Continuous Space Pattern Reduction for Genetic Clustering Algorithm

Chun-Wei Tsai
Applied Geoinformatics
Chia Nan Univ. of Pharmacy & Science
Tainan, Taiwan, R.O.C.
cwtsai0807@gmail.com

Tzu-Yuan Lin
Computer Science and Engineering
National Sun Yat-sen Univ.
Kaohsiung, Taiwan, R.O.C.
m983040031@gmail.com

Ming-Chao Chiang
Computer Science and Engineering
National Sun Yat-sen Univ.
Kaohsiung, Taiwan, R.O.C.
mcchiang@cse.nsysu.edu.tw

Chu-Sing Yang
Electrical Engineering
National Cheng Kung Univ.
Tainan, Taiwan, R.O.C.
csyang@ee.ncku.edu.tw

Tzung-Pei Hong
Computer Science & Information Engineering
National Univ. of Kaohsiung
Kaohsiung, Taiwan, R.O.C.
tphong@nuk.edu.tw

ABSTRACT

We have recently proposed a highly effective method for speeding up metaheuristics in solving combinatorial optimization problems called pattern reduction (PR). It is, however, limited to problems with solutions that are either binary or integer encoded. In this paper, we proposed a new pattern reduction algorithm named continuous space pattern reduction (CSPR) to overcome this limitation. Simulations show that the proposed algorithm can significantly reduce the computation time of k-means with genetic algorithm (KGA) for solving the data clustering problem using continuous encoding.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; G.1.6 [**Mathematics of Computing**]: Optimization

Keywords

Genetic algorithm, clustering, pattern reduction

1. INTRODUCTION

Several methods for speeding up clustering algorithms by eliminating unnecessary computations have been proposed [3, 4, 6]. The underlying idea of dimensional reduction [3] is to remove features that have small or no influence on the final result from the original data set to obtain a reduced data set in such a way that the size of the reduced data set will be much smaller. Unlike the dimensional reduction method, the basic idea of centroid reduction [4] is to find out clusters the centroids of which remain the same as those in the previous iteration, so that the distance calculations between all the patterns and these static centroids can be avoided to speed up the clustering process. Fundamentally different from the dimensional reduction method and the centroid reduction method, the pattern reduction (PR) method [6] works by eliminating at each iteration *patterns* (or subsolutions) that are unlikely to change their membership thereafter to speed up the clustering process.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

2. THE PROPOSED ALGORITHM

Like the PR, the proposed algorithm consists of two operators: the detection operator and the compression operator. Unlike the PR, the detection operator of CSPR is divided into two steps. How it works for a clustering algorithm with floating-point representation is as shown in Fig. 1.

```
1.  // Step 1. Check to see if all the subsolutions at each locus have the same
    // or similar value.
2.  for j = 1 to n // for each subsolution
3.     let τ_j = 1
4.     for i = 2 to m // for each solution
5.        if |s_ij − s_1j| ≤ T_a then let τ_j = τ_j + 1
6.     endfor
7.  endfor
8.  // Step 2. Check to see if applying each of the subsolutions found at each
    // locus in Step 1 to all the subsolutions at the same locus in turn gives the
    // same or similar fitness.
9.  let R = ∅.
10. for j = 1 to n
11.    if τ_j = m then
12.       let f = true
13.       for i = 1 to m
14.          for k = 1 to m
15.             if k ≠ i then
16.                let s'_k = s_k
17.                let s'_kj = s_ij
18.             endif
19.             if |F(s_k) − F(s'_k)| > T_b then let f = false
20.          endfor
21.       endfor
22.       if f = true then let R = R ∪ {s_1j, s_2j, ..., s_mj}
23.    endif
24. endfor
25. output R
```

Figure 1: Outline of the detection operator.

At step 1, as shown on lines 2 to 7 in Fig. 1, the detection operator of CSPR is aimed at finding out subsolutions at the same locus that have the same or similar value. These subsolutions are called the candidate subsolutions because they may have reached the final state (i.e., they may end up being part of the final solution) faster than the other subsolutions or before the maximum number of iterations is reached. At the end of step 1, the count τ_j will tell whether all the subsolutions at locus j, denoted s_{ij}, are approximately equal to s_{1j} or not. Note that in Fig. 1, m and n denote, respectively, the population size and the number of subsolutions.

Table 1: Relative performance of the clustering algorithms evaluated in this paper (with respect to k-means).

Method	GKA	PREGKA	KGA	CSPR
Average Time	1,292.45%	-54.19%	853.58%	-62.81%
Average SSE	-1.15%	2.63%	-1.67%	-1.13%

N.B.: For both Time and SSE, a more negative value implies a greater enhancement.

