Clustering under Multiobjective Optimization Constraints with Genetic Algorithms

Albert Ali Salah*, Stanislav Redman, Gabriella Kovács

Abstract— **In this paper, we attempt to solve a clustering problem, which requires simultaneous optimization of several objective functions, with a genetic algorithm. Our emphasis is on the representation of the problem, and the choice of an appropriate aggregate fitness function. On the toy problem of assigning the students of the Santa Fe Complex Systems Summer School into working groups, we try to point out some important general aspects of the task.**

Keywords— **Multiobjective constraint optimization, clustering, genetic algorithms, aggregate fitness**

I. Introduction

The multiobjective optimization problem is an open problem that has found applications in many domains. Roughly, it corresponds to the case where the aim is to select values for a set of decision variables that will satisfy or optimize several, possibly conflicting functions.

In the presence of multiple criteria, it is not clear how the trade-off between individual constraints should be resolved. For this problem, the notion of *Pareto optimality* is introduced [15]. A solution is Pareto optimal, if there is no feasible set of variables that will lead to a better value in some of the constraints without worsening the value on others. However, there usually is a **set** of Pareto-optimal solutions, and even if that set can be found analytically, the task of selecting one solution over the others persists. (For more information on Pareto optimality and how it relates to the performance of multiobjective genetic algorithm, see [4].)

If a global optimum is not sought, the genetic algorithm (GA) is suitable for this kind of problems, and used frequently in the literature ([6], [7], [20], [21], [22], [23], [24], [25]). Most of the applications involve function optimization, and none of them involves clustering. (For a good survey with pointers to applications, see [2].)

The clustering imposes additional structure on the optimization problem. In the clustering problem, the decision variables indicate which data points belong together as a cluster. The constraints that are to be optimized can be related to individual data points, or to the whole assignment. An example for the former is constrained clustering, where we have partial information about the data; we know that several data points belong to the same cluster, and several data points must be in separate clusters [10]. An example for the second case is the constraint that the clusters should be balanced in size. This property pertains to the whole assignment.

This type of clustering is similar to the knapsack problem, and GAs are frequently employed for NP-hard problems. There are however only a few instances where clustering problems are solved with GAs (for example [3], [5], [14], [17]), and the representations used in those cases are not scalable to larger problem instances [5].

This paper is organized as follows. In the next section, we formalize the problem in its general form. Then we introduce the GA methodology, and give detailed considerations on the representation and fitness issues. In the fourth section, we introduce the case study, and describe the aggregate fitness function we have used to solve it. The fifth section briefly summarizes our comparative results. In the last section we conclude and give a glimpse of future directions for this work.

II. Formalization of the Problem

In this section, we will give a formal definition of the clustering problem, bearing in mind that it is going to be used in GA. We begin by making two simplifying assumptions. Without loss of generality, we assume that all the constraints are posed as maximization problems. The second assumption looks more drastic, but changes little in the actual implementation. We define our problem to be a free optimization problem, where all the solutions are feasible. The aim is still to optimize the constraint functions, but there are no fixed thresholds to make a possible solution infeasible. We hope that the formulation of the aggregate fitness function will be enough to deal with actual infeasibilities in the problem. One can always check thresholds and add a large penalty to naturally eliminate the infeasible solutions.

Given N multi-dimensional data items, and a number M to indicate the maximum number of clusters, the problem is to find an assignment

$$\Delta^{\star}_{i,j} = \begin{cases} 1 & \text{if data item } i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

with

$$i = 1 \ldots N,$$
$$j = 1 \ldots M$$

such that for C constraints,

$$\overline{f} = \{f_1, f_2, \ldots, f_C\} \quad (2)$$

is Pareto optimal. f_k are the individual constraints.

*Corresponding author

Albert Ali Salah is with Perceptual Intelligence Laboratory, Boğaziçi University, 80815, Istanbul, Turkey. E-mail: salah@boun.edu.tr.
Stanislav Redman is with Economics Department, St.Petersburg State University, St.Petersburg, Russia. E-mail: stas@sr1983.spb.edu.
Gabriella Kovács is with Department of Mathematics and Computer Science, North University of Baia Mare, 4800 Baia Mare, Str. Victoriei 76, Romania. E-mail: kovacsg@univer.ubm.ro

III. The Genetic Algorithm

The genetic algorithm is a heuristic optimization tool that works on a *population* of individual solutions. The individuals are coded in *chromosomes* that are usually binary bit strings, and the search for a good individual is performed on the chromosome space. The chromosome, or the *genotype* of the individual, is *decoded* to reveal the properties of the solution, also called the *phenotype*, and the phenotype is used to evaluate a *fitness function* that indicates the goodness of the solution.

