Memetic algorithms applied to clustering

A genetical engineering endeavour

José A. Magaña Mesa [jose\_a\_magana@hotmail.com](mailto:jose_a_magana@hotmail.com)

*Abstract*— (*Abstract*)

Keywords— (key words)

# Problem statement and goals

Clustering of complex networks, also known as Community Detection, is a technique that has been approached from plenty of different perspectives and different methods have been developed [1]: hierarchical, graph, partitional, spectral,… Many of these methods are based on mathematical models that explode physical and mathematical properties and metrics of the network and hence the results are heavily supported by theoretical grounds. The methods can be considered “exact” in the sense that they produce a result that is the deterministic and mathematical outcome of applying a certain model. These techniques are though quite often computationally expensive.

On the other hand, Genetic Algorithms (GA) can generate, for many problems and specifically for Clustering problems, solutions that are reasonably good and in many cases match or improve the Best Known Results (BKR) for some well-known datasets. GA is though a wide category with different variations being Memetic Algorithms (MA) [3] one of them. MA is a “mutation” of GA in which the mutation procedure that is applied to the result of combination of existing solutions is replaced by a local search procedure that applies domain knowledge in a local search procedure to modify the individual to improve its fitness value. If GA can be compared to natural reproduction, MA through this local search incorporates to the individuals the capacity to learn, each of this improvements obtained through the local search is a ‘meme’ [2], something that the individual learns (imitates in the original: ” *so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation*”) during his lifetime and makes him better. Local search only incorporates these memes when they contribute to improve the individual; it is a selective evolution process.

This use of selective evolution and in general the local search procedure to select the changes that the individual takes is not usually applied in the recombination operator and the combination is done following much more stochastic techniques that while they contribute to keep the diversity of the individuals in the MA population do not foster the quick improvement of the population and hence, the discovery of good solutions.

In the same way that Genetics Engineering[4] (GE) (referred to the medical and biology field) is able to introduce changes in the genes of an embryo so that parents can choose its gender or guarantee that will not suffer specific genetic diseases, in the computing homologous, for problems, as Clustering, where there is a large knowledge about the problem, if the genes are encoded in an appropriate way, a recombination algorithm can be used to extract from the parents the best genes and combine them to produce an offspring whose fitness is better than its parents with a much higher probability than stochastic recombination would provide. Same as GE is considered by some a Pandora box of humans playing God without being aware of the side effects of gene manipulation, there are also risks on the approach of GE in computing field, as some of the design criteria well established in GA or MA may not hold for GE.

Keeping this idea in mind and taking as starting point two papers recently published regarding the use of MA in Clustering, described in II Previous work, my objective has been to design an algorithm that maintaining the basic scheme of MA algorithms incorporates in the recombination operator, domain knowledge criteria regarding networks structure. The implementation has been done from scratch using Python and the NetworkX library [11], apart from some other modules for common calculations (numpy, pyparsing) , that has only been used to store the graph structure and retrieve neighbor nodes and edges as required by the algorithm. Everything else has been coded for the project.

The design of the algorithm has followed an iterative approach on which changes have been introduced to compensate or fix the drawbacks detected during the design phase. For the design phase, a simple well-known network, Zachary Karate Club [5], has been used as testing field. It is a simple network compared to other datasets in use but large and complex enough for not being a trivial problem.

Once the algorithm has been considered to overcome reasonably well the common issues to consider in this kind of problems the resulting algorithm has been used with the same network and a second network to proof its performance. The second network is the Dolphins network [6]. This is a larger network than the previous one but still small to keep computational costs limited and guarantee that any experiment can be executed in the available time for this project.

