Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2007)

Evolutionary Parameter Setting of Multi-clustering

Dan Ashlock
Department of Mathematics
and Statistics
University of Guelph,
Guelph, Ontario, N1G 2R4
dashlock@uoguelph.ca

Ling Guo
Bioinformatics Program
Iowa State University,
Ames, Iowa, 50011
lguo@iastate.edu

ABSTRACT

Multi-clustering **is a technique for amalgamating the results of many runs of a standard clustering algorithm to obtain a clustering of data which avoid artifacts introduced by the underlying metric. Multi-clustering also yields an advisory, called a *cut plot*, as to the number of "natural" clusters present in the data. In order to perform multi-clustering a number of parameters must be chosen. This paper tests evolutionary algorithms that perform parameter setting for multi-clustering on synthetic data set with designed numbers of clusters. A evolutionary algorithm and an evolution strategy are compared. The superior algorithm, the ES, is then used to set parameters for four microarray-like data sets. Evolutionary parameter setting is found to more than double the range in which the cut plot detects the correct number of clusters when compared to hand-chosen parameters arrived at by serial parameter optimization. This paper also presents a new technique for accelerating multi-clustering, iteration limiting, and demonstrates that the technique may be implemented to speed up multi-clustering without impairing performance. The evolutionary results support the use of iteration limiting in multi-clustering.**

I. INTRODUCTION

Multi-clustering is a clustering technique that combines the results of many clustering runs to achieve a type of clustering that avoids including artifacts from the underlying metric used by the clustering algorithm. The goal of this study is to demonstrate that parameter selection for multi-clustering can be automated prior to its application to the clustering of microarray data sets. Multi-clustering is defined in [1] and many mathematical properties of the algorithm are given there. The authors are only aware of a modest amount of other work in algorithmic parameter setting for clustering methods. An example appears in [7]. Another approach to called multi-clustering that fuses data from multiple runs of a clustering algorithm appears in [2].

An advantage of amalgamating many runs of a basic clustering algorithm with multi-clustering is that the clusters found do not have a natural "shape". If k-means clustering is performed with the Euclidean metric then the clusters have a strong tendency to be approximate spheres, the most compact shape in that metric. Figure 1 shows the result of applying k-means clustering to a *donut-and-ball* data set in which the two "natural" clusters only one of which is an approximate sphere. With two clusters, the correct number, k-means clustering divides the data neatly and inappropriately in half. With six clusters the central sphere is correctly identified as a cluster but the outer ring is divided into five convex clusters. The algorithm for k-means clustering is given as Algorithm 1.

The output of Algorithm 1 is a category function,

$$C : S \to \{0, \ldots, k-1\}.$$

If two points i and j have the property that $C(i) = C(j)$ then we say that i and j are *in the same cluster*. We also say that i *is in cluster number* $C(i)$. The category function C is a convenient mathematical way of summarizing the clusters. It gives the number of the cluster containing a point. The first cluster in the k-means algorithm given here is cluster number 0. A feature of k-means clustering is that it is sensitive to its random initialization. If we were to re-run the k-means algorithm used to produce the two pictures in Figure 1 with a different set of initial cluster centers we would often get a different clustering. In performing multi-clustering we will exploit this sensitivity to random initialization.

Algorithm 1: **k-means**

Input: 1) A set S of points in $\mathbb{R}^n$
2) A desired number k of clusters.
3) A bound B on the number of cycles permitted

Output: A category function
$C : S \to \{0, \ldots, k-1\}$.

Details:
Choose k distinct points in S as initial cluster centers.
Repeat
Assign each point to the cluster whose center it is closest to, breaking ties at random*.
Recompute cluster centers as the average of all points in the cluster.
Until (no points change their cluster assignment or B cycles have occurred+)
Report the assignment of points to clusters as C.

* for real-valued data such ties seldom occur
+ for real-valued data B is seldom required

Suppose that we run k-means with too many clusters. In the bottom picture in Figure 1 we see that running k-means

Fig. 1. The result of using k-means clustering with two (upper) and six (lower) clusters on the donut-and-ball data set. Cluster membership is shown by glyph type.

on the first donut-and-ball data set with six clusters correctly discovered the central grouping. It also broke the donut into five pieces. In this clustering, every pair of points that are in a cluster together belong together. The problem is that most of the pairs of points in the "donut" cluster are in different clusters. Running a k-means clustering with an excess of clusters gives information about which pairs of points belongs together, but in a one-sided fashion. Positive examples are correct while negative examples are uninformative. The key observation that leads to multi-clustering is as follows. First, any one k-means clustering with an excessively large number of clusters yields useful information about which pairs of points should be associated. Second, rerunning the k-means algorithm yields potentially *different* information about which points should be associated. If we could group information from multiple k-means clusterings then we would get a much better notion of which points should be associated.

