Graph Clustering Based Model Building

David Iclănzan[1,2] and Dumitru Dumitrescu[2]

[1] Department of Electrical Engineering,
Sapientia Hungarian University of Transylvania,
Şoseaua Sighişoarei 1C, 547367, Corunca, Romania
david.iclanzan@gmail.com
[2] Department of Computer Science, Babe s-Bolyai University,
Kogălniceanu no. 1, 400084, Cluj-Napoca, Romania
ddumitr@cs.ubbcluj.ro

Abstract. Probabilistic models of high-order statistics, capable of expressing complex variable interactions, have been successfully applied by estimation of distribution algorithms (EDAs) to render hard problems tractable. Unfortunately, the dependence structure induction stage in these methods imposes a high computational cost that often dominates the overall complexity of the whole search process.

In this paper, a new unsupervised model induction strategy built upon a maximum flow graph clustering technique is presented. The new approach offers a model evaluation free, fast, scalable, easily parallelizable method, capable of complex dependence structure induction. The method can be used to infer different classes of probabilistic models.

1 Introduction

Estimation of Distribution Algorithms (EDAs) extend the classical framework of Evolutionary Algorithms (EAs) with a novel approach consisting in learning and exploiting information from selected individuals. Global statistical information is extracted from promising solutions and used to infer a probabilistic model. New solutions are then sampled from the probability distribution model in order to generate the next population.

The search for an appropriate model in EDAs capable of modeling higher order dependencies, requires many model evaluations with regard to the population. Given the implied population sizes as the dimension of the problems increases, the computational cost of model building may quickly exceed economical practicality. Recent benchmarking and profiling results showed that easily more than 90% of EDAs running time may be spent in the model building phase [1].

Recent efforts have aimed making higher order EDAs computationally less expensive. Enhancements and modifications of the original methods considered parallelization [2,3] and hybridization with local search methods [4], the usage of iterative [5] and sporadic model building [6] or incorporation of initial knowledge [7]. More direct approaches aim to reduce the complexity of model building by restricting the search over a reduced set of variables in each epoch [1].

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 506–515, 2010.

Another line of research concentrate on the usage of global statistics extracted from the data to reduce or bypass the number of model accuracy evaluations. The improved Estimation of Dependency Networks Algorithm [8] uses a multivariate dependency network approximation by considering only bivariate statistics. In another work [9], the $O(n^3)$ model building of the Extended Compact Genetic Algorithm (eCGA) is successfully replaced by a variable correlation guided search of linear complexity. Some methods completely avoid the goodness-of-fit evaluations of the models with regard to the data, by clustering a pairwise variable interaction matrix. The Dependency Structure Matrix Genetic Algorithm (DSMGA) [10] and its extension to hierarchical problems DSMGA++ [11] both use dependency structure matrix clustering techniques for linkage learning. These methods still employ a costly search process to find a clustering setting that minimizes a metric based on the Minimum Description Length (MDL) principle. In [12] the authors use the affinity clustering of the sampled mutual information matrix to obtain a marginal product model factorization which is not able to represent overlapping linkages but may suffice for many applications.

In this paper we further explore the confluence between clustering algorithms and EDAs, where graph clustering algorithms are applied to pairwise interaction statistic matrices to reveal dependency structures. We term this class of methods as Graph Clustering assisted EDAs (GCEDAs). We are especially interested in finding efficient clustering algorithms allowing the induction of various probabilistic model classes.

Here, we focus on the class of flow-based graph clustering algorithms as they are know to be relatively fast and simple while some variants still being able to handle overlapping clusters. From the variants built upon this idea we have chosen to use the Markov Clustering Algorithm (MCL) [13], as it has a simple and elegant formulation based on just two operators, proved effectiveness in clustering real-world biological data [14], good documentation and available source code under GNU General Public License[1].

The main technical contribution of this paper lays in showing that given the pairwise interaction map of the variables, a simple *unsupervised* graph clustering algorithm is able to assist qualitative linkage learning by allowing the inference of different probabilistic models like Bayesian Networks, overlapping linkage models and marginal product models. Hard optimization problems, characterized by non-separable, high-order of interactions among the variables, with deceptive subproblems are solved.

The following section provides some preliminaries and a discussion about how an EDA can use the result of graph clustering for probabilistic model building. Section 3 presents an EDA, which implements the ideas and particular clustering techniques discussed in Section 2. Section 4 contains the description of our experimental setup, the results and a discussion of our findings related to the MCL assisted EDA are given in section 5. Finally, Section 6 discusses results and implications of this work and outlines some future work.