The ideas used are not very sophisticated and they come as very straightforward for any expert in the field so, it is my assumption that no success

# Previous work

## A little bit of background

To hold common ground and future reference, let’s remind what the traditional structure and parameters of a MA is:

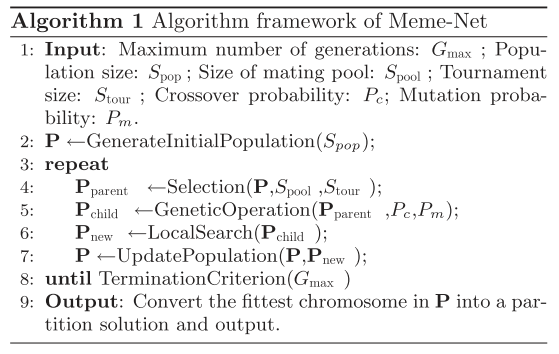


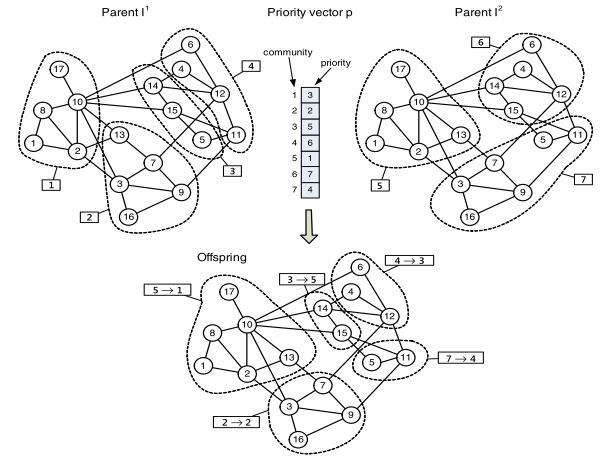
Figure 1 MA algorithm

The recombination algorithm used in one of the papers used as reference and also in my implementation is named Partition Crossover (PX) [12]. The algorithm generates from two different complete partitions of the same network a new complete partition for the network. For doing so the different clusters (genes) in the two networks are combined in a single list and the list traversed. In this configuration each node will be in the list in two clusters, one proceeding from each of the original networks. As the list of clusters is traversed, the cluster is added to the new partition being created and the nodes part of this cluster are removed from the clusters still on the list. In a greedy version of the algorithm, GPX, the clusters are ordered so that, at each iteration, the cluster with higher cardinality is chosen.

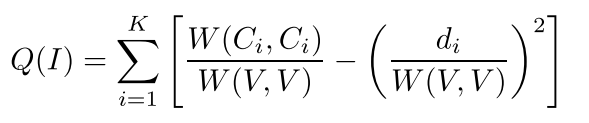
Coming back to PX, nodes that belong to the same cluster in both input partitions (parents) will also be in the same partition in the offspring individual. While nodes in different clusters will be added in different steps and will remain in different clusters. Just a bit of thinking on the algorithm will bring us to some conclusions that are interesting for the design of our algorithm:

1. The order in which the clusters are considered changes the result.
2. The number of clusters in the offspring will not be less than the minimum number of clusters in the parents.
3. The resulting number of clusters can be greater than the maximum number of clusters in the parents
4. The combination of one element with itself generates the same element: A \* A = A

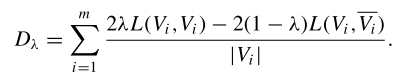
An example, extracted from [9] is shown on the application of the PX operator for two partitions and a certain partition order (priority):



Our driver is to use domain knowledge information in our problem. We will require then a metric to measure the clustering quality that we will use as fitness function. The metric must be calculated for the whole partition but also for each of the clusters in the partition. The problem here is not to find one but to choose one, as the literature [1] refers or provides many different Modularity (Q) functions that measure the cohesion of a clustering. To limit our search, benchmark and do not deviate prematurely much from a known good solution for the purpose of the project the modularity function from [9] has been chosen:

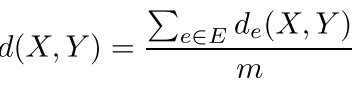
 (Equation 1: Modularity)

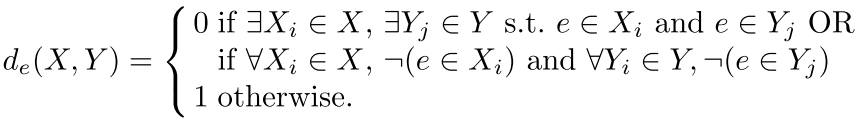
A second option could have been [10], but it is more parametrical and attacks multi-resolution clustering while its λ-free version does not reach the BKR for the problems being used:



An important factor in any GA algorithm is population diversity. The UpdatePopulation() method must consider this in addition to individual fitness. For this, we require a distance function that can be used to compare the similarity of two individuals beyond its fitness value.

