Multiple Ant Colony System for Substructure Discovery

Oscar Cordón[1], Arnaud Quirin[1], and Rocío Romero-Zaliz[2]

[1] European Centre for Soft Computing, Mieres (Asturias), Spain
{oscar.cordon, arnaud.quirin}@softcomputing.es
[2] Dept. of Computer Science and Artificial Intelligence,
University of Granada, Granada, Spain
rocio@decsai.ugr.es

Abstract. A system based on the adaptation of the search principle used in ant colony optimization (ACO) for multiobjective graph-based data mining (GBDM) is introduced in this paper. Our multiobjective ACO algorithm is designed to retrieve the best substructures in a graph database by jointly considering two criteria, support and complexity. The experimental comparison performed with a classical GBDM method shows the good performance of the new proposal on three datasets.

1 Introduction

Graph-based data mining (GBDM) involves an effective and efficient manipulation of relational graphs towards discovering important patterns [13]. It is now an established area which allowed the solving of a significant number of problems such as analysis of micro-array data in bioinformatics, pattern discovery in a large graph representing a social network, and analysis of transportation networks, among many others [5]. Likewise, *Pareto-based multiobjective search strategies* [2] have also gained much importance in data mining and machine learning communities. That is due to the advantage for the user of retrieving a Pareto set composed of multiple non-dominated solutions with a different trade-off in the satisfaction of some conflicting learning problem objectives [11]. Nevertheless, up to our knowledge, the idea of performing GBDM within a *multiobjective optimization* (MOO) framework, which seems to be a natural extension, has not been widely explored in the specialized literature.

MOO basics are not described in this contribution due to a lack of space, but the interested reader is referred to [2, 6, 3]. In short, MOO problems are characterized by several conflicting objectives which have to be simultaneously optimized [2]. The goal is to find a set of solutions, described by what is called a *decision vector*, which are superior to all the reminder and equally preferable among them, because an improvement in one objective/dimension will degrade the solution in another one. Those solutions constitute the so-called Pareto-optimal set or non-dominated solution set.

This contribution is aimed to bridge the latter gap by proposing a *multiobjective ant colony optimization* (MOACO) algorithm [9] to perform multiobjective

GBDM. This novel application of MOACO is however quite natural since, as described in [8], mining a graph database can be modeled as a search in the lattice of all possible subgraphs (also called substructures). Hence, considering the use of an ACO algorithm to perform GBDM is a rather meaningful idea as these family of metaheuristic approaches are based on building solutions to combinatorial optimization problems by exploring a construction graph representing the problem space. In this way, the graph database lattice itself becomes a natural representation for the construction graph.

The method introduced in this paper is based on the classical *multiple ant colony system* (MACS)[3] [1], although any other MOACO algorithm could be considered [9]. Our multiobjective graph mining algorithm, consequently named *multiple ant colony system for substructure discovery* (MACSD), is expected to optimize two conflicting goals (viz. support and complexity) during the evaluation of the discovered subgraphs. In such a way, a Pareto set of non-dominated and meaningful substructures extracted from a graph database can be found in a single run.

The good performance of MACSD will be demonstrated when benchmarking it against the classical Subdue GBDM method [4] in an experimental study considering an artificial and two real-life datasets. Subdue [4] is one of the most extended methods in the area of GBDM [13, 8] for tasks as frequent substructure discovery, graph compression and hierarchical clustering. It is based on a classical beam search driven by an heuristic, the minimum description length (MDL) principle.

The paper is organized as follows. Section 2 describes our novel methodology. Section 3 shows the performance of the MACSD algorithm on the three datasets. Finally, some conclusions are given in the Section 4.

2 Our Proposal: A Multiple Ant Colony System for Substructure Discovery

In this section, we will describe the main components of our proposal, based on the MACS algorithm. To do so, the first subsection is devoted to introduce the problem representation, i.e., the mapping of the substructure mining task to a combinatorial optimization problem representation that can be used by the artificial ants to build solutions. The second subsection will review the generic structure of our MACSD algorithm. It will also describe some specific issues related to its customization to the introduced problem representation.

2.1 Problem Representation

The design decisions taken to represent the graph mining task in such a way it can be tackled by any ACO algorithm are described below. We have followed

[3] Notice that, the term *Multiple* in MACS refers to the handling of multiple objectives and not to the use of multiple ant colonies.

the standard nomenclature and refer the reader to Dorigo and Stützle's book [7] for more information.