[1] `http://www.micans.org/mcl/`

2 Preliminaries

Let $G = (V, E)$ denote the input graph for the graph clustering algorithm, with V and E denoting the node set and edge set respectively. Let A be the $|V||V|$ adjacency matrix, with $A(i, j)$ denoting the weight of the edge between the vertex v_i and the vertex v_j . In our setting, this weight represent the strength of the pairwise interaction between variables, as extracted from the data available for model building. In this paper, the pairwise dependency is quantified by sampled mutual information.

2.1 Graph Clustering Paradigm, Stochastic Matrices and Flows

Maximum flow clustering algorithms rely on the following core idea: by simulating a special flow within a graph, which promotes flow where the current is strong, and reduces flow where the current is weak will reveal the cluster structure within the graph, as the flow across borders between different groups diminish with time, while it increases within the group.

Simulation of flow through a graph is easily done by transforming the adjacency matrix into a column-stochastic square matrix, where each column sums to 1. This matrix, which we denote by M, can be interpreted as the matrix of the transition probabilities of a random walk (or a Markov chain) defined on the graph, where $M(j, i)$ represents the probability (stochastic flow) of a transition from vertex v_i to v_j.

The flow matrix M is obtained by normalizing the columns of the adjacency matrix to sum up to 1. Flow expansion can be simulated by computing powers of the flow (Markov) matrix M.

2.2 Markov Clustering Algorithm

The MCL algorithm [13] is a fast and scalable unsupervised graph clustering algorithm, based on simulation of stochastic flow in graphs. It offers several advantages, like a simple, elegant mathematical formulation, robustness to topological noise [14], support for easy paralellization and adaptation via a simple parameter enables the obtaining of clusters of different granularities.

MCL iteratively simulates random walks within a graph by applying two operators called expansion and inflation, until convergence occurs. At the end of each inflation step a pruning step is also performed, in order to reduce the computational complexity by keeping M sparse.

Intuitively, the MCL process may be regarded as alternative expansion and contraction of the flow in the graph. The expansion step is responsible in spreading the flow out of a vertex to potentially new vertices and with the strengthening of the flow to those vertices which are reachable by multiple paths. This has the effect of enhancing within-cluster flows as there are more paths between the nodes belonging to the same cluster. The inflation operator is responsible for both strengthening intra-cluster flow and weakening inter-cluster flow of current and by this, introducing a non-linearity in the distribution of the flows. At the

beginning the flow distribution is relatively smooth and uniform, but with each iteration it becomes more and more peaked. In the end, all the nodes within a tightly-linked group of nodes will start to flow towards one node within the group, forming star sub-graphs associated with the MCL limits.

The idealized Markov Cluster process, consisting just from the expansion and inflation operators is known to converge quadratically in the neighborhood of so called doubly idempotent matrices [13]. In practice, the numbers of epochs until convergence is reported to be nearly always far below 100.

2.3 Interpretation of MCL Clustering as Dependency Models

MCL iterants M_t are generally diagonally positive semi-definite matrices. Using the property that minors of a diagonally positive semi-definite matrix are non-negative, in [15] it is shown that M_t-s have a structural property which associates a directed acyclic graph (DAG) with each of them. These DAGs generalize the star graphs associated with the MCL limits.

We present several approaches on how the information from MCL iterants can be conveyed in dependency models able to represent and exploit linkages.

In the *first* approach, the DAGs represented by M_t-s are directly used for defining the structure for a Bayesian network, with the edges representing the conditional dependencies between variables. Then, the parameters, which consist of the conditional probabilities of each variable given the variables that this variable depends on, are extracted from the data in the same way as in BOA [16] or EBNA [17]. The obtained Bayesian network will encode the joint probability distribution of the variables and can be used to sample the next generation. This approach presents two small impediments. First, one has to decide from which M_t to construct and use the Bayesian network, thus a few model evaluations against the data still have to be computed. The second issue relates to the rare occasions where a MCL iterant contains cycles. Before performing the parameter extraction, these cycles must be detected and eliminated.

In a *second* approach, DAGs are interpreted as clusters by taking as cores all end nodes (sinks) from the DAG, and by attaching to each core all the nodes that reach it with a flow amount greater than a threshold. This procedure may result in clusters containing overlap. The extracted linkages can be used to perform building block wise crossover like in DSMGA [10] or DSMGA++ [11] or they can be used to build overlapping linkage model based probability distributions.

The *third* approach is the cheapest one, as it deals only with the last iterant, when the stochastic flow matrix M is completely converged. Here, the nodes have found one "attractor" node to which all of their flow is directed, corresponding to only one non-zero entry per column in M. Nodes sharing the same "attractor" node are grouped in clusters. This approach is suitable for modeling non-overlapping building blocks, by building marginal product models as in eCGA. [18].