Informally, k-means based multi-clustering proceeds as follows. The user picks some number N of clusterings to perform. He then picks a distribution D of possible numbers of clusters. The algorithm performs N clusterings, selecting the number of clusters in a given clustering from D. Before clustering the algorithm initializes a set of pairwise connection strengths for each pair of points with an initial strength of zero. Whenever a k-means clustering places two points in a cluster together the algorithm increases their connection strength by 1. After running all N k-means clusterings, connection strengths are divided by the number of clusterings performed to yield connections strengths in the range $[0, 1]$. After all the clustering is done and the final connection strengths have been computed a cutoff value C is chosen. Only connections with strength exceeding C are retained. If we view the surviving connections as edges of a combinatorial graph [6] that has the data items as vertices then the clusters are the connected components of this graph. If we choose to make D a uniform distribution on a number of possible cluster sizes then multi-clustering requires three parameters: N and the upper and lower bounds of D. The cutoff value C is not supplied as a parameter. Rather we graph the impact of all possible values of C to make a decision about the value of C used. This graph is the *cut plot*. The multi-clustering algorithm is given formally as Algorithm 2.

The cut plot is a nice tool for allowing the user to see if there is a natural number of clusters. It is a function that maps possible cut values C onto the number of connected components that would result if the given cut value were used. The cut plot yields information about "natural" numbers of clusters. An example of such a cut plot for the donut-and-ball data set is shown in Figure 2. Note that it has a large flat spot at two clusters (the correct number of clusters).

Before proceeding to the problem of setting multi-clustering parameters, one additional parameter that yields substantial speed benefits needs to be added. Running a single instance of k-means clustering gives no reason not to run the k-means algorithm to completion. If k-means is run dozens or even thousands of times there is a reason to stop the algorithm before it runs to completion. Most of the information about which points are together are harvested in the first few iterations of the k-means algorithm. To exploit this a cut-off number of iterations μ is added to the algorithm. If our k-