There are several operators that guide the search in a GA. Since the algorithm works with a population, maintaining the *diversity* in the population is of great concern. The unary operator *mutation* is used to randomly change bits of the chromosomes with a small probability. The binary operator *crossover* is used to exchange parts of two chromosomes, belonging to different individuals. Crossover is the most important operator in GA, as it can combine the strengths of different individuals to produce better solutions [19]. Holland developed the schema theorem to explain why this selection of operators works, but his claims do not go undisputed [11].

Most of the GA work in a generative way. Given a randomly created initial population, the algorithm proceeds by generating a new population from the old one at each iteration. The *selection* operator can take many forms, usually it is an elitist strategy that allows better individuals to be selected more over poor individuals, and thus the average fitness of the population increases steadily. One simple and popular scheme is the roulette-wheel selection, where each individual gets a slice on the probabilistic wheel proportional to its fitness. Two individuals are selected with replacement to be *parents*, they undergo crossover and mutation with some probability, and new individuals are created (called the *children*), and placed in the new population. There are many different selection strategies, but the principle is the same [8]. There are also nongenerative ways of proceeding in GA. At each iteration, the worst individual can be selected and replaced, instead of the whole population [23].

The designer of a GA is faced with a number of questions. Apart from setting the parameters of the GA (population size, mutation and crossover probabilities, selection scheme), the representation of solutions and the fitness function that guides the search are of paramount importance.

A. The Representation

In GA, the first task of the coder is to represent the problem. Usually the information that is contained in a possible solution is coded into a binary string. However, there are many ways of doing this (see for example [9]). A good GA representation is:

1. **Unambiguous.** There should be no randomness involved in decoding the chromosome, and each genotype should map to a single phenotype.
2. **Short.** The search space and the computational complexity are dependent on the chromosome length. If the chromosome is k bits long, the size of the search space will be 2^k, and this number should be as close as possible to the actual solution space of the problem. If the representation involves too much redundancy, the performance will degrade. However, there is a trade-off, and redundancy has its own uses [12].
3. **Smooth with respect to the fitness landscape.** If similar chromosomes have similar fitness values, the changes due to genetic operators will move an individidual on a smoother fitness landscape. This is valuable, because such a landscape will have fewer peaks to explore, and local search performs better on smooth landscapes than spiky ones.
4. **Robust to mutations.** Some problems are hard to represent with binary strings, and therefore not frequently attempted with GA. For example in the travelling salesman problem, the aim is to find the shortest tour that passes through all vertices of a graph once, and returns to the starting vertex. Since a valid solution must be connected and should connect all the vertices, many randomly selected arc sets will not represent valid solutions. Given a representation, if a mutation is able to produce a non-valid solution from a valid solution, the representation is not robust; the fitness drops to zero with a small push. It is much better if a bad mutation produces a valid individual with low fitness value rather than a non-valid individual.
5. **Free of preferential bias.** The representation should not favour parts of the search space more, unless information about the problem domain is used specifically to bias the representation.
6. **Simple to decode.** Since decoding will be performed frequently, it should not be a performance bottleneck.

B. The Fitness Function

The fitness function guides the search, and has a huge impact on the success of the algorithm. A good fitness function is:

1. **Complete.** It should assign a value to each individual on the search space.
2. **Correct.** The function should be a correct ordering on the individuals with respect to their closeness to the optimal solution, or a good approximation to it.
3. **Informative.** The function values should indicate the relations between possible solutions as well. If the fitness difference between a and b is much larger than the fitness difference between b and c, this should indicate that the goodness of these possible solutions is approximately related proportional to their fitness differences. This is especially important if a fitness-based selection procedure is used.
4. **Pragmatic.** Fitness functions are usually interpreted probabilistically, and therefore created to produce fitness values between zero and one. If the boundary conditions are too far to the actual solutions, trying to stretch the interval between 0 and 1 to span the search space can be problematic. For example if a variable controls the fitness, and the individuals are clustered around a small subinterval of the complete set of possible values for this vari-

able, stretching will assign to individuals fitness values that are very close to each other. If the selection operates on fitness value probabilistically, the noise in the random process will be greater than the fitness differences, and the search will degenerate into random search. A good fitness function will devote a greater interval to the range of actual solutions, and care less about the boundaries.