Excepting the first approach, the dependency structure inferring is completely autonomous, as its does not need to check the model fit with regard to the data.

3 MCL Assisted EDA

Wishing to present an EDA with unsupervised model building, capable of modeling complicated variable interactions, we employ the second interpretation of the MCL iterants to obtain a overlapping linkage model based probabilistic model. We name this algorithm Markov Clustering EDA (MCEDA). The details of the algorithm are presented in the followings.

In this paper, the degree of pairwise dependency between variables is calculated using sampled mutual information between two variables and record into an adjacency matrix A, which will be the input of the graph clustering algorithm.

The transformation of A in a stochastic Markov matrix is handled by the MCL algorithm by normalization of the columns to sum up to 1.

3.1 The Overlapping Linkage Model (OLM)

In MCEDA, the multivariate variable interactions are modeled with the use of overlapping linkage models (OLMs), which closely resembles the marginal product model adopted by the eCGA [18]. The difference is that OLM models subsets of variables jointly as clusters, allowing overlaps, in contrast with partitions, which always divides the variables in collectively exhaustive and mutually exclusive blocks. The clusters can naturally represent building blocks, providing a direct linkage map of the variables, thus we will use this terms interchangeably in the context of OLMs. Clusters together with the marginal distributions over them form the OLMs.

3.2 Dependency Structure Building and Sampling

The clusters that form the basis of the OLMs are extracted from the iterants M_t of the MCL algorithm.

For each node a potential building block is formed, by grouping together all nodes that reach it with a flow amount greater than a threshold F_{min}. Basic clusters of size 1 are only allowed, if the described single position is not contained in any other cluster. After all iterants have been processed, the procedure returns the unique entries of the potential building block list. This sorting must be performed, as the same cluster may be detected several times from different iterants, or even from the same M_t in the rare cases when it contains cycles.

The building block extraction is depicted in Function `ExtractBBs`.

Please note that a practical implementation does not have to store all the iterants for a final batch processing. It is described in this way to keep the presentation clean and simple. A clever building block extraction works in interplay with the MCL, processing each iterant right after it was computed, retaining the unique building blocks in the same fashion.

After the building blocks are determined, their probability distribution is estimated by simply counting the frequencies in the data.

As the performance of the method did not seem to be very sensible on the setting of F_{min}, we use a fixed value of $10e-4$.

Function `ExtractBBs(`*Mlist*`)` returns BBlist

```
1  BBlist ← ∅;
2  //each iterant from the MCL algorithm is processed
3  foreach M_t from Mlist do
4      //for all nodes find significant incoming flows
5      for i ← 1 to size(M_t) do
6          pBB ← find(M_t(i,:) > F_min(i));
7          if length(pBB) > 1 then
8              BBlist ← BBlist ⋃ pBB;
9  BBlist ← unique(BBlist);
```

Algorithm 2: The Markov Clustering EDA

```
1  pop ← RandomInit();
2  repeat
3      ps ← Selection(pop); //select promising solutions
4      {ps ← ReduceEntropy(ps)}; //optionally reduce entropy by LS
5      A ← MutualInformation(ps); //extract global statistics
6      Mlist ← MCL(A); //apply graph clustering
7      BBlist ← ExtractBBs(Mlist); //extract dependency structure
8      freq ← FrequencyCount(BBlist, ps); //compute marginal probabilities
9      olm ← BuildMPM(BBlist, freq); //combine results into a OLM
10     pop ← Sample(olm); //generate a new population using the model
11 until convergence criteria is met ;
```

The building blocks, together with the frequencies obtained from the data form the OLM model. This probabilistic model is sampled by enumerating the building blocks in a random order, and choosing a configuration according to the registered probabilities.

3.3 The Markov Clustering EDA

Starting from a random population, the MCEDA applies the process of evaluation, selection, MCL assisted OLM model-building and sampling until a halting criterion is met, which is usually a combination of several criteria like computational and time resources spent, amount of progress between epochs, the energy of best solution found so far etc.

As the method relies only on pairwise information statistics, the signal of this measure must be relatively strong. A good entropy reduction of the samples used to generate the statistics can be achieved by performing a local search, and/or using a big population size with a high selection pressure.

Algorithm 2 summarizes the structure and workings of the method.

4 Experiments

In this paper the class of additively decomposable functions (ADFs) with deceptive trap subproblems is considered, a test bed which is widely used in the literature as benchmarking problems[16,19,20].