Because it may take a long time for all the subsolutions at locus j to converge to the same value, a predefined threshold T_a is required to speed up checking whether all the subsolutions s_{ij} are approximately the same or not. If $|s_{ij} - s_{1j}| \leq T_a$, $\forall i \neq 1$, the proposed algorithm will consider all the subsolutions at locus j as the same, thus being the candidate subsolutions for compression

At step 2, as shown on lines 9 to 25, the detection operator of CSPR will check whether applying each of the candidate subsolutions s_{ij} to the other subsolutions at the same locus, i.e., s_{kj} for $k \neq i$, will affect the fitness values or not. If the results of applying each of the candidate subsolutions s_{ij} to the other subsolutions at the same locus show that the fitness values will not be affected, then these candidate solutions will have a high probability of ending up being part of the final solution. Same as in step 1, the performance issue requires us to predefine another threshold T_b for checking the results of applying each of the candidate subsolutions s_{ij} to the other subsolutions at the same locus. If the difference between the new solution s'_j and the original solution s_j is less than T_b, i.e., $|F(s_j) - F(s'_j)| \leq T_b$, then the proposed algorithm will consider it as having reached the final state and thus can be compressed and removed. That is, candidate subsolutions passing this check mechanism will be compressed and removed by the proposed algorithm.

In summary, a two-step detection operator is employed to check the subsolutions to see whether they have reached the final state or not, as shown in Fig. 2. The first step is aimed at finding out subsolutions at each locus that have the same or similar value, or to filter out subsolutions that have "different" values, so that they can be considered as the candidate subsolutions for compression. The second step is aimed at ensuring that the candidate subsolutions found in the first step have reached their final state so that any further computations are essentially a waste and thus can be compressed and eliminated.

3. RESULTS AND CONCLUSION

Ten different kinds of data sets from UCI (Iris, Wine, SPECT, SPECTF, Ecoli, Haberman, Liver-disorders, Balance-scale, Yeast, and Abalone) are used to evaluate the performance of the clustering algorithms compared in this paper. Each algorithm is carried out for 30 runs, and for each run, the number of iterations is set equal to 1,000. The thresholds T_a and T_b are defaulted to 10^{-8}. By using k-means as the baseline for comparison, the simulation results described in Table 1 show that CSPR outperforms not only the pattern reduction [2] algorithm we proposed previously which uses discrete encoding but also GA-based clustering algorithms such as [5] and [1] in terms of the computation time. The simulation results also show that in terms of the quality, CSPR not only beats k-means and PREGKA; it narrows down the gap between GA-based clustering algorithms and the PR enhanced versions too. Moreover, with minor modification, the proposed algorithm can be applied to many other efficient heuristic algorithms. Also, it is interesting to note that all the algorithms with PR are faster than the k-means algorithm because the operators of k-means are performed 1,000 times each run for all the subsolutions whereas the operators of all the algorithms with PR are performed for only some of the subsolutions

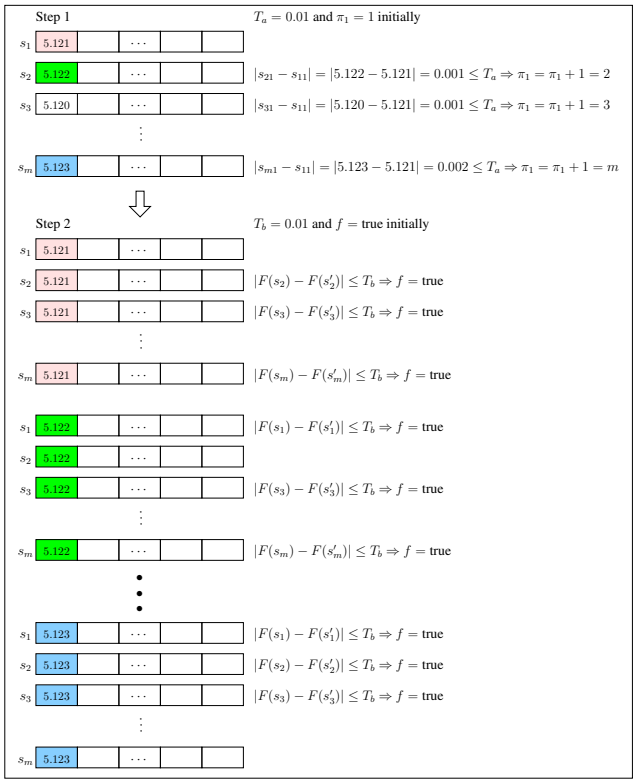

Figure 2: Example showing how the detection operator of the proposed algorithm works.

because most of the subsolutions have reached the final state and thus are compressed and removed before the maximum number of iterations is performed.

Acknowledgment

This work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC100-2218-E-041-001-MY2, NSC98-2221-E-110-049, and NSC99-2221-E-110-052.

4. REFERENCES

[1] S. Bandyopadhyay and U. Maulik. An evolutionary technique based on k-means algorithm for optimal clustering in R^N. *Information Sciences*, 146(1-4):221–237, 2002.

[2] M. C. Chiang, C. W. Tsai, and C. S. Yang. A time-efficient pattern reduction algorithm for k-means clustering. *Information Sciences*, 181(4):716–731, 2011.

[3] C. Ding and X. He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, volume 69, pages 225–232, 2004.

[4] T. Kaukoranta, P. Fränti, and O. Nevalainen. A fast exact GLA based on code vector activity detection. *IEEE Transactions on Image Processing*, 9(8):1337–1342, 2000.

[5] K. Krishna and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on System, Man and Cybernetics—Part B:Cybernetics*, 29(3):433–439, 1999.

[6] C. W. Tsai, S. P. Tseng, M. C. Chiang, and C. S. Yang. A framework for accelerating metaheuristics via pattern reduction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 293–294, 2010.