Construction graph. As said, the ACO algorithm will take advantage of the fact that substructure mining is based on exploring a graph (the lattice representing all possible substructures). Hence, the construction graph traveled by the ants, $G_C = (C, L)$, constitutes a representation of the substructure lattice $G = (N, A)$. The set of solution components C corresponds to the set of the database graph arcs A. There is one node ij in the construction graph for each of the existing arcs between every database graph node (i, j) (at most, $|C| = n^2$). The connections L fully link the components, i.e., $|L| = |C|^2$. In this way, the construction graph includes all the possible substructures of the original lattice, i.e. all substructures have at least a support of 1. A feasible solution S generated by an ant when traveling G_C is a set of arcs (solution components) of any dimension composing a connected substructure G_S.

Constraints. The constraints enforce that a valid connected substructure included of the substructure lattice is built. Hence, they will depend on the specific kind of substructures which are to be extracted from the database (for example, they will be different in case we are extracting subgraphs or subtrees). The constraints are implicitly enforced through the solution construction process followed by the ants by properly defining the feasible neighborhood N_i^k of an ant k in node i at each construction step.

Objective functions. The multiobjective substructure discovery problem deals with the maximization of the extracted substructures complexity and support. The final aim is to uncover a non-dominated solution set composed of a variety of substructures with different trade-offs between complexity and support, which is not possible if only a single-objective algorithm such as Subdue [4] is considered.

Let $S = (N_S, A_S)$ be a feasible substructure, with $N_S \subseteq N$ being its set of nodes and $A_S \subseteq A$ being its set of arcs. We can mathematically formulate our two objectives as follows:

$$f_1(S) = Complexity(S) = \frac{|N_S| + |A_S|}{|N_{G_{max}}| + |A_{G_{max}}|} \tag{1}$$

$$f_2(S) = Support(S) = \frac{\#graphs\ in\ G\ including\ S}{card(G)} \tag{2}$$

with $card(G)$ being the cardinal of the set of graphs G composing the data base and G_{max} corresponding to a graph in G having highest sum of nodes and edges.

Pheromone trails. The pheromone trails τ_{ij} have to memorize the preference of traveling to node ij in the construction graph, i.e., of adding arc (i, j) to the substructure currently built by the ant. Hence, a pheromone trail is associated to each construction graph node ij.

Heuristic information. This information is not considered in the current algorithm version.

Solution construction. Every ant produces a single solution to the problem which corresponds to a specific extracted substructure. The final approximation set P_A built by MACSD will constitute a full solution to the problem since it will provide the user with a non-dominated set of substructures with different trade-offs between support and complexity.

To do so, each ant k starts by selecting an initial construction graph node ij (i.e., an initial database graph arc (i, j)) as its first solution component s_1^k. Instead of uniformly drawing that node, we consider it to be selected according to the following probability distribution:

$$p(ij) = \frac{\tau_{ij}}{\sum_{lm \in C} \tau_{lm}} \tag{3}$$

Therefore, the most visited arcs by the ants in the previous stages of the search are most likely to be selected as initial arcs for the exploration performed by the new ants in the current iteration.

Let $S_h^k = (s_1^k, \ldots, s_h^k)$ be the partial solution (i.e., the partial substructure) built by ant k after h construction steps. Its feasible neighborhood $N(S_h^k)$ is composed of every arc (i, j) (every construction graph node ij) such that:

1. $(i, j) \notin S_h^k$.
2. Either i, j, or both of them are included in S_h^k, i.e., they are the head, the tail, or the head and the tail of some arc included in S_h^k.

2.2 Customization of Multiple Ant Colony System for Substructure Discovery

The MACS algorithm was first proposed for vehicle routing problems [1] as an extension of the classical ant colony system (ACS) [7]. To design our MACSD proposal, we have considered the original definition of MACS and have taken some additional design decisions, which are described as follows.

External Pareto archive initialization and update. We consider an initial set of random substructures of size up to $Size_{\text{M}}$ nodes to constitute the initial Pareto archive. The archive is updated *after each single ant move* and the dominated solutions are removed during each update.

Modified solution construction process. We must deal with the problem of not knowing the size of the optimal solutions in advance. To do so, in each iteration, a fixed percentage γ of the ants in the colony will build their solutions from scratch, and the remaining $1 - \gamma$ ants will randomly select one solution from the current Pareto archive and will start their construction process from it. In addition, at each step, we also decide when to stop the construction process of each ant according to a probability distribution: $p_{stopping}(S^k) = step^k / Size_{\text{M}}$, with $step^k$ being the number of construction steps taken by ant k in the current iteration.

Transition rule. MACSD uses a single pheromone trail matrix, τ. The following expression is considered for the transition rule:

$$ij = \begin{cases} \arg\ \max_{lm \in N(S^k)}\ \tau_{lm}, & \text{if } q \leq q_0, \\ \hat{ij}, & \text{otherwise.} \end{cases} \tag{4}$$

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}}{\sum_{lm \in N(S^k)} \tau_{lm}}, & \text{if } lm \in N(S^k), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Pheromone trails update. Every time an ant travels to the node ij, it performs the local pheromone update as follows: $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$, where ρ is the rate of pheromone evaporation.