4.1 Test Functions

The *concatenated trap-5* [16] is an ADF based on unitation (number of ones from a binary string) measures, exhibiting a single global optimum in the string formed exclusively from ones.

The input string is partitioned into disjoint shuffled groups of five bits each. A 5-bit trap function is applied to each of the groups and the fitness of the individual is the sum of the contributions of each 5-bit group.

Each subproblem of five variables has two competing schemata which are maximally distant: (0, 0, 0, 0, 0) and (1, 1, 1, 1, 1). The fitness gradient leads towards the string formed by zeros, thus low order statistics, taking into account less than 5 variable statistics, may lead away from the optimal value, deceiving the search.

Non-separability can be introduced by applying a fixed length, circular overlapping scheme between the trap-5 functions [19]. For example, for a problem with 3 subproblems and overlap length $l = 2$, the fitness is given by $trap5(y1y2y3y4y5)+trap5(y4y5y6y7y8)+trap5(y7y8y9y1y2)$, where yi is a random permutation of the variables xi, meant to break the tight linkage. Every building block shares $2l$ variables, l with each of its two neighbor.

4.2 Numerical Results

Experiments are performed with the simple concatenated trap-5 function without overlap (denoted by ctf5o0), with overlap 1 (ctf5o1) and overlap 2 (ctf5o2).

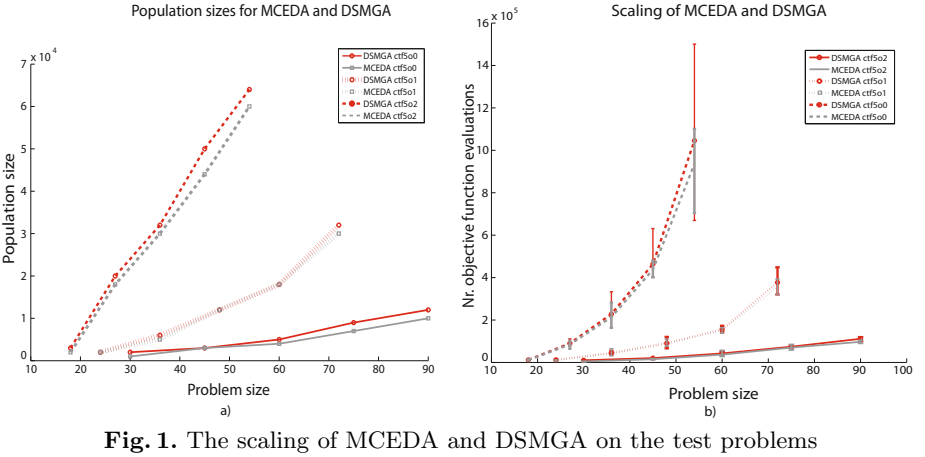

Fig. 1. The scaling of MCEDA and DSMGA on the test problems

In order to test the scalability, for each ADF the number of subproblems k is scaled from 6 to 18 by increments of 3, resulting in various problem sizes up to 90 variables.

The MCEDA performance is compared with the DSMGA having the following parameterization: tournament selection of size 8, crossover probability 1 and no mutation. Population sizes for both algorithms were determined by the bisection method, requiring the methods to converge to the global optima in 10 out of 10 independent runs. The obtained population sizes are depicted in Figure 1 a).

In these experiments, the MCEDA did not use local-search for entropy reduction. The selection operator chooses the winners based on truncation selection, where the best 10% of the population is promoted.

Figure 1 b) presents the scaling of the methods for the different problem types and sizes. The results show a similar scaling of the two methods. MCEDA uses slightly fewer objective function evaluations and works with smaller population sizes than the DSMGA. The performance difference is most likely explained by the following two factors:

- DSMGA uses a crisp value when dealing with variable interactions. The mutual information matrix is transformed in a binary matrix according to a threshold, where two variables are considered fully interacting or completely independent. In contrast, the MCEDA works directly with the normalized mutual information values, which can describe more nuanced, weighted levels or interactions, facilitating the earlier discovery of better models.
- The MCEDA has a better diversity maintenance mechanism as a higher number of building-blocks are extracted due to the usage of early iterants of the MCL process. Furthermore, the method samples according to the exactly observed frequencies in the data. DSMGA may confront the hitchhiking phenomena [21] where some low fitness alleles are promoted together with high-quality building-blocks in above average individuals and the right crossover must be performed to eliminate them.

The MCL graph clustering is much faster than the MDL based clustering used by DSMGA, having a worst case complexity $O(nk^2)$ [13], where k is the pruning factor (at most how many non-zero entries will be in a row of the stochastic matrix - a very small number in practice, $k << n$). Clustering of 5000 nodes by the MCL takes only a few seconds, compared with minutes, in the case of DSMGA model-building.

