Clustering of Activity Patterns Using Genetic Algorithms

Ondrej Pribyl
Pennsylvania Transportation Institute & Pennsylvania State University
201 Transportation Research Bldg., University Park, PA 16802
e-mail: oxp112@psu.edu; phone:(814)863-5493

Abstract

Finding groups of individuals with similar activity patterns (a sequence of activities within a given time period, usually 24 hours) has become an important issue in models of activity-based approaches to travel demand analysis. This knowledge is critical to many activity-based models, and it aids our understanding of activity/travel behavior. This paper aims to develop a methodology for the clustering of these patterns. There is a large number of well-known clustering algorithms, such as hierarchical clustering, or k-means clustering (which belong to the class of partitioning algorithm). However, these algorithms cannot be used to cluster categorical data, so they do not suit the problem of clustering of activity patterns well. Several other heuristics have been developed to overcome this problem. The k-medoids algorithm, described in this paper, is a modification of the k-means algorithm with respect to categorical data. However, similar to the k-means algorithm, the k-medoids algorithm can converge to local optima.

This paper approaches the medoids-based formulation of clustering problem using genetic algorithms (GAs), a probabilistic search algorithm that simulates natural evolution. The main objective of this paper is to develop a robust algorithm that suits the problem of clustering of activity patterns and to demonstrate and discuss its properties.

1. INTRODUCTION

Finding groups of individuals with similar activity patterns (a sequence of activities within a given time period, usually 24 hours) has become an important issue in models of activity-based approaches to travel demand analysis. This knowledge is critical to many activity-based models, and it aids our understanding of activity/travel behavior.

Previous research (Pas 1983, Ma 1997) gave us evidence that people with similar socio-demographic characteristics have also often similar schedules.

One direction of research focuses on the microsimulation of the observed activity patterns (Kulkarni and McNally 2000). An essential step in their modeling effort is to find an average or typical schedule that represents the activity/travel behavior of these individuals. The ultimate goal of this project, then, is to find a relatively small set of activity patterns that would represent all observed patterns in given data set.

The problem as stated belongs to the so-called *subset selection problem* that is interpreted as a special case of *cluster analysis*. The number of different partitions of *N* objects into *K* clusters, *p(N,K)*, is extremely large. Everitt et al. (2001) provides an equation of how to compute *p(N,K)*. To illustrate the complexity, let's look at the following examples: $p(10,3)=9330$, and $p(50,4)$? 5.3×10^{67}. In our case, the size of the data set, N, as well as the number of clusters, is expected to be even higher. For this reason we are not able to evaluate all possible combinations.

The data used in this paper are of the categorical type (as will be shown in section two). Many well-known clustering algorithms, such as hierarchical clustering and k-means clustering (Everitt et al. 2001), are not suitable for this type of data. They are based on Euclidean distances, and they need to find an average of several patterns. In our case, however, there is not a simple and realistic way to find the average of several schedules.

Several heuristics have been developed to overcome this problem. For example, Kaufman and Rousseeuw (1990) described a modification of the k-means algorithm to suit categorical data. This algorithm is called *medoid-based clustering*, and it is described in greater detail in this paper.

Similar to the k-means algorithm, the medoid-based algorithm can converge to local optima. For this reason this paper uses an alternate heuristic for the same formulation of the problem. It is based on the principles of genetic algorithms (GAs), probabilistic search algorithms that simulate natural evolution.

The main objective of this paper is to develop a robust algorithm that suits the problem of clustering activity patterns, and to demonstrate its properties. The algorithm should suit relatively large data sets to enable its application to real-world problems.

Approaches based on genetic algorithms have usually many parameters that need to be set, such as the size of population, number of runs, probability of mutation, and others. The performance of the algorithms is often very sensitive to the setting of these parameters, but at the same time no general guidance about the parameter setting exists, since it is problem specific. This paper shows the performance of the developed algorithm for different settings of parameters. The objective is to provide recommendations concerning parameter settings that would enable the broader use of this tool without the need for further in-depth study.

2. THE DATA

Knowledge of the data set's structure would help to evaluate the algorithm and also it would enable more truthful discussion of its properties. For this reason, artificially generated data rather than observed data are used in this preliminary phase of the project.
Before we describe the data set used for this analysis, the representation of activity patterns used in this paper must be discussed.