In the original MACS algorithm, the initial value for the pheromone trails τ_0 is calculated from a set of heuristic solutions by taking their average costs, $\hat{f}^0$ and $\hat{f}^1$, in each of the two objective functions, f_1 and f_2, and applying the following expression: $\tau_0 = 1/(\hat{f}^0 \cdot \hat{f}^1)$. In our case, we have considered the use of the set of non-dominated solutions composing the initial Pareto archive P_A. τ_0 is then computed from the average values of the latter solutions in the two optimization criteria, complexity and support, $\hat{f}_1$ and $\hat{f}_2$, respectively, by using the previous MACS expression. Of course, the τ_0 value is recomputed after each Pareto archive update.

3 Experiments and Analysis of Results

In this section we analyze the behaviour of the MACSD algorithm by means of various metrics proposed in the EMO literature [3]. Firstly, we describe the datasets and the parameter values, then we report a comparison with Subdue.

3.1 Datasets

We have used three different application domains, an artificial dataset (*shapes*) and two real-world datasets: visual science maps (*scientograms*) and web pages (*www*), which are described as follows:

shapes. This dataset [4] consists of 100 randomly generated stacks of geometrical objects and has a complexity of 500 nodes, 400 directed edges, and 6 unique labels.

scientograms. This dataset [12] is comprised by 10 scientograms of the scientific production of the USA for period 1996-2005 and has a complexity of 2762 nodes, 2769 undirected edges, and 293 unique labels.

www. This real web pages database [10] is available online on the Subdue website[4] and has a complexity of 832 nodes, 885 directed edges, and 511 unique labels which include self-connection edges.

[4] http://ailab.wsu.edu/Subdue/datasets/webdata.tar.gz

3.2 Experiments

Subdue was run 3 times, each time using one of its three different criteria as a goal (namely, complexity, support, and MDL). The results of these three runs were joined in a single Pareto set approximation (only non-dominated solutions are kept). We used the default parameters but the number of solutions to be found was set to 33 for each run, in order to have a maximum of 100 generated solutions. The MACSD algorithm was run 10 times, as a consequence of being non-deterministic. Its parameter values are as follows: 3600s. of execution time, 10 ants, $Size_{\mathrm{M}} = 3$, $\tau_0 = 0.4$, $\rho = 0.2$, $q_0 = 0.2$, and $\gamma = 0.8$. For the *shapes* dataset, we set up some specific parameters: 300s. of execution time, $Size_{\mathrm{M}} = 5$, and $q_0 = 0.5$. Those parameter values were selected from a preliminary experimentation.

The comparison between the two algorithms will be developed by considering three classical evolutionary MOO performance indicators (metrics) [6, 3]: the cardinality of the Pareto set approximation, the area (S) of the Pareto front approximation, and the coverage (C) of the Pareto fronts obtained by each algorithm over those obtained by the other.

3.3 Results

The results obtained in the application of our MACSD and the Subdue algorithm to the three previously described datasets are analyzed as follows:

shapes. This dataset is small enough to be checked exhaustively: 31 non-dominated substructures have to be found, corresponding to only 8 different decision vectors. Subdue finds 16 of them (with 8 different decision vectors), getting a S-metric value of 0.108. MACSD finds 21 of them for all its runs, also obtaining the 8 possible decision vectors and the same S-metric value as Subdue. The comparison between MACSD and Subdue showed that the fronts are equal (the C-metric value is 0 in every case) but MACSD achieved a better diversity of solutions.

scientograms. This real-life dataset is more complex than the *shapes* domain. The S values for MACSD (0.206 in average on the 10 runs) are better than those obtained by Subdue (0.177). The C values obtained when comparing MACSD vs. Subdue (0.94 in all cases) are significantly greater than those obtained when comparing Subdue vs. MACSD (0.361 in average), meaning that MACSD dominates more solutions from Subdue than in the opposite comparison. Subdue achieves a higher value of cardinality (35) than MACSD (10.8 in average) as a consequence of its worst convergence to the optimal Pareto front.

Nevertheless, there are two solutions found by Subdue that MACSD did not reach, corresponding to subgraphs with the smallest support and the largest complexity values. The reason probably comes from the small number of ants allocated to MACSD.

www. This real-life dataset is as complex as *scientograms* and it also contains loops over the same node. The S values for MACSD (0.00331 in all cases) are better than that obtained by Subdue (0.00258). The C values achieved when comparing MACSD vs. Subdue (0.875 in all cases) are much greater than those obtained when comparing Subdue vs. MACSD (0.4 in all cases), meaning that MACSD dominates more solutions from Subdue than in the opposite comparison. Again, Subdue gets a higher value of cardinality (8 vs. 5) due to its worst convergence to the Pareto-optimal front. There is one solution found by Subdue that MACSD cannot reach, probably for the same reason as before.