5. **Simple.** Often, the calculation of fitness values involves complex mathematical operations and simulations. For example in a GA used to find a good strategy to play Prisoner's Dilemma, each chromosome codes a strategy, and the fitness values are found by simulating games between the individuals [1]. This has a huge computational load on the system, and should be avoided if possible. Frequently, complex fitness functions are replaced with simpler approximations, with the assumption that the algorithm will make good use of the gained time and will come up with a better solution.

C. Fitness Functions in Multiobjective Optimization

In the multiobjective case, we have different dimensions, each of which contributing a fitness value. The task of the designer is to combine these fitness functions in an *aggregate fitness function*. The most important concern in this process is to avoid domination by one fitness term, which means only the dominating term is optimized at the end. There are several approaches to aggregate fitness functions, and the selection of any particular method over others depends on the designers knowledge of the problem domain and how much he or she is willing to trade-off computational efficiency with accuracy. We will list some of the interesting methods of aggregation, the interested reader should refer to [2].

1. **Weighted sum.** The individual fitnes terms are multiplied with weights that indicate their relative importance, and then summed. If the fitness terms are between 0 and 1, and the weigths sum up to 1, the resulting total fitness will be between 0 and 1 as well. The designer needs information to set the weights, but this method is simple, and computationally cheap.

2. **Goal programming.** The designer sets goal values for the constraints. If the actual term deviates from the goal, it is penalized in the fitness function. Also computationally cheap, this method requires even more information about the problem.

3. **ε-constraint method**: In this method, one of the constraints is used as the fitness criterion, and the other constraints are fixed. This method is usually used in hybrid schemes, where the GA is supported with a local, gradient-based search algorithm.

4. **VEGA.** Vector Evaluated Genetic Algorithm is an interesting method that modifies the selection operator in face of multiobjective constraints [21]. For each constraint, a subgroup of the population is selected for generating individuals for the next generation. It is like running different, smaller roulette-wheels, where individuals are not evaluated with their total fitness, but have different slices from each wheel, depending on their fitness terms. This roughly corresponds to specialization, as individuals of the n^{th} subgroup have better values for the n^{th} constraint. This approach misses potentially valuable individuals with above average values on most of the constraints, and special heuristics should be used to deal with it.

5. **Nongenerative techniques.** In nongenerative techniques, we can have a mixed selection criterion to decide on the worst individual in the population. This individual is going to be replaced [23]. The selection can be performed by looking at the minimum, mean, average, or weighted fitness values. A round-robin on the constraints can be used to replace the individual with the worst fitness value on each of the constraints.

6. **Relative fitness.** Fonseca and Fleming suggest a scheme where each individual gets a rank based on the number of individuals that dominates it on one or more fitness terms [7]. The fitness assignment is then made based on this rank.

Armed with the design considerations on representation and fitness, we will introduce the problem in the next section, and propose a GA to solve it.

IV. Case Study - Assignment of People to Working Groups

A. Introducing the Problem

The problem that initiated the effort for the present work was the assignment of students in the interdisciplinary Santa Fe Complex Systems Summer School into working groups. The summer school had 52 students, coming from different disciplines, with different interests and skills. These people needed to be placed in groups so that they can discuss and come up with project ideas that would allow them to work together. One loose constraint was that each group should not contain more than 10 persons. To complicate matters, there were no general topics to assign to groups in advance.

The manual solution to the problem consisted of asking people to come up with ideas for discussion and projects, gathering them under general titles, and checking whether the present list of group titles ensures a fair distribution of people into groups. Consequently, it required many iterations, resulted in unbalanced groups, and created an uneven distribution of basic skills across groups.

If quantitative data about the skills and interests of the students can be gathered in advance, this problem can be solved offline to create an initial assignment of people into groups. Although there are many constraints and variables in the real problem that cannot be modeled for optimization (how can an 'attractive' student be modeled for instance?), the computer generated solution may serve as a basis for improvement. We propose to gather student data with a questionnaire, and create the initial assignment using a GA.

B. Formalization

The problem can be posed as a clustering problem with the following properties:

1. Each data point (student) is represented as a multidimensional vector. The dimensions are integer valued, and represent the knowledge level and interests of the student.
2. The clusters are the working groups, and their number is fixed a priori.
3. In addition to the distance minimization, the clustering will take account of a number of different constraints. In the absence of constraints, the problem degenerates into k-means clustering [13].