2.1. REPRESENTATION OF ACTIVITY PATTERNS

The proposed algorithm will be evaluated on its ability to cluster activity patterns. An essential feature for forming the activity patterns is the type of each activity. Though there are many different activity types that an individual can perform, only a limited number of activities will be considered in this work, for simplicity. The activities considered are: ***Home*** (everything that is an in-home activity), ***Work*** (work, work related or school), ***M**aintenance* (dining out, shopping and others) and ***D**iscretionary* (social events, recreation, visiting friends and others)[1].
Within this project we use a discretized representation of the activity patterns. For each individual in the sample, we look at his activity participation in regular time intervals (10 minutes in this project) and record it. It means that the whole 24 hour long activity pattern consists of 144 values. Each of the values corresponds to an activity type. Integer values from the range 1 to 4 are used to represent given activity types; Home(1),Work(2), Maintenance(3), and Discretionary(4). It must be clearly stated that these values do not have real numerical meaning. The activity types are clearly of a categorical nature.
For example, we can see ten activity patterns in Figure 1. They correspond to different types of activity patterns as described in the next chapter. The x-axis in this figure represents time per day (hours) and the y-axis represents the four activity types as described above. For example, the first pattern shows the schedule of an individual who left from home to work at about 9:30 am, he went for lunch at about 11 am (maintenance activity) and returned back to work at about noon. This individual returned back home shortly before 6 pm. The other patterns in Figure 1 can be understood similarly.

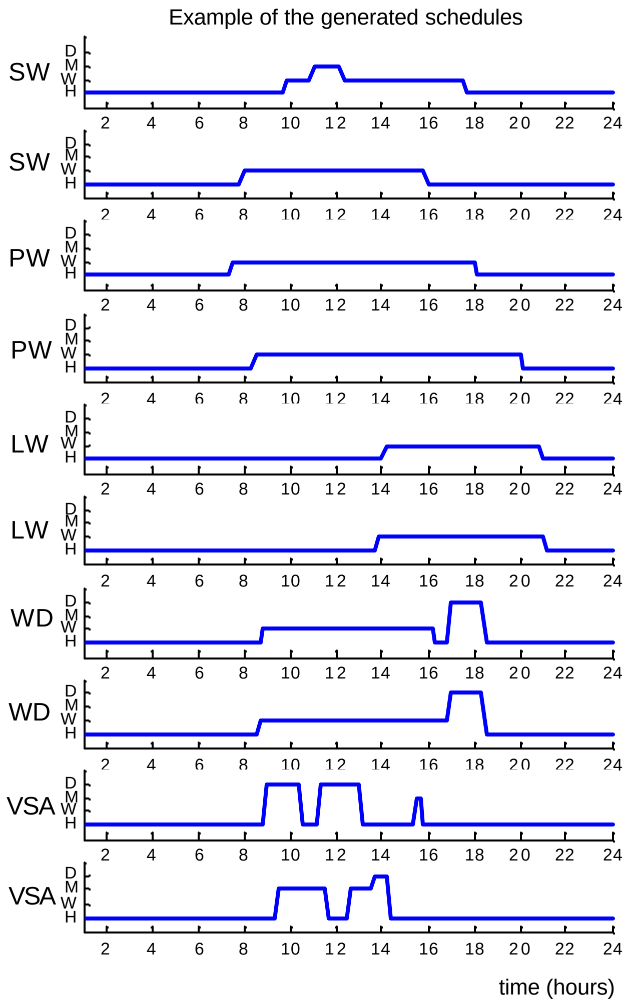

Figure 1: Examples of generated activity patterns. There are two patterns of each type; standard work (SW), power work (PW), late work (LW), work discretionary (WD), and various short activities (VSA), as described in chapter 2.2.

2.2. SYNTHETIC DATA

The data used in this paper are generated artificially so their structure is known, but at the same time they should realistically represent the observed patterns. In order to meet these objectives, five different groups of schedules, using results reported by Kulkarni and McNally (2000) were generated in this paper. Examples of two schedules from each group are shown in Figure 1.
The first group is called *standard work* (SW). As the name implies, people in this group work for about 8 hours. Their work starts sometimes in the morning (between 7 am and 9 am). Some of these individuals go

[1] The algorithm developed in this project should be general enough to enable later extension to more activity types. We might want to distinguish if the four activities were undertaken alone or joint with some other family member[1]. This step will increase the number of activity types up to ten.

for lunch at about noon (maintenance activity up to one hour long).