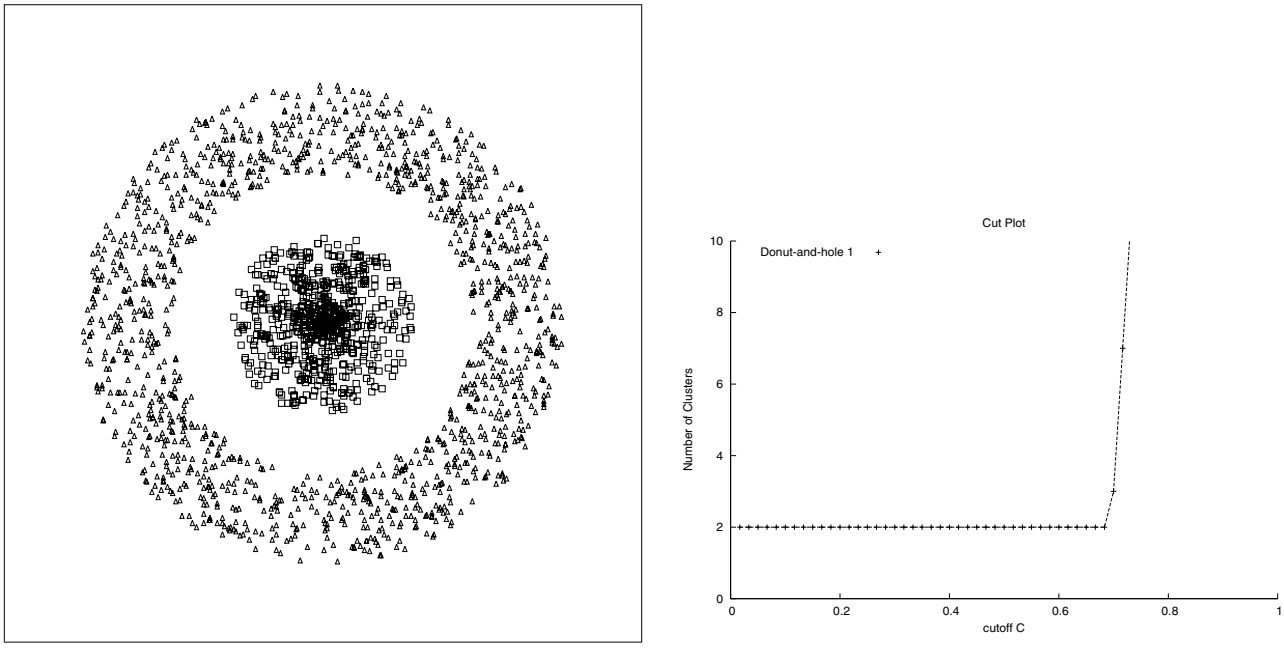

Fig. 2. The data set, as clustered with multi-clustering, and the cut plot for the donut-and-ball data set.

means algorithm does not finish in μ steps then algorithm stops anyway. Testing on the data sets used in this study show that the number of data points moving between clusters drops to less than 1% of the data in 5-10 iterations. The incorrectly clustered points are thus a small fraction of the points clustered. The number μ is one of the parameters to be set by the evolutionary algorithm. If a large value of μ is required then it can arise though evolution. This new version of k-means clustering is called *iteration limited* k-means clustering. The authors thank Steven Willson of the Iowa State University Department of Mathematics for pointing out the potential of iteration limiting k-means clustering.

The remainder of the paper is structured as follows. Section II formally states the parameter setting problem and the fitness function used to evolve solutions to it. Section III gives the design of two different evolutionary algorithms used to set the parameters. Section IV gives results while Section V discusses the results and draws conclusions. Section VI gives tentative next steps.

Algorithm 2: **k-means multi-clustering**

Input: 1) A set S of r points in $\mathbb{R}^n$
2) A number N of clusterings to perform.
3) A distribution D of numbers of clusters.
4) A weight cutoff C, $0 \leq C \leq 1$

Output: A category function
$C : S \to \mathbb{Z}$.
A cut plot
$f : [0, 1] \to \mathbb{Z}$

Details:
Initialize an $r \times r$ matrix $M$of pairwise connection strengths to contain all zeros
Repeat N times
Select an integer d from D
k-means cluster S with d clusters and stop number μ.
For each $\{i, j\} \in S \times S$ with i,j in the same cluster
Increment $M[i][j]$
Increment $M[j][i]$
end For
end Repeat
Normalize M[i][j] by dividing each entry of M[i][j] by N
Denote by W the graph on S with edge weights M[i][j]
For l equals 0 to N
Construct graph G with $V(G) = S$, $E(G)$ pairs of points for which $M[i][j] > l/N$
Compute number of connected components c of G
Add the point $(l/N, c)$ to the cut plot
end For
For x with $\frac{l}{N} < x < \frac{l+1}{N}$, $f(x) = f\left(\frac{l}{N}\right)$.
Build a new graph on S with edges where $M[i][j] > C$
Enumerate the connected components of this graph.
C[i] is the number of the connected component containing i.

II. The Parameter Setting Problem

Application of Algorithm 2 requires that the user choose the number N of iteration limited k-means clusterings (ILKMC) to perform, the distribution D of numbers of clusters to requested from the ILKMC algorithm in the form of D_{min} and D_{max} (the distribution is uniform on $[D_{min}, D_{max}]$), and the iteration limit μ. A parameters setting for a run of multi-clustering is thus the 4-tuple $(N, D_{min}, D_{max}, \mu)$. Given a set of parameters, the multi-clustering algorithm is run on a test data set with a designed number of clusters. The *fitness* of the parameter set is the number of cut values C of the form $\frac{i}{100}$,

$0 \leq i \leq 100$, that yield the designed number of clusters. This fitness function optimizes the largest flat region in the cut-plot to agree with the designed number of clusters.

It is perhaps important to note that it is possible to design test data sets that have ambiguous "correct" numbers of clusters. A data set with a hierarchical clusters of clusters, for example, can be designed to yield a cut plot with multiple broad flat regions. None of these are "right", they simply detect different parts of the hierarchy. This explains the use of the term *designed clusters* in this study. The evolutionary algorithms described in Section III optimizes multi-clustering parameters to detect the designed clusters; different designed clusters will yield different parameter settings which are good for different purposes. The data sets used to test the the parameter setting evolutionary algorithms are shown in Figure 3. The upper data set is called the *double horseshoe* data set while the lower is called the *spiral* data set. Both contain 600 points; the horseshoe has two designed clusters while the spiral has three.