We have identified the following constraints to be of importance for our problem:
1. The interests of each student must be as close as possible to the average interest vector of the working group the student belongs to.
2. The working groups should be balanced in the number of students.
3. There are basic (for the present context, of course) skills that will be needed in each group. The maximum level of competence for each of these skills should be maximized for each group. This means that there will be at least one competent person in each group for any given basic skill. We have identified four basic skills: mathematics, programming, English, and statistics.
4. Each group should have enough domain knowledge on the areas of its interest. The skills and interests are treated in different terms, as it is possible for a neuroscientist to be interested in doing a project on financial time series analysis. But the group that contains the neuroscientist should contain at least one other person that is experienced in finance, if possible. The fields of interests were chosen to be anthropology, biology, cognitive science, computer science, economics, evolution, information theory, multi-agent systems, neural networks and simulation, neuroscience, optimization, philosophy, physics, psychology, quantum consciousness (out of sheer curiosity), self-organization, and social networks.

C. Representation

We have considered two possibilities for the representation. The problem with clustering representations in GA is that they don't scale to large data sets, but since our problem is small, this is no obstacle.

In the first representation, we use three bits for each individual, and the bits code the binary number that is the label of the cluster the individual is assigned to. We fix the number of clusters to be 8, for simplicity. The length of the chromosome is 156 bits. The advantage of this representation is that it's simple. A mutation relocates an individual, so the effect is not very drastic. However, it is not very smooth, there is no useful redundancy, and it is difficult to preserve structure in crossover.

Considering these drawbacks, we considered another possibility (that didn't work for reasons we will explain). Suppose we code the labels of two data points consecutively in the chromosome, indicating that they belong to the same cluster. We can code a fixed number of these pairs (or rather edges), and the result will be a clustering, with arbitrary number of clusters.

We have analyzed the clustering behaviour of this representation. As the number of edges are increased, the number of clusters drop, as expected (Fig 1). Following this regime, the number of edges necessary for a targeted number of clusters can be determined. However, a look at the size of the greatest cluster reveals that a giant component is created very early in the process (Fig 2).