The second group, called *power work* (PW), contains of people whose schedules are similar to those in the previous group, only their work activity is in average two hours longer, for example from 8 am to 6 pm.

The next group is called *late work* (LW). People in this group also have, on average, an 8 hour-long working day (similar to SW), but their work starts in the afternoon, around 2 pm.

The fourth group is called *work discretionary* (WD). People in this group have standard work starting in the morning, but they participate in some discretionary activity after work. Some people in this group go to the discretionary activity directly from work, some people first stop at home.

The last group is called *various short activities* (VSA). People in this group participate in several activities (up to five per day). Their type varies randomly among work, maintenance and discretionary type.

The code enables the user to set the number of schedules generated in each group, as well as set all probabilities (for example probability to go for lunch for the first two groups) and the starting and ending times for each activity. The data, as generated, are rather similar to the observed schedules of individuals. However, this brings also a challenge to the algorithm. The groups (or clusters) of the data are overlapping. It is not true that schedules in one generated group are the closest to each other based on the used dissimilarity measure. Unfortunately, this is also the case of any observed data.

3. MEDOID BASED CLUSTERING

The method in this paper is based on so called *k-medoid clustering* (Kaufman and Rousseeuw 1990, Huang 1997). Consider a set S={S_1, …, S_N} of N objects (in our case activity patterns). Each object S_i is a vector, containing of L integer values describing an activity type at particular time instant. Our objective is to find K objects, $K<<N$, which represent all objects in the data set. The remaining objects are then assigned to the nearest representative object, using a given dissimilarity measure.

A simple example of 12 two-dimensional objects is shown in Figure 2. The arrows in this figure point to objects that represent each of the clusters. These representative objects are called *medoids*, because we expect them to be located in the center of each cluster. Each object in the data set belongs exactly to one medoid. All objects belonging to the same medoid form a *cluster*. The objects in each cluster are clearly more similar (based on given dissimilarity measure) to each other than to objects in any other group.

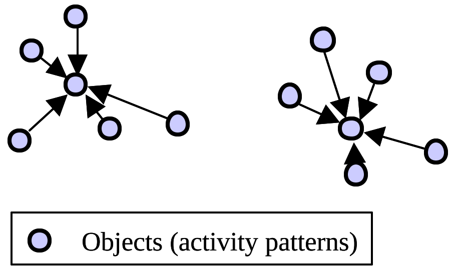

Figure 2: Suggested principle of k-medoid clustering approach

The developed algorithm requires the number of clusters, K, to be known. In order to find the optimal number of clusters, the algorithm will be repeated for different numbers of clusters. The optimal number will be chosen off-line, after the clustering problem for different K will be performed.

Mathematically, we are looking for a set of K objects that minimize the following objective function:

$$D_K = \sum_{k=1}^{K} \sum_{i \; assigned\_to\_medoid \; k} d(k, S_i) \qquad (1)$$

D_K is the sum of dissimilarities of the particular medoids to all other members of the same cluster. In case the arrows in Figure 2 represent the dissimilarity among objects, the objective function D_K can be understood as sum of the length of all the arrows. We are seeking the medoids so that this sum is minimal.

3.1. DISSIMILARITY MEASURES

A very important issue in any clustering algorithm is definition of dissimilarity between two objects. We cannot use common Euclidean distance, because we deal with objects of a categorical type (activity types). The distance measure that used in this paper is defined in Huang (1997) as follows: The dissimilarity measure between two objects S_1, and S_2 reflects the total number of mismatches of activity types at a corresponding time index. This can be written as:

$$d(S_1, S_2) = \sum_{j=1}^{ls} \delta(S_1(j), S_2(j)), \qquad (2)$$

where

$$\delta(S_1(j), S_2(j)) = \begin{cases} 0 & (S_1(j) = S_2(j)) \\ 1 & (S_1(j) \neq S_2(j)) \end{cases}$$

and ls denotes the length of the schedule (in our case ls equals to 144). Therefore, the value of the dissimilarity measure between two patterns can range in our case from 0 (the two patterns are the same) to 144 (totally different).

4. GA BASED APPROACH TO K-MEDOID CLUSTERING

Several different representations have been used in the literature (for an overview see Lozano and Larranaga 1996, or Estivill-Castro and Murray 1997, Moraczewski et. al. 1995, Maulik and Bandyopadhyay 2000, and others). The most suitable representation for the given problem (considering the size of the problem and also its categorical character) can be adapted, for example, from Lucasius, et al. (1993).