The more effective algorithm, the ES as we will see in Section IV, was also applied to perform parameter setting on four sets of data designed to be similar to microarray time-series. These data sets are shown in Figure 4. All of these sets of data have five designed clusters and are in six dimensions. The microarray-like data sets were created by choosing a reasonable time-series profile as the starting point for each designed cluster. The members of each cluster were generated by selecting a standard deviation and adding Gaussian noise with the deviation to each point of each of the original profiles to generate the cluster members. The number of points in the data sets are 358, 527, 552, and 594 respectively as the sets appear from top-to-bottom in Figure 4. The details of the micro-array like clusters are given in Table I. These examples were chosen to give the multi-clustering algorithm a diversity of challenges in a small number of data sets.

An issue that deserves discussion is that of the choice of an evolutionary algorithm for parameter setting. With only four parameters a regular grid would seem to be a natural method for parameter setting. The parameters in question are, however, not ones on which the performance of the algorithm depends in a reasonable, continuous fashion. The parameters are in fact not even of the same type and their interactions are badly understood. Thus the a-priori knowledge was insufficient to design a search grid with any confidence. An evolutionary algorithm was much simpler than performing an immense number of runs needed to characterize the fitness landscape for the parameters. In addition it is clear that this fitness landscape must be searched again for new data sets - the optimal parameters for different types of data sets are found to vary.

Fig. 3. The data sets used for testing the parameter setting evolutionary algorithms.

III. The Evolutionary Algorithms

Two evolutionary algorithms are tested for setting the parameters of the multi-clustering algorithm. The first is an evolutionary algorithm(EA) operating on a population of 20 sets of parameters for the multi-clustering algorithm. The second is a (1,10) evolution strategy(ES). Members of an initial population for either algorithm are generated within the following rages for parameters. The value of N in the range [100,200]. The values of $Dmin$ and D_{max} are chosen at random in the range $[5, 60]$ subject to the limitation that $D_{max} - D_{min} \geq 3$. The parameter iteration cutoff μ for k-means is chosen uniformly in the range [2,30].

The evolutionary algorithm is steady-state[5] using tournaments of size two. In a mating event a tournament of two population members is chosen uniformly at random. The better parameter set is copied over the worse one, breaking ties uniformly at random. The copy is then mutated. In the ES mutations of the best parameter set replace all other population members. The mutation operator used replaces one of N, D_{min}, D_{max}, or μ with a new value selected according to the distribution used in the population initialization. The

TABLE I

NUMBER OF MEMBERS AND PER-POINT STANDARD DEVIATION FOR THE DESIGNED CLUSTERS IN THE MICROARRAY-LIKE DATA SETS.

Data Set	Cluster	Members	Std. Dev.
1	0	69	0.15
1	1	42	0.05
1	2	118	0.05
1	3	24	0.15
1	4	105	0.13
2	0	183	0.10
2	1	95	0.15
2	2	181	0.14
2	3	47	0.08
2	4	21	0.15
3	0	180	0.15
3	1	33	0.20
3	2	130	0.08
3	3	145	0.11
3	4	64	0.20
4	0	129	0.08
4	1	43	0.08
4	2	182	0.20
4	3	103	0.17
4	4	137	0.09

evolutionary algorithm is run for 215 mating events. The ES is run for 25 updatings. The evolution strategy thus examines 10+9x25=235 new structures each time it is run and the EA also examines 20+215=235 structures.

The small numbers of fitness evaluations used in these evolutionary algorithms reflect the very high cost of the fitness evaluation. At worst a fitness evaluation uses 200 iterations of k-means with an iteration cutoff of 30 (typically k-means would converge before this point), with only 5 clusters requested (more clusters yield results faster). At these parameter settings, multi-clustering is quite slow.

IV. RESULTS

Both algorithms, the EA and the ES, proved able to improve the number of cut-plot points out of 100 that found two clusters on the double horseshoe data set and three in the spiral data set. The improvement in quality of the the best structure is about 13% for the horseshoe data set and about 10% for the spiral data set. Table II gives a statistical summary of the results for the horseshoe and spiral data sets.