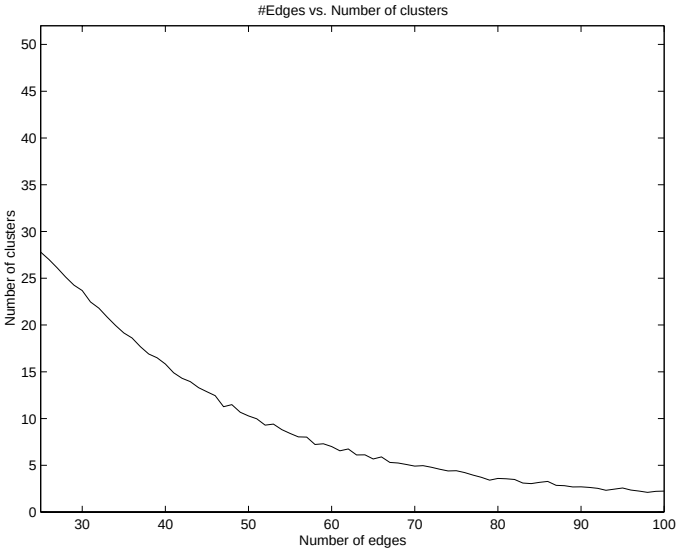

Fig. 1. Number of Clusters vs Number of Edges

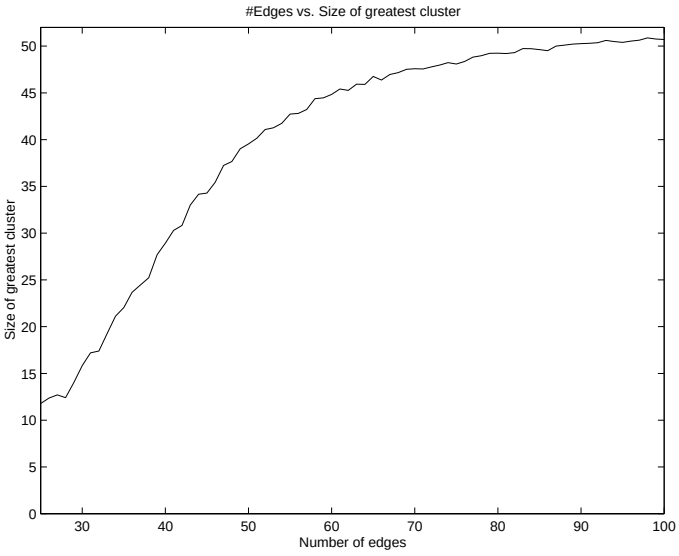

Fig. 2. Size of the Greatest Cluster vs Number of Edges

We have also tried a two-level scheme, where in addition to the chromosome piece that codes edges between data points, we have edges that connect the first-level edges. This way, more structure can be preserved as clusters can be joined or separated. Also, the mutations can change the genotype without affecting the phenotype. This type of redundancy can be useful in preserving information.

D. Fitness Terms

We have decided to use the weighted-sum aggregate fitness function for our problem. Assume $f_1, f_2, \ldots, f_C$ are the individual fitness functions that order the solutions according to individual constraints. The total fitness function is

$$f = \sum_{i=1}^{C} \alpha_i f_i \tag{3}$$

where

$$\sum_{i=1}^{C} \alpha_i = 1, \quad \alpha_i \geq 0 \tag{4}$$

We have four fitness terms that were explained previously. Now we derive the exact formulations.

D.1 Interest Term

The aim of the interest term is to ensure that the group members have similar interests. The interests are integer valued between 1 and 4. If the number of interests is S, and the number of students is N, the maximum interest difference on any dimension is $9SN$. Using the maximum to scale the difference between zero and one, we get

$$f_1 = \frac{9SN - \sum_{i=1}^{N} \sum_{j=1}^{M} (p_i - g_j)^2 \Delta_{i,j}}{9SN} \tag{5}$$

where M is the number of clusters, p_i is the interest vector of student i, g_j is the mean interest vector of group j, and $\Delta_{i,j}$ is the Kronecker delta, defined in Eq 1.

The problem with this formulation is that it does not take the nature of the data into consideration. It is desirable to give the fitness function a little elbow room around the empirical mean value of the average distance parameter, as the success of each individual in the population depends linearly on its fitness. The roulette-wheel selection of parents requires the fitness differences between individuals to be as large as possible to overcome the noise in random selection. The function we propose is equal to 1 if the interests of the individual are equal to the group mean, and it decreases rapidly after a certain value of difference:

$$f_1 = 0.8^{\frac{1}{SN} \sum_{i=1}^{N} \sum_{j=1}^{M} (p_i - g_j)^2 \Delta_{i,j}} \tag{6}$$

D.2 Balance Term

We want to have the groups balanced in number of students. Assuming the average number of students per group is N/M, the fitness term looks like:

$$f_2 = \frac{N^2 - \sum_{j=1}^{M} \left(\left(\sum_{i=1}^{N} \Delta_{i,j} \right) - \frac{N}{M} \right)^2}{N^2} \tag{7}$$

D.3 Skill Term

We want each group to have students who could provide expertise in basic skills. The basic skills are integer valued between 1 and 4. We will try to maximize the maximum of existing basic skills in each group:

$$f_3 = \frac{9MB - \sum_{j=1}^{M} \sum_{k=1}^{B} \left(4 - \arg\max_i \left(b_{i,k} \Delta_{i,j} \right) \right)^2}{9MB} \tag{8}$$

where B is the number of basic skills, and $b_{i,k}$ is the k^{th} basic skill of student i.

D.4 Knowledge Term

The knowledge terms are integer valued between 1 and 4. We consider the three main interests of each group, and require that they have enough expertise on these areas:

$$f_4 = \frac{27M - \sum_{j=1}^{M} \sum_{k=1}^{S} \left(4 - \arg\max_i \left(h_{i,k} \Delta_{i,j} \Phi_{j,k} \right) \right)^2}{27M} \tag{9}$$

where $h_{i,k}$ is the k^{th} knowledge term of student i and $\Phi_{j,k}$ is 1 if the k^{th} interest term is among the first three interests of group j, 0 otherwise.