Each chromosome is a vector of the length K (number of clusters), and every element is obtained from a uniform distribution in the range (1,N). The i^{th} value is an index of the i^{th} median. For example, if K equals 6, and the medoids are patterns with numbers 4, 23, 43, 54, 120, and 130, the result chromosome is, for example, the following vector [4, 120, 43, 23, 54, 130] (there is more than just one representation of the same solution). The objective function of each chromosome is computed based on equation (1) in this paper.

One advantage of integer representation is that we do not have to convert chromosomes into phenotypes before computing value of the objective function, and so we save computational time.

Two different algorithms have been developed. *Model 1* is *an integer coded GA with (?+?) binary tournament selection* (the symbols ? and ? are common in the GA related literature and represent the parent and children population respectively).

Model 2 is a modification of model 1. It modifies the selection operator to maintain more diversity in the population. The principle of the selection operator is briefly described in this paragraph and also shown in the provided pseudo-code. Given two randomly selected parents, we apply a recombination and mutation operator and so produce two offspring. The selection operator is applied to the set of both parents as well as both offspring. Two best chromosomes are deterministically selected from this set. The pseudo-code for both models is provided in the section below.

Notation:

GN	... number of generations
NP	... size of population
p_i	... parent i
c_i	... offspring i
P_{mB}	... probability of in-built mutation
P(t)	... population at time t

Pseudo-code for Model 1:

```
for t = 1: GN,
      P(t) ← shuffle (P(t-1))
     for j=1:NP/2-1,
       p1 = $P_{2j+1}$(t)
       p2 = $P_{2j+2}$(t)
       {$c_1$,$c_2$} ← recombine ($p_1$,$p_2$)
       {$c_1$',$c_2$'} ← mutate ($c_1$,$c_2$)
       append {$c_1$',$c_2$'} to matrix C
       end
   Create matrix Q that combines both,
   parents and offsprings {P;C}
   Binary tournament selection (matrix Q)
end
```

Pseudo-code for Model 2:

```
for t = 1: GN,
   P(t) ← shuffle (P(t-1))
   for j=1:NP/2-1,
      p1 = $P_{2j+1}$(t)
      p2 = $P_{2j+2}$(t)
      {$c_1$,$c_2$} ← recombine ($p_1$,$p_2$)
      {$c_1$',$c_2$'} ← mutate($c_1$,$c_2$)
      select 2 best individuals from the
      set { $p_1$, $p_2$, $c_1$',$c_2$'} and place them
      to the new population P(t+1)
   end
end
```

4.1.1. Population Management

In order to improve the performance of the algorithm, a method of multi-population approach similar to Reed (2002) is used. The algorithm performs several iterations. The population size is doubled in every iteration, by adding a randomly initiated population. The inserted population has the same size as the current population.

This doubling of population aims to decrease the computational time, it helps to prevent premature convergence, and also it aims to decrease the sensitivity of the algorithm to the setting of its parameters, such as size of population, number of runs, or the random number seed used.

Figure 3 shows an example of progress of the best and average fitness function for each generation. In this example there are 300 objects in the data set (N). The algorithm performs forty runs (GN) for each population size (NP = 10, 20, 40, and 80).

The points of injection (every GN = 40 populations) show a large increase in the average fitness function. This reflects the fact that the diversity in the population increases by inserting random individuals. This also often leads to improvement of the best fitness function shortly after insertion of new random individuals.

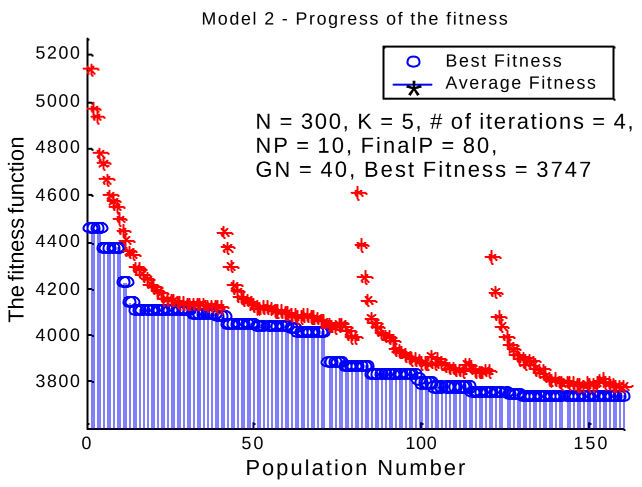

Figure 3: The progress of the best and average fitness for DATA 300 and four iterations (3 injections) - example.

