorithms, *IV International Conference on GA* (Belew, R. K. and Booker, L. B., eds.), 442-449.

Kakizawa, Y., Shumway, R. H., Taniguchi, M. (1998) Discrimination and clustering for multivariate time series, *Journal of the American Statistical Association*, 93, 328-340.

Lütkepohl, H. (1996) *Handbook of Matrices*, Wiley.

Maharaj, E. A. (2000) Clusters of time series, *Journal of Classification*, 17, 297-314.

Maulik, U., Bandyopadhyay, S. (2000) Genetic algorithm-based clustering technique, *Pattern Recognition*, 33, 1455-1465.

Milligan, G. W. (1985) An algorithm for generating artificial test clusters, *Psychometrika*, 50, 123-127.

Piccolo, D. (1990) A distance measure for classifying ARIMA models, *Journal of Time Series Analysis*, 11, 153-164.

Sahni, S., Gonzalez, T. (1976) P-Complete approximation problems, *Journal of the Association for Computing Machinery*, 23, 555-565.

Selim, S. Z., Al-Sultan, K. S. (1991) A simulated annealing algorithm for the clustering problem, *Pattern Recognition*, 24, 1003-1008.

Shumway, R. H., Unger, A. N. (1974) Linear discriminant functions for stationary time series, *Journal of the American Statistical Association*, 69, 948-956.

Sung, C. S., Jin, H. W. (2000) A tabu-search-based heuristic for clustering, *Pattern Recognition*, 33, 849-858.

Tseng, L. Y., Yang, S. B. (2001) A genetic approach to the automatic clustering problem, *Pattern Recognition*, 34, 415-424.

Wichmann, B. A., Hill, I. D. (1982) An efficient and portable pseudorandom number generator, *Applied Statistics*, 31, 188-190; correction, 33 (1984), 123.

Wichura, M. J. (1988) The percentage points of the normal distribution, *Applied Statistics*, 37, 477-484.

Zani, S. (1983) Osservazioni sulle serie storiche multiple e l’analisi dei gruppi, *Analisi Moderna delle Serie Storiche* (Piccolo D., ed.), 263-274.