TABLE II

GIVEN BELOW ARE BEST VALUES AND 95% CONFIDENCE INTERVALS FOR THE 100 BEST-OF-RUN STRUCTURES FOR THE EA AND ES ALGORITHMS ON THE HORSE-SHOE AND SPIRAL DATA SETS.

Fitness summary			
Algorithm	Problem	Best Fit	95% confidence interval on mean
EA	Horseshoe	59	(54.7,55.4)
ES	Horseshoe	61	(55.5,56.1)
EA	Spiral	32	(26.7,27.3)
ES	Spiral	32	(27.5,28.2)

Estimating mean performance with 100 runs of each algorithm shows the ES is likely to be superior to the EA on the

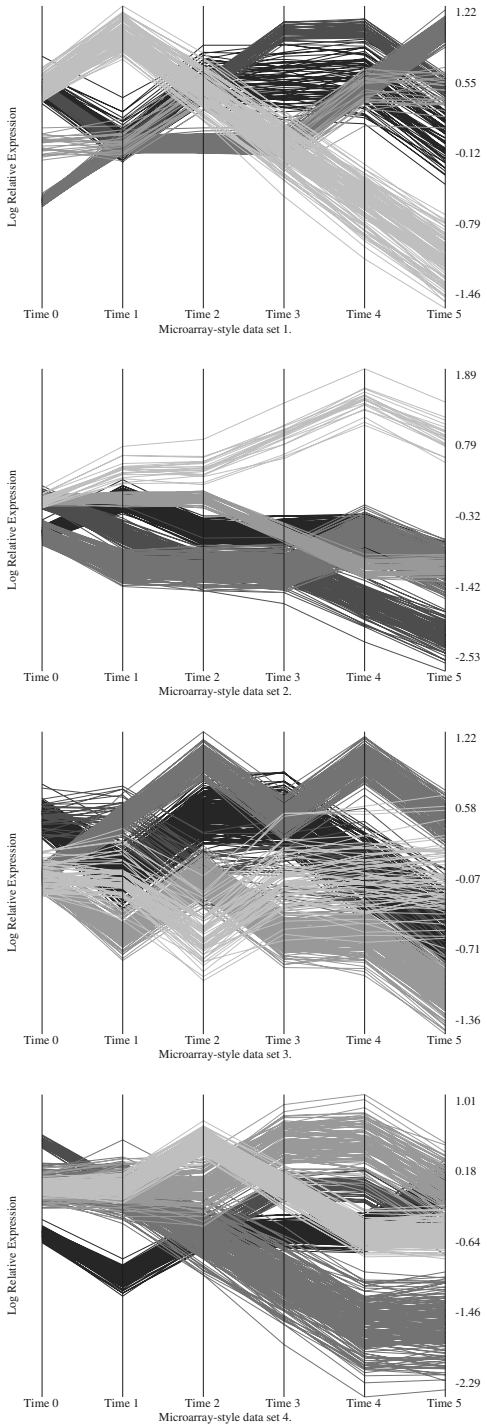

Fig. 4. Parallel coordinate plots for the four synthetic microarray data sets used for multi-clustering parameter setting.

TABLE III

BEST RESULTS ON THE SPIRAL DATA SET FOR BOTH THE EA AND ES.

N	D_{min}	D_{max}	mu	fitness
EA				
138	32	55	3	32
120	40	50	3	32
107	40	58	2	32
ES				
114	30	47	4	32

horseshoe data set but the difference is not significant at the 95% level. The best result for the ES on the horseshoe data set, placing 60% of the cut plot on the correct value (predicting two clusters) was $N = 138$ D uniform on the range $[21, 35]$ and an iteration cutoff of $\mu = 4$. For the EA the best result, placing 58% of the cutplot on the correct value was $N = 196$, D uniform on the range $[24, 36]$ and iteration cutoff $\mu = 4$. On the spiral data set the ES was significantly better than the EA. The best parameter sets for the spiral data set are shown in Table III.

The results for the application of the ES to the four sets of data similar to microarray time series yielded the results shown in Table IV. The difficulty of these data sets varied substantially but the algorithm was able to find set of parameters for all for data sets so that the correct number of clusters was at least 30% of the cut plot.

TABLE IV

GIVEN BELOW ARE BEST VALUES AND 95% CONFIDENCE INTERVALS FOR THE 100 BEST-OF-RUN STRUCTURES FOR THE ES ALGORITHMS ON THE FOUR SETS OF DATA SIMILAR TO MICROARRAY TIME SERIES.