4.2. DETAILED DESCRIPTION OF THE ALGORITHMS

The whole initialization phase, as well as the recombination and mutation operator, is the same for both developed models. The main difference is in the management of the population and in the selection process. The following paragraphs explain the elementary building blocks of the algorithm in more detail.

Create initial population.

The size of each population is denoted *NP*. The initial population is created randomly. Each value is an integer number in the range of 1 to *N* (number of objects in the data set). It is ensured that the values within each chromosome are unique (not repeated).

D_MX recombination with in-built mutation

Several recombination operators were considered in this project. The use of the standard operators, such as uniform crossover, is problematic, because they produce infeasible offspring (the values in each chromosome cannot be repeated). The Random Respectful Recombination (R^3) as described by Estivill-Castro and Murray 1997 was also considered. Its principle is based on the theory of building blocks (De Jong et al. 1997). If a certain value is present in both parents, it must be present also in both children. The rest of the algorithm is the same as for uniform crossover (however, feasibility of offspring is ensured by this algorithm).

The author decided not to use this operator, because it might decrease the diversity of offspring and lead to premature convergence to a suboptimal solution. This is true especially in case of clustering, when the length of each chromosome is short.

The recombination operator, ***D_MX*** (*Mixed Subset Recombination*), used in this project was slightly modified from Lucasius et al. (1993). The “D” in its names reflects the fact that it is applied “directly” to the phenotype, rather than to the binary representation (genotype). The operator, when applied to two parent strings (P_1, P_2) produces two offspring (C_1, C_2), and works as follows:

1) Mix P1 and P2:
    a. Append copy of P_1 at the end of P_2, and form a vector Q.
    b. Randomly scramble the elements in Q.
2) Apply the built-in mutation to all elements in Q (this step is different from Lucasius et al. (1993), who applies the built-in mutation only to the first k elements). If an element is chosen (with probability P_{mB}), its value is replaced by some other random value, that points to some other medoid (random integer number in the range 1 to number of objects in the data set).
3) Randomly scramble elements in Q again.
4) Create first offspring, C_1, by copying k elements from Q into C_1, starting at the leftmost element. Elements that are already in C1 are skipped (to ensure the feasibility of offspring).
5) Similarly, create second offspring, C_2, by copying k elements from Q into C_2, this time starting from the rightmost element.

With a certain level of simplification, the operator can be considered to be a uniform crossover with in-built mutation that ensures generating of feasible offspring (feasibility for this application was discussed above).

Selection ***Model 1***

In the previous literature, most of the researchers used roulette wheel selection (Lucasius et al. 1993). In order to improve its performance, a linear scaling had to be applied to the objective function. For the purpose of this project, we decided to apply the standard *binary tournament selection*. It is well supported by theory and previous work has shown its superiority to roulette wheel selection (for example, a “super solution” - solution with much higher fitness function than the rest of the population - would occupy most of the roulette wheel area, so the next generation would be occupied mostly by this solution (for more detailed comparison of different selection operators see, for example, De Jong 1997).

Selection ***Model 2***

The two best solutions are deterministically selected from a set of parents and their offspring {p_1, p_2, c_1, c_2}. The application of the selection operator only on a small subset of the population and not on the entire population should again increase diversity in the population and prevent preconvergence.

In order to evaluate each solution, the objective function, $D_{K,}$ for each chromosome (solution) in the population is computed using equation 2. We use the dissimilarities among objects as stored in the dissimilarity matrix D.

5. RESULTS

All results were obtained using an original code developed in the software environment MATLAB, Mathworks, Inc.

This paper focuses on showing properties of the algorithm for different sizes of data sets (from one hundred to one thousand). Also, it aims to provide recommendations concerning the setting of its parameters and so enable its usage for application.
Because of the limited space available, the number of clusters is set to five. For future use, this number would not be known in advance and the results for different number of clusters would be compared.

Approaches based on genetic algorithms are stochastic in nature. We do not obtain the same result every time we run the algorithm. For this reason we performed 10 runs for each setting of the algorithm. All tables in this chapter show the average values of these ten runs.