A common distance function is the Edge Rand Index (ERI) that measures how many coincidences are there for edges of the graph whose ends are in the same cluster in both partitions or none of the partitions. The definition can be seen in [9] as:



 (Equation 2: Edge Rand Index)

ERI being a ratio is bounded [0,1] while Q takes values in the range [-0.5,1].

## Loans and inspirations

The first, and main algorithm, is MA-COM [9]. This MA algorithm uses in its Genetic Operation (Recombination) a Priority-based Partition Crossover (PPX) operator in which the genes that correspond to clusters in the solution are sorted randomly.

A variation of PPX has been implemented. Clusters priority is given according to its individual modularity calculated using (Equation 1).

The initialization of the population is, in my humble opinion, the weak point of the algorithm. The BGLL [13] algorithm that is able to generate for many of the datasets used optimal or close to optimal solutions is used what really makes MA-COM not a Clustering algorithm but more an “optimizer” for existing good clustering approximations.

The second algorithm is MEME-NET [10]. MEME-NET instead uses a more reasonable initialization procedure where initially each node is a cluster and randomly a certain number of nodes are chosen and all its neighbors in the graph assigned to the cluster the node belongs to. This has been used as initialization in the implemented algorithm. This method requires deciding how many times a node is going to be chosen. The paper uses a factor of 0.2 times the number of nodes in the graph that has also been fixed for the implementation done.

The implementation for the UpdatePopulation() while inspired in the MEME-NET has been modified significantly. MEME-NET uses a parameter δ

End Criterion

For the two networks used in the experiments, it is required not only to obtain the network but also to know the optimal partition. This permits to benchmark our results. The ground truth for the Zachary Karate Club has been obtained from [10] while the ground truth for Dolphins has been obtained from [8] and a correction/clarification provided by Dr. Gach (coauthor of [10])

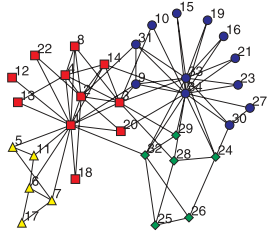


Figure 2 Zachary Karate Club Ground Truth



Figure 3 Dolphins

# The CI Methods

## Chromosome coding

Individuals in the population must be coded in a way that enables the operators for combination and mutation to act efficiently and guarantee that the result of the operations is also a solution for the problem. To achieve this, the coding chosen uses a tuple with the following components:

* Modularity of the solution, in the range 0 to 1.5 (after applying an offset as explained in B Selection)
* Dictionary mapping nodes to clusters, 1 to 1. For the Dolphins ground truth:

{0: 4, 1: 1, 2: 4, 3: 5, 4: 3, 5: 1, 6: 1, 7: 1, 8: 5, 9: 1, 10: 4, 11: 3, 12: 2, 13: 1, 14: 2, 15: 3, 16: 2, 17: 1, 18: 3, 19: 1, 20: 4, 21: 3, 22: 1, 23: 3, 24: 3, 25: 1, 26: 1, 27: 1, 28: 4, 29: 3, 30: 4, 31: 1, 32: 1, 33: 2, 34: 2, 35: 3, 36: 5, 37: 2, 38: 2, 39: 5, 40: 2, 41: 1, 42: 4, 43: 2, 44: 4, 45: 3, 46: 2, 47: 4, 48: 1, 49: 2, 50: 2, 51: 3, 52: 2, 53: 2, 54: 1, 55: 3, 56: 1, 57: 1, 58: 2, 59: 5, 60: 1, 61: 2}

Where the first integer (key dictionary) in each pair refers to the node identifier and the second to the cluster number.