5 Conclusions and Future Work

The paper proposes a new model inferring method for EDAs, where unsupervised graph clustering algorithms are applied to global statistics extracted from the population data, in order to reveal dependency structures that facilitates the induction of various probabilistic model types. As there is no explicit search for models, with computationally expensive fit-to-data evaluations, this approach has the potential to alienate the model building cost bottleneck in EDAs, enabling them to scale up to truly large problem sizes. Graph clustering algorithms

are usually highly paralellizable, some are able to work in a decentralized, distributed fashion and they are known to be able to cluster graphs containing millions of nodes.

Test results show that the method is able to efficiently solve well known benchmark problems, exhibiting deceptiveness and non-separable building blocks. The strength of the algorithm is also its weakness: as it relies solely on pairwise interaction information, this information needs to be of good quality.

The great advantage of the presented method is that it allows the induction of various probabilistic model classes.

Future work will advance the study and exploration of the GCEDA framework by interpreting clusterings as directed acyclic graphs, in order to build Bayesian networks. Other research will focus in the development of highly parallel model building and large scale optimization.

Acknowledgments

This work is supported by the Grant IDEI 508 "New Computational Paradigms for Dynamic Complex Problems" funded by the Ministry of Education, Research and Innovation, Romania and by the Sapientia Institute for Research Programs (KPI).

References

1. Duque, T.S., Goldberg, D.E., Sastry, K.: Enhancing the efficiency of the ECGA. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 165–174. Springer, Heidelberg (2008)
2. Sastry, K., Goldberg, D.E., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584. ACM, New York (2007)
3. Ocenásek, J., Schwarz, J.: The parallel bayesian optimization algorithm. In: Proceedings of the European Symposium on Computational Inteligence, pp. 61–67. Springer, Heidelberg (2000)
4. Pelikan, M., Hartmann, A.K., Sastry, K.: Hierarchical BOA, cluster exact approximation, and ising spin glasses. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 122–131. Springer, Heidelberg (2006)
5. Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: the incremental bayesian optimization algorithm. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 455–462. ACM, New York (2008)
6. Pelikan, M., Sastry, K., Goldberg, D.: Sporadic model building for efficiency enhancement of the hierarchical BOA. Genetic Programming and Evolvable Machines 9(1), 53–84 (2008)
7. Baluja, S.: Incorporating a priori knowledge in probabilistic-model based optimization. Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, pp. 205–219 (2006)

8. Gámez, J.A., Mateo, J.L., Puerta, J.M.: Improved EDNA(estimation of dependency networks algorithm) using combining function with bivariate probability distributions. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO 2008, pp. 407–414. ACM, New York (2008)
9. Iclanzan, D., Dumitrescu, D., Hirsbrunner, B.: Correlation guided model building. In: GECCO 2009: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, July 8-12, pp. 421–428. ACM, New York (2009)
10. Yu, T.L., Goldberg, D.E., Yassine, A., Chen, Y.P.: Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1620–1621. Springer, Heidelberg (2003)
11. Yu, T.L., Goldberg, D.E.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: GECCO 2006, pp. 1385–1392. ACM Press, NY (2006)
12. Santana, R., Larrañaga, P., Lozano, J.A.: Estimation of distribution algorithms with affinity propagation methods. Technical Report EHU-KZAA-IK-1/08, Department of Computer Science and Artificial Intelligence, University of the Basque Country (January 2008)
13. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, U. of Utrecht (2000)
14. Brohée, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 7, 488 (2006)
15. van Dongen, S.S.: A stochastic uncoupling process for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam (2000)
16. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Wu, A., et al. (eds.) GECCO 1999, Orlando, FL, July 13-17, vol. I, pp. 525–532. Morgan Kaufmann Publishers, San Fransisco (1999)
17. Etxeberria, R., Larranaga, P.: Global optimization using Bayesian networks. In: Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), pp. 151–173 (1999)
18. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report no. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (February 4, 1999)
19. Yu, T.L., Sastry, K., Goldberg, D.E.: Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1217–1224. ACM, New York (2005)
20. Correa, E., Shapiro, J.: Model complexity vs. performance in the bayesian optimization algorithm. In: Parallel Problem Solving from Nature-PPSN IX, pp. 998–1007 (2006)
21. Mitchell, M., Holland, J.H.: When will a genetic algorithm outperform hill climbing? In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo, CA, USA, p. 647. Morgan Kaufmann, San Francisco (1993)