Figure 4 shows the average objective function of the algorithm on DATA300 in case we did not use any injections. We can see results for both models (model 1 and model 2), different sizes of initial population (NP = 50, 100, and 150), and different number of generations (GN = 50, 100, 150 and 200).

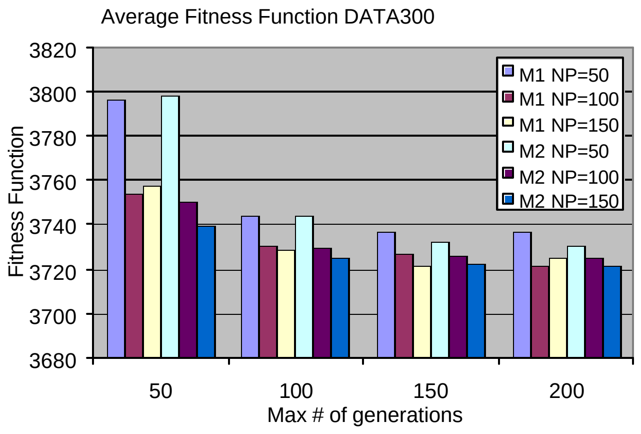

Figure 4: Fitness function (average of 10 runs) for both models (M1, M2), and different settings (NP,GN) on the DATA300.

The figure suggests that the performance is rather sensitive to the parameters of the algorithm. Also, the obtained results depend on the random seed used. In case we use a multiple population approach, the effect of random seed number, as well the sensitivity to particular parameters, is reduced. Results supporting this argument are provided in the following section.

5.1. PERFORMANCE FOR VARIOUS SIZES OF THE DATA SETS

In this part, I would like to show the performance of the algorithm for different sizes of data sets. We will consider the following data sets: DATA100, DATA300, DATA600, and DATA1000. Each number represents the size of particular data set.

Notation used in Figure 5.

:

Model	Type of model (1 or 2)
NP	Initial size of population
FinalP	Size of population at the end of run
Iter	Number of iterations
GN	Number of generations for each iteration
PmB	Probability of in-built mutation

Evaluation:

Fitness	Average fitness function
Time	Computational time of the algorithm
# of min	How many times out of 10 runs did the algorithm reach the minimum fitness function

The value of computational time provided in the results is only informative. It is clearly a function of population size and number of generations for each iteration. In some cases, the final solution can be found already in an initial phase of the computation, although the algorithm continues to run without any improvement of the solution. Termination criteria that are better than simply reaching the maximal number of generation should be used in order to improve the computational performance of the algorithm. However, since the algorithm does not have to perform in real time applications, the computational time is not the most important issue and its current values are satisfactory.

All parts of Table 1Figure 5.

are ordered by the fitness function (the lowest solution first). It means that the settings in first rows of each table could be recommended for given data set.
In the top of each of the tables above we provide the name of each data set used, as well as value of the minimum fitness function obtained. It is not guaranteed that this is a

global minimum. However, it indicates the minimal value obtained out of all runs of the algorithm. We could expect that this number is close to global optimal. This belief is supported also by the results discussed in description of Figure 5.

Table 1: The obtained results for different sizes of data sets (100, 300, 600, and 1000) and for two types of models, and different parameter settings

DATA 100 Min Fitness = 1222

Model	NP	FinalP	Iter	GN	PmB	Fitness	Time (sec)	# of min
2	20	320	5	60	0.05	1222	119	**10**
2	20	320	5	60	0.1	1222	115	**10**
2	10	640	7	50	0.05	1222	200	**10**
2	20	320	5	60	0.01	1223	124	8
2	10	160	5	50	0.05	1223.2	49	7
1	20	320	5	60	0.1	1223.5	124	8
1	10	160	5	50	0.05	1224	53	**9**
1	20	320	5	60	0.01	1224.5	135	7
1	10	640	7	50	0.05	1225.5	218	6
1	20	320	5	60	0.05	1226	130	6

DATA 300 Min Fitness = 3721

Model	NP	FinalP	Iter	GN	PmB	Fitness	Time	# of min
2	20	320	5	60	0.05	3721	135	**10**
1	30	1920	7	60	0.05	3721	1001	**10**
2	20	320	5	60	0.1	3721.7	123	7
2	20	320	5	60	0.1	3724	124	8
2	20	320	5	60	0.05	3724.2	127	8
1	30	1920	7	60	0.01	3727	929	8
1	20	320	5	60	0.1	3727.5	131	**9**
2	20	320	5	60	0.01	3729.9	133	4
1	20	320	5	60	0.1	3730	134	7
1	20	320	5	60	0.05	3733.7	141	5