V. Results

We have compared the results obtained from the GA with several clustering techniques. Since traditional clustering doesn't allow for multiobjective optimization, we used the most straightforward optimization criterion - the interests of the people in the same group had to be as similar as possible. We will give the distance function used in each clustering, and then we present our results without much comment, except stating that the GA works good for a problem of this size (See Table I). The GA we have selected has a population size of 100, mutation probability 0.001, single-point crossover with probability 0.4, equal α_i coefficients, and 20 generations. We have also obtained statistics about the interests and knowledge areas of the students of the Santa Fe School in the process. Interested readers can check [16] for these results.

A. Clustering Methods

We have used four different clustering techniques. In each method, the Euclidean distances between 17-dimensional interest vectors was calculated first. Then the clustering was carried out according to a rule of measuring distance $d(r, s)$ between clusters r and s. In the following equations, n_r and n_s stand for the number of students in clusters r and s, $\bar{x}_r$ and $\bar{x}_s$ are the mean interest vectors of the clusters, and $dist(x, y)$ is the Euclidean distance between vectors x and y. The ward linkage performed best amongst them, although the GA outperforms it. We indicate the distance functions used for each algorithm.

A.1 Nearest Neighbour

This rule ensures that if two students in different groups have similar interests, those groups are likely to merge. According to this rule, 45 out of 52 students were assigned to the same group, and the rest of the groups consisted of individuals. Distance measure is:

$$d(r, s) = \min \left(dist(x_{ri}, x_{sj}) \right) \tag{10}$$

with

$$i = 1 \ldots n_r$$
$$j = 1 \ldots n_s$$

A.2 Furthest Neighbour

This rule makes sure that the individuals with opposite interests never get into the same group. The distance measure is:

$$d(r,s) = \max\left(dist\left(x_{ri}, x_{sj}\right)\right) \tag{11}$$

A.3 Average Linkage

Average linkage rule takes into account all individuals in a considered group, and uses the average distance.

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist\left(x_{ri}, x_{sj}\right) \tag{12}$$

A.4 Ward Linkage

Ward linkage favors relatively balanced groups. This rule produced the best results among the considered clustering techniques.

$$d(r,s) = \frac{n_r n_s dist\left(\bar{x}_{ri}, \bar{x}_{sj}\right)^2}{(n_r + n_s)} \tag{13}$$

TABLE I
VALUES OF FITNESS TERMS WITH DIFFERENT METHODS

	Interests	Balance	Skills	Knowledge
Nearest Neighbour	0.85	0.37	0.72	0.93
Furthest Neighbour	**0.89**	0.93	0.96	0.96
Average Linkage	0.88	0.82	0.90	0.95
Ward Linkage	**0.89**	0.97	0.97	0.97
GA	0.85	**0.99**	**0.99**	**0.98**

VI. Conclusion and Future Directions

We have used GA to solve a multiobjective optimization problem that has the additional handicap of being a clustering problem. This is a novel combination, and there is a lot to investigate. We barely scratched the surface because of time constraints. In this section we attempt to indicate possibilities for further research.

The weights of the individual terms effect the solution found by the algorithm. One interesting question is whether we can observe the behaviour and characteristics of the individual fitness terms as the algorithm is running and change the weigths on the fly to obtain a better solution. This resembles a continuous version of switching from one highly biased fitness function to a tighter fitness function after a pre-determined number of generations (called the *switching time*) [18]. Our initial aim was to use a second level GA to select the weights. Each weight assignment takes the GA on clustering to some other solution, possibly on the Pareto front. The problem turned out to be too small for that scheme, a good solution is quickly found by the single level GA. A two-level GA can be employed for larger problems, but it may rather be necessary to consider a faster weight optimization algorithm, as a two level GA is computationally expensive.

It seems essential to select a larger problem as a testbed. Only on the larger problem can the representational hurdles of clustering with GA be attacked. We believe that a multi-level link representation with regulation on the giant component can be a solution, but this needs investigation.

One simple heuristic which may be useful in determining the weights can be a random sampling of the chromosomes to derive a statistical approximation to the behaviour of the individual fitness terms. If their spread can be determined, they can be normalized without manual intervention, according to some simple principles. Whether a random sample will work or not should be determined by simulation.

In almost all GA applications special care is given to ensure diversity in the population. In small problems, it is almost impossible to distinguish between fast convergence and premature convergence, as the algorithm usually finds a good solution. We have yet to test our scheme in a large problem to see whether special measures are needed to ensure diversity.

References

[1] Axelrod, R., "The Evolution of Strategies in the Iterated Prisoner's Dilemma", in L.D. Davis (ed.) *Genetic Algorithms and Simulated Annealing,* Morgan Kaufmann, 1987.