Microarray fitness summary		
Problem	Best Fit	95% confidence interval on mean
Set 1	30	(22.3,23.3)
Set 2	76	(70.9,70.6)
Set 3	45	(37.0,38.2)
Set 4	36	(28.1,28.6)

V. CONCLUSIONS AND DISCUSSION

Both the EA and ES are capable of setting the parameters for multi-clustering to values significantly better than those in a random population. In the initial report of multi-clustering [1] the parameters setting, chosen by hand, were $N = 60$, D uniform in the range [10,100], and $\mu = 30$. This default set of parameters from the initial study were used 30 times and found to be much worse than the evolved parameter settings. A 95% confidence interval for the percentage of the cutplot yielding two clusters is (21.2,23.8). Compare this with the mid-fifties found routinely by the parameter setting algorithm. The atrocious performance is probably the result of the initial parameter setting studies having been performed on the donut-and-ball data set where these settings yield a correct number of clusters about 65% of the time (see the cutplot in Figure 2). This in turn suggests that parameter setting is needed for each type of data.

The two evolutionary algorithms tested in this study are both adequate for parameter setting and automate the process. The by-hand parameter setting study used serial parameter variation and, as such, is probably intrinsically inferior to either of the evolutionary algorithms. On one of the two data sets where the two algorithms were compared the ES proved significantly better.

Examining the values of μ in best results from the two evolutionary algorithms we see that in both $\mu <= 4$. The algorithm was permitted to select in the range [2,30]; in both algorithms on the spiral and horseshoe data sets the values of μ for best-of-run parameter sets were in the range 3-8 with values above 6 being quite rare. This means that the potential of μ to speed the multi-clustering algorithm without impairing quality is supported by the evolutionary results. The value $\mu = 30$ represents a situation in which the underlying k-means algorithm almost always terminates; evolutionary parameter selection showed that this is an unnecessary luxury, at least for the test data set used. This justification of the use of small values of μ is the most important result of this study.

The mean value of μ was computed for the best-of-run parameters found for all four sets of microarray-like data and these means are summarized in Table V. These results also support the idea that the value of mu can be profitably set to small values. Since this yields a substantial time savings this is indeed good news.

TABLE V

GIVEN BELOW ARE MEAN VALUES AND 95% CONFIDENCE INTERVALS FOR MEAN VALUES OF μ IN THE 100 BEST-OF-RUN STRUCTURES FOR THE ES ALGORITHMS ON THE FOUR SETS OF DATA SIMILAR TO MICROARRAY TIME SERIES.

Microarray fitness summary		
Problem	Mean μ	95% confidence interval on μ_{avg}
Set 1	3.55	(3.39,3.71)
Set 2	4.67	(4.41,4.93)
Set 3	6.20	(5.93,6.47)
Set 4	4.11	(3.88,4.34)

The attempt to select a set of microarray-like data sets that would provide a diversity of challenges to the multi-clustering algorithm was done by the use of intuition; the broad variety of fitnesses obtained suggests that the effort succeeded. The variability of the outcomes for the microarray data is substantial with the easiest data set being correctly clustered for 76% of the length of the cutplot while the hardest was correct along only 30% of the cut plot. The hardest of the 6-dimensional data sets was only slightly harder than the 2-dimensional spiral data set.

The results of this study suggest that there is no good default set of parameters for multi-clustering. Instead, different data sets have different good sets of parameters. The μ parameter has a gratifyingly small range, and one in its lower values, among the best-of-run evolved parameter sets. Other parameters show no clear pattern beyond being similar when they were evolved for the same data set. A larger set of

experiments is needed to determine if there are good rules of thumb for setting the parameters.

One obvious rule of thumb is that the distribution D should have a range that only includes numbers larger than the actual number of clusters in the data. This is obvious because we do not wish to force members of distinct natural clusters to be together in any of the individual k-means clusterings. Disconcertingly this rule of thumb is also wrong: during testing of the k-means multi-clustering algorithm on a wide variety of microarray-like data sets it was found that superior results could be obtained by letting the number of clusters chosen by the distribution D include numbers smaller than the designed number of clusters. The reason for this is not clear and is under investigation. This discovery also motivated the current evolutionary parameter setting study.