DATA 600 Min Fitness = 7239

Model	NP	FinalP	Iter	GN	PmB	Fitness	Time	# of min
1	20	1280	7	60	0.05	7239.6	657	**9**
2	20	1280	7	60	0.05	7240.7	620	**9**
2	20	640	6	60	0.1	7242.5	308	**9**
2	20	640	6	60	0.01	7243.1	323	7
2	30	480	5	40	0.1	7243.4	145	4
1	20	640	6	60	0.05	7244.4	345	8
1	20	640	6	60	0.1	7244.4	313	8
2	30	480	5	40	0.05	7246	148	5
2	20	640	6	60	0.05	7247.9	315	7
1	30	480	5	40	0.1	7249.1	149	5

DATA 1000 Min Fitness = 11780

Model	NP	FinalP	Iter	GN	PmB	Fitness	Time	# of min
2	30	480	5	40	0.1	11780	189	**9**
2	20	640	6	60	0.05	11794	414	**9**
1	20	640	6	60	0.05	11794	421	**9**
1	30	480	5	40	0.05	11802	184	4
1	30	480	5	40	0.1	11804	177	6
2	30	480	5	40	0.05	11817	194	2
1	10	160	5	60	0.05	11830	108	1
1	30	480	5	40	0.01	11842	190	4
2	10	160	5	60	0.05	11855	137	1
2	30	480	5	40	0.01	11900	197	1

An example of the resulting “typical patterns” (medoids) found by the algorithm is shown in Figure 5. Since the objective function does not have a very clear meaning, these medoids can serve as an additional evaluation of results. Clearly, the medoids really represent the five original groups, SW, PW, LW, WD, and VAS. Similar results were obtained in every run of the algorithm.

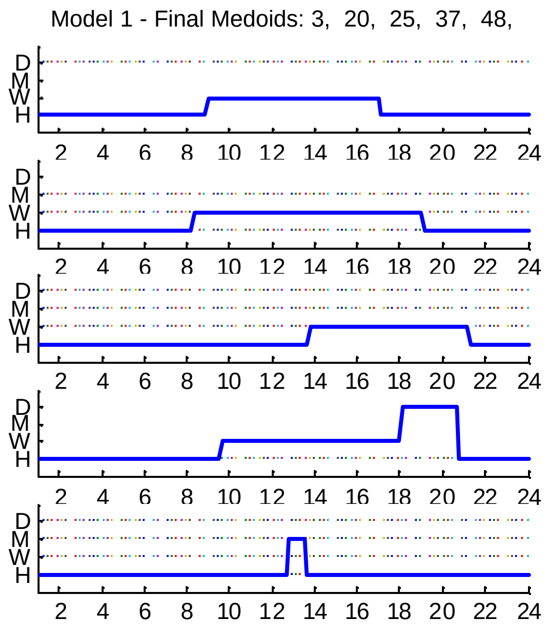

Figure 5: An example of the resulting medoids for DATA 300, model 1, and 4 iterations.

5.2. RECOMMENDED PARAMETER SETTING

There are several suggestions that can be implied from Table 1. First we can look at performance of particular models. The majority of the best few solutions was obtained using model 2. Also, the computational time of the second model outperform the first model for the same settings.

The table also implies that the probability of in-built mutation, P_{mB}, should be set in the range from 0.05 to 0.1, and values closer to 0.05 can be preferred.

It is not surprising that the algorithm perfoms better with a larger population size, higher number of generations, and also higher number of iterations. All these parameters increase the size of space being searched. On the other hand, increasing these parameters also increases the number of function evaluations, and so leads to an increase in computation time. Finding the equilibrium between these two features is desirable. It will clearly depend on the desired precision.

In general it is possible to recommend setting the initial population size, *NP*, equal to 20 or 30. The number of iterations, *Iter*, can be set in the range from 5 to 7, and the number of generations for each iteration, GN, can be set to 50 or 60. All these parameters should have higher values for data sets of a larger size.