[2] Coello, C.C.A., "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques," *Knowledge and Information Systems,* vol.1(3), pp.129-156, 1999.

[3] Cucchiara, R., "Genetic Algorithms for Clustering in Machine Vision", *Machine Vision and Applications*, vol.11, pp.1-6, 1998.

[4] Deb, K., "Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems", *Evolutionary Computation*, vol.7(3), pp.205-230, 1999.

[5] Demiriz, A., Bennett, K.P., and Embrechts, M.J., "Semi-Supervised Clustering Using Genetic Algorithms", in *Proc. Artificial Neural Networks in Engineering*, 1999.

[6] Eiben, G., and van Hemert, J., "SAW-ing EAs: Adapting the Fitness Function for Solving Constrained Problems", in D. Corne, M. Dorigo, and F. Glover (eds.) *New Ideas in Optimization,* McGraw-Hill, 1999.

[7] Fonseca, C.M., and Fleming, P.J., "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Proc.* 5^{th} *Int. Conf. on Genetic Algorithms*, pp.416-423, Morgan Kaufmann, 1993.

[8] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, 1989.

[9] Goldberg, D.E., Deb, K., Kargupta, H., and Harik, G., "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms", in *Proc.* 5^{th} *Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 1993.

[10] Gordon, A.D., "A Survey of Constrained Classification", *Computational Statistics & Data Analysis*, vol.21, pp.17-29,1996.

[11] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[12] Kargupta, H., "SEARCH, Polynomial Complexity, and the Fast Messy Genetic Algorithm", Doctoral Dissertation, University of Illinois at Urbana-Champaign, 1995.

[13] MacQueen, J., "Some Methods for Classification and Analysis of Mulivariate Data", in 5^{th} *Berkeley Symposium*, vol.1, pp.281-297, 1967.

[14] Murthy, C.A., and Chowdhury,N., "In Search of Optimal Clusters Using Genetic Algorithms", *Pattern Recognition Letters*, vol.17, pp.825-832, 1996.

[15] Pareto, V., *Cours D'Economie Politique*, vols.I & II, F. Rouge, Lausanne, 1896.

[16] Salah, A.A., Redman, S., Kovacs, G., "Clustering under Constraints with Genetic Algorithms", http://www.cmpe.boun.edu.tr/~salah/santafe/presentation.ppt, 2002

[17] Sarkar, M., Yegnanarayana, B., and Khemani, D., "A Clustering Algorithm Using an Evolutionary Programming-Based Approach", *Pattern Recognition Letters*, vol.18, pp.975-986, 1997.

[18] Sastry, K., and Goldberg, D.E., "Genetic Algorithms, Efficiency Enhancement, and Deciding Well with Differing Fitness Bias Values", IlliGAL TR-2002003, University of Illinois at Urbana-Champaign, January 2001.

[19] Sastry, K., and Goldberg, D.E., "Analysis of Mixing in Genetic Algorithms: A Survey", IlliGAL TR-20022012, University of Illinois at Urbana-Champaign, April 2002.

[20] Schaffer, J.D., "Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms", Doctoral Dissertation, Vanderbilt University, Nashville, Tennessee, 1984.

[21] Schaffer, J.D., "Multiple Objective Optimization with Vector Evaluated Genetic Algoritmhs", in *Genetic Algorithms and their Applications: Proc.* 1^{st} *Int. Conf. on Genetic Algorithms,* pp.93-100, Lawrence Erlbaum, 1985.

[22] Tamaki, H., Kita, H., and Kobayashi, S., "Multi-Objective Optimization by Genetic Algorithms: A Review", in *Proc. 1996 Int. Conf. on Evolutionary Computation,* pp.517-522, Nagoya, 1996.

[23] Valenzuela-Rendón, M., and Uresti-Charre, E., "A Non-Generational Genetic Algorithm for Multiobjective Optimization", in *Proc.* 7^{th} *Int. Conf. on Genetic Algorithms*, pp.658-665, San Mateo, California, 1997.

[24] Van Veldhuizen, D., and Lamont, G.B., "Multiobjective Evolutionary Algorithm Research: A History and Analysis", Technical Report Number TR-98-03. Wright-Patterson AFB, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Ohio, 1998.

[25] Zitzler, E., Deb, K., and Thiele, L., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, vol.8(2), pp.173-195, 2000.