The decision not to use any of the available biological microarray data sets in this study was driven by the lack of a good division into clear clusters in published biological data. The yeast time-series data at Stanford, a standard data set for microarray analysis software testing, was analyzed with hierarchical clustering. This causes it to lack clear clusters for testing the multiclustering software. Until the behavior of the multi-clustering algorithm is more fully understood we are reluctant to apply it to messy biological data.

VI. Next Steps

One limitation of the multi-clustering technique is that it is slow. When multi-clustering becomes the fitness function of an evolutionary algorithm this limitation becomes painful, yielding the small numbers of repetition and short length of evolution in this study. Possibly the most important result of this study is to show that stringent upper bounds on μ can probably be used without impairing performance. If the result stands it will speed both multi-clustering and additional evolutionary exploration of the multi-clustering parameter space.

In addition to the time spent on individual k-means clusterings, substantial time is also spent constructing the cut-plot. In normal use the cut plot is of substantial utility; in the parameter setting algorithms is functions as a fitness function. There is room for algorithmic improvements to speed up the cut plot algorithm. In order to compute the cut plot the number of clusters resulting from some set of of cut-values must be computed. Theorem 1, proved in [3], gives a finite bound on the number of cut-values that must be computed.

Theorem 1: The cut plot $f(x)$ is a non-decreasing step function with at most $r - m$ steps, where m is the number of connected components in V, the weighted graph constructed by the multi-clustering algorithm. These steps occur at the weights of the edges of a maximal spanning forest of V.

Examine the cutplot shown in Figure 2. Most of the jumps in the cutplot happen at the right hand side of the plot, well after any large flat spots that indicate a natural number of clusters. Construction of a maximal spanning forest can be done with an efficient greedy algorithm such as a simple modification of Kruskal's algorithm[4]; using *regula-falsi* on the critical weights obtained will permit the computation of the exact fraction of the cutplot that yields the correct number of clusters with roughly $log(r - m)$ evaluation of the number of clusters where r and m are as in Theorem 1. This should permit substantial speed improvements to the fitness function used in this study.

Assuming that the speed improvements have been implemented then the parameter setting algorithm should be applied to a greater variety of data sets and to biological microarray data sets. Such additional data should allow an analysis to determine which parameters as soft with respect to good performance and which are not. Such an added collection of data will permit an analysis to determine if good settings for multiclustering are idiosyncratic to particular data sets or if there are good default settings or rules of thumb for broad classes of data sets.

Another avenue for investigation is the use of multi-clustering for hierarchical clustering. This form of clustering is most commonly used with micro-array data and the weighted graph located with multiclustering can be used to create a hierarchical clustering. As the cut-value is increased, the resulting partitions of the data are consistent; once separated points stay separated. Mathematically the successive partitions resulting from increasing the cut-value refine one another. This means that as we sweep the cut-value we obtain a hierarchical relationship among the clusters for different cut values. The result is a tree-structured description of the way the data clusters at all cut-values, similar to the results of standard hierarchical clustering algorithms. The details of using multi-clustering as a hierarchical clustering tool are given in [3]; it remains to test this tool.

VII. Acknowledgments

The first author would like to thank the University of Guelph for its support of this research. Both authors thank Steven Willson of the Iowa State University Department of Mathematics for pointing out the potential of adding the μ parameter which substantially speeds the process of multi-clustering.

References

[1] D. A. Ashlock, E.Y. Kim, and L. Guo. Multi-clustering: avoiding the natural shape of underlying metrics. In C. H. Dagli et al, editor, *Smart Engineering System Design: Neural Networks, Evolutionary Programming, and Artificial Life*, volume 15, pages 453–461. ASME Press, 2005.

[2] D. S. Frossyniotis, M. Pertselakis, and A. Stafylopatis. A multi-clustering fusion algorithm. In *SETN02: Proceedings of the Second Hellenic Conference on AI*, pages 225–236, New York, 2002. Springer.

[3] Eun-Youn Kim. *Analysis of Game Playing Agents with Fingerprints*. PhD thesis, Iowas State University, Ames Iowa, 50010, 1995.

[4] J. B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[5] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. In *Foundations of Genetic Algorithms*, pages 94–101. Morgan Kaufmann, 1991.

[6] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ 07458, 1996.

[7] Wei Yang, L. Rueda, and A. Ngom. A simulated annealing approach to find the optimal parameters for fuzzy clustering microarray data. In *SCCC 2005. 25th International Conference of the Chilean Computer Society*, pages 7–10, 2005.