6. CONCLUSION AND FUTURE RESEARCH

In this paper, algorithms to cluster activity patterns were developed. These algorithms as designed can be used for clustering large data sets of categorical objects. A dissimilarity measure that suits categorical data was applied here.

In general, the algorithm performs well even for large data sets. The algorithm as developed is not overly sensitive to the exact setting of its parameters. It performs reasonably well for a wide range of these parameters. This feature is of an important concern of all approaches based on genetic algorithms. The recommended setting of the parameters is provided in the previous part of this paper.

In order to be able to evaluate its performance with more implications, the algorithm needs to be compared to other tools used for solving of the same problem. For example, the development of other heuristic for the k-medoid algorithm, such as CLARA (Kaufman and Rousseeuw 1990), should be considered for future research.

Also, a more detailed study should be dedicated to the problem of dissimilarity among activity patterns. For example, the use of multidimensional sequence alignment method as described in Joh, and Arentze (2002) should be studied.

ACKNOWLEDGEMENT

The author would like to thank to Dr. Patrick Reed for providing many valuable comments and additional references concerning this paper.

7. REFERENCES

Everitt, B.S., S. Landau, and M. Leese. *Cluster Analysis*. Fourth Edition: Arnold, A member of the Hodder Headline Group, London, 2001.

Estivill-Castro, V. and A.T. Murray. "Spatial Clustering for Data Mining with Genetic Algorithms." http://citeseer.nj.nec.com/estivill-castro97spatial.html: NEC Research Institute. (1997)

De Jong, K., D. Fogel, and H.-P. Schwefel. *Handbook of Evolutionary Computation*. IOP Publishing Ltd. and Oxford University Press, 1997.

Goldberg, D.E. "The Race, the Hurdle, and the Sweetspot: Lessons From Genetic Algorithms for the Automation of Design Innovation and Creativity.," IlliGAL Report No. 98007. April 1998.

Huang, Z. "Clustering Large Data Sets with Mixed Numeric and Categorical Values." *In Proceeding of the First Pacific-Asia Conference on Knowledge Discovery and Data Minning* (Singapore, World Scientific) (1997).

Joh, Ch-H., T. Arentze, F. Hofman, and H. Timmermans. "Activity Pattern Similarity: a Multidimensional Sequence Alignment Method." *Transportation Research, Part B* (Elsevier Science Ltd.) 36 (2002): 385-403.

Kaufman, L., and P.J.Rousseeuw. *Finding Groups in Data, An Introduction to Cluster Analysis*. John Willey & Sons, Inc., 1990.

Kulkarni, A.A., and M.G. McNally. "An Activity-Based Travel Pattern Generation Model.," Institute of Transportation Studies, University of California, Irvine, December, 2000. UCI-ITS-AS-WP-00-6.

Lozano, J.A., and P. Larranaga. "Using Genetic Algorithms to Get the Classes and Their Number in a Partitional Cluster Analysis of Large Data Sets." http://citeseer.nj.nec.com/457425.html: NEC Research Institute.

Lucasius, C.B., A.D.Dane, and G.Kateman. "On k-Medoid Clustering of Large Data Sets with the Aid of a Genetic Algorithm: Background, Feasibility and Comparison." *Analytica Chimica Acta* (Elsevier Science Ltd) 282 (1993): 647-669.

Ma, June. "An Activity-Based Approach and Micro-simulated Travel Forecasting System: A Pragmatic Synthetic Scheduling Approach." PhD Thesis, The Pennsylvania State University, Department of Civil and Environmental Engineering, University Park, Pennsylvania (1997).

Maulik, U., S. Bandyopadhyay. "Genetic Algorithm-Based Clustering Technique." *Pattern Recognition* 33 (2000): 1455-1465.

Moraczewski, I.R., W. Borowski, and A. Kierzek. "Clustering Geobotanical Data with the Use of a Genetic Algorithm." *COENOSES* (C.E.T.A., Gorizla, Italy) 10, 1 (1995): 17-28.

Pas, E.I. "A Flexible and Integrated Methodology for Analytical Classification of Daily Travel-Activity Behavior." *Transportation Science* (Operations Research Society of America) 17, 4 (1983): 405 - 429.

Reed, P.M., "*Striking the Balance: Long-Term Groundwater Monitoring Design for Multiple Conflicting Objectives*", PhD Thesis, Graduate College of the University of Illinois at Urbana-Champaign, Urbana, Illinois (2002).