Finally, a graphical representation of the aggregated Pareto front approximations found for each dataset is shown in Fig. 1. Although we clearly identify the said three solutions (two in *scientograms* and the other in *www*) generated by Subdue which dominate their MACSD counterparts (see the left-most extent of the Pareto fronts), MACSD extracted better Pareto fronts for both domains and found bigger substructures than Subdue for the other extent where the largest possible support value substructures are located.

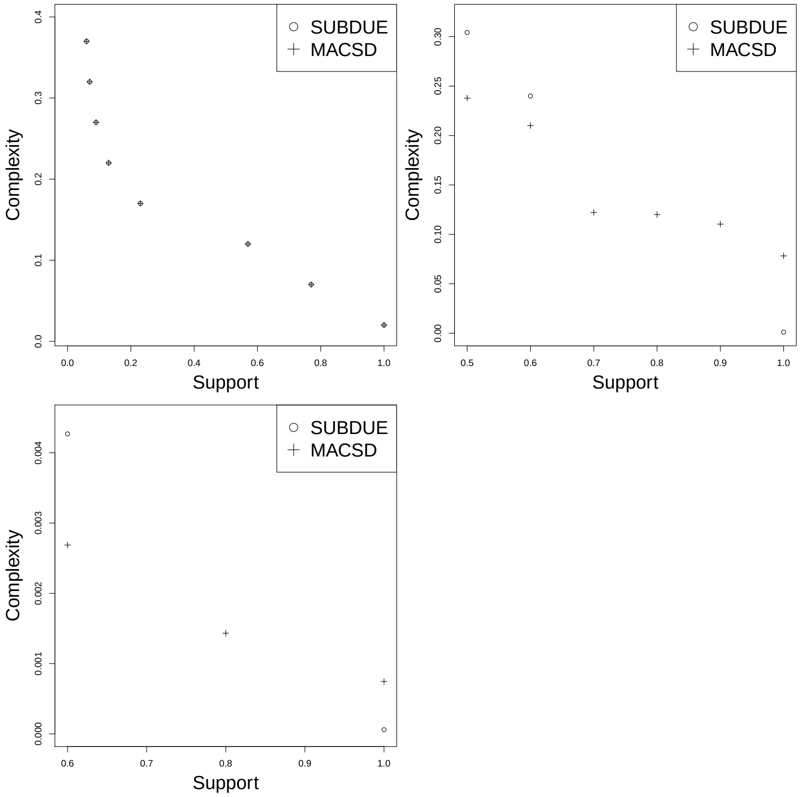

Fig. 1. Graphical representations of the Pareto front approximations for the *shapes*, *scientograms*, and *www* datasets.

4 Conclusion

In this paper, we have shown how the search principle used by ACO can be naturally adapted to perform graph mining. Besides, it has been demonstrated that its combination with a MOO design (e.g. MACS) in a MOACO-based GBDM algorithm designed to retrieve the best substructures from a graph database by jointly considering the support and the complexity can report an outstanding performance. The proposed method, called MACSD, has outperformed the classical Subdue GBDM algorithm on three different datasets. As future works, we plan to design a better heuristic information definition and to test more MOACO schemes.

Acknowledgments. This work has been supported in part by the Spanish Ministry of Science and Technology under project TIN2009-07727, including EDRF fundings.

References

1. Barán, B., Schaerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: IASTED Conf. pp. 97–102. Innsbruck (Austria) (2003)
2. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making Theory and Methodology. North-Holland, Amsterdam (1983)
3. Coello, C.A., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, Berlin (2007)
4. Cook, D., Holder, L.: Graph-based data mining. IEEE Intelligent Systems 15, 32–41 (2000)
5. Cook, D., Holder, L. (eds.): Mining graph data. Wiley, London (2007)
6. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK (2001)
7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. Fischer, I., Meinl, T.: Graph based molecular data mining - an overview. In: IEEE Int. Conf. on Systems, Man and Cybernetics. vol. 76, pp. 4578–4582 (2004)
9. García Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180(1), 116–148 (2007)
10. Gonzalez, J.: Empirical and Theoretical Analysis of Relational Concept Learning Using a Graph Based Representation. Ph.D. thesis, Department of Computer Science & Engineering, University of Texas, Arlington, USA (2001)
11. Jin, Y., Sendhoff, B.: Pareto-based multi-objective machine learning: An overview and case studies. IEEE Trans. Syst., Man, Cybern. C, Appl. Rev 38, 397–415 (2008)
12. Quirin, A., Cordón, Ó., Vargas-Quesada, B., Moya-Anegon, F.: Graph-based data mining: A new tool for the analysis and comparison of scientific domains represented as scientograms (2010), *in press*
13. Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explorations 5(1), 59–68 (2003)