* Dictionary mapping clusters to nodes, 1 to N. For the Dolphins ground truth:

{1: [1, 5, 6, 7, 9, 13, 17, 19, 22, 25, 26, 27, 31, 32, 41, 48, 54, 56, 57, 60],

2: [12, 14, 16, 33, 34, 37, 38, 40, 43, 46, 49, 50, 52, 53, 58, 61],

3: [4, 11, 15, 18, 21, 23, 24, 29, 35, 45, 51, 55],

4: [0, 2, 10, 20, 28, 30, 42, 44, 47],

5: [3, 8, 36, 39, 59]}

Where the first integer is the cluster number, not necessarily consecutive, for construction reasons, and the list corresponds to the nodes in that cluster.

* List of the modularity of each of the clusters in the partition in the same order than the clusters to nodes mapping.

## Selection

None of the two referenced algorithms describe the implementation used for the Selection method. The implementation used selects two members of the population based on a probability that is proportional to the modularity of the individual.

For this being possible an offset (+0.5) has been applied to the modularity to make it non-negative. All the modularities in the population have been added (total\_q) and a random number generated between 0 and total\_q to choose two parents. These two parents cannot be the same element so the selection must be done without replacement.

## Combination

As explained in the previous section, the Combination algorithm used is a variation of the PPX. The clusters in the parents are sorted according to its modularity. The list is then traversed and the PX algorithm is applied. The calculation of the modularity is static; the modularity is not recalculated for the clusters when nodes are removed because the nodes are part of a cluster already added to the output partition. This is a design decision motivated by the intuition that it is better to incorporate a part of a good cluster existing in the parents than a not so good whole cluster whose incorporation would continue fragment the former one.

From the operation, it is obtained a partition that is made up of the common denominator of the parents and a list of small clusters that includes the remains of this common denominator extraction that the PX performs. In particular, a list of clusters with one node exists. In the initial population this list is long because the initialization method applies the merge as many as 20% of the number of nodes. As the population evolves, the number of single nodes gets reduced. Also single nodes appear as a result of combining very different parents.

The existence of these single nodes will be exploded in the LocalSearch method of the algorithm.

## LocalSearch

Shuffle nodes

Pm

## UpdatePopulation

No repeated elements

Take Over time --- Replace closer element vs replace closer if delta min or replace further (if better)

## TerminationCriterion

fixed number of generations

## Parameters

# Results and Discussion

# Extensions, strengths and weaknesses

No more parametric than GA

# Conclusions

1. Table Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. *(Table footnote)*

##### References

1. Fortunato, S. (2010). "Community detection in graphs." Physics Reports 486(3): 75-174.
2. Moscato, P. and C. Cotta (2003). A Gentle Introduction to Memetic Algorithms. Handbook of Metaheuristics. F. Glover and G. Kochenberger, Springer US. **57:** 105-144.
3. Neri, Ferrante, Carlos Cotta, and Pablo Moscato, eds. "Handbook of memetic algorithms." (2011).
4. http://en.wikipedia.org/wiki/Genetic\_engineering
5. W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33, 452-473 (1977).
6. A.Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behavioral Ecology and Sociobiology 54, 396-405 (2003).
7. Wu, Qin, et al. "“Follow the Leader”: A Centrality Guided Clustering and Its Application to Social Network Analysis." The Scientific World Journal 2013 (2013).
8. Hao, J.-K. (2012). Memetic algorithms in discrete optimization. Handbook of Memetic Algorithms, Springer**:** 73-94.
9. Gach, O. and J.-K. Hao (2012). A memetic algorithm for community detection in complex networks. Parallel Problem Solving from Nature-PPSN XII, Springer**:** 327-336.
10. Gong, M., et al. (2011). "Memetic algorithm for community detection in networks." Physical review E **84**(5): 056101.

1. <https://pypi.python.org/pypi/networkx/>
2. P. Galinier and J.K. Hao. Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization, 3(4):379–397, 1999.
3. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp., P10008 (October 2008), doi:10.1088/1742-5468/2008/10/P10008

Maria Angelova and Tania Pencheva, “Tuning Genetic Algorithm Parameters to Improve Convergence Time,” International Journal of Chemical Engineering, vol. 2011, Article ID 646917, 7 pages, 2011. doi:10.1155/2011/646917

<http://eislab.gatech.edu/people/scholand/gapara.htm>

### DeJong and Spears, 1990

##### appendix A. proof of theoretical results

##### appendix b. implementation details