A Multiobjective Variant of the Subdue Graph Mining Algorithm based on the NSGA-II Selection Mechanism

Prakash Shelokar, Arnaud Quirin and Óscar Cordón

Abstract— **In this work we propose a Pareto-based multi-objective search strategy for subgraph mining in structural databases. The method is an extension of Subdue, a classical graph-based knowledge discovery algorithm, and it is thus called MultiObjective Subdue (MOSubdue). MOSubdue incorporates the NSGA-II's crowding selection mechanism in order to retrieve a well distributed Pareto optimal set of meaningful subgraphs showing different optimal trade-offs between support and complexity, in a single run. The good performance of the proposed approach is empirically demonstrated by using a real-life data set concerning the analysis of web sites.**

I. Introduction

The need of mining structural data to uncover objects or concepts that relates objects (i.e., subgraphs that represent associations of features) has increased in the last two decades, thus creating the area of *graph-based data mining* (GBDM) [1], [2]. A significant number of applications require a graph-based representation of structural information such as analysis of microarray data in bioinformatics, pattern discovery in a large graph representing a social network, analysis of transportation networks, and community discovery in Web data, among many others [3].

Mining graph-based data involves an effective and efficient manipulation of relational graphs, towards discovering important patterns. Several approaches for knowledge discovery in graph data have been proposed in the literature including Subdue, the apriori family of methods, frequent subgraph discovery, Gaston, gSpan, MoFa/MoSS, JoinPath, CloseGraph, FFSM, Spin, CLOSECUT and SPLAT, gPrune, etc [3]. Roughly, all the existing methods work by performing a search in the lattice of all possible subgraphs. The underlying search process is usually guided by a single objective which represents a unique and specific user preference. For example, the extraction of those subgraphs satisfying some minimum frequency threshold or of those subgraphs whose size is less than/equal to some given maximum integer value are typical choices.

On the other hand, Pareto-based multiobjective evolutionary search strategies [4], [5] have recently gained much importance in data mining and machine learning communities. This is due to one common benefit observed in the different multiobjective learning approaches: a deeper insight into the learning problem can be gained by analyzing a Pareto set composed of multiple nondominated solutions with a different trade-off in the satisfaction of the conflicting learning problem objectives. Besides, some empirical studies in [6], [7] have shown that Pareto-based learning approaches may help the search process underlying the learning algorithm to escape from local optima, thus improving the accuracy of the learning model. For a recent review on Pareto-based multiobjective machine learning the interested reader is referred to Jin and Sendhoff [8]. The edited book by Jin [9] also constitutes a very good snapshot of the area.

Prakash Shelokar, Arnaud Quirin, and Óscar Cordón are with the European Centre for Soft Computing, Mieres-33600, (Asturias), Spain. Email: prakash.shelokar, arnaud.quirin, oscar.cordon@softcomputing.es. This work has been partially supported by the Spanish Ministry of Science and Innovation (MICINN) under project TIN2009-07727, including EDRF fundings.

In this paper we propose the incorporation of *Pareto-based multiobjective search strategies* [4], [5] to graph mining techniques in order to allow them to handle the simultaneous optimization of several conflicting goals representing different user preferences. To do so, we have selected Subdue [10], [11], the first and the most classical graph-based knowledge discovery system, and have extended it to deal with the latter scenario. This issue has been tackled by incorporating a Pareto-based multiobjective component proposed in the nondominated sorting genetic algorithm (NSGA-II) [12] to the selection strategy of Subdue in order to decide the search direction to be explored. This method has been called MOSubdue (Multiobjective Subdue).

The subgraph discovery process of MOSubdue evaluates every subgraph by jointly considering two preferences, viz. support and complexity, in order to explore different search directions within the multiobjective problem landscape. It is worth to note that MOSubdue is able to retrieve a Pareto set of nondominated, meaningful subgraphs from a structural database in a single run of the algorithm, showing different optimal trade-offs between support and complexity. In the current contribution we empirically demonstrate the good performance of the proposed approach by using a real-life data set concerning the analysis of web sites [13].

The paper is organized as follows. Section II presents some preliminaries, including the operation mode of Subdue algorithm for subgraph discovery, a brief literature survey, and some basic definitions on multiobjective optimization related to our work. Section III describes the MOSubdue methodology. Section IV provides the results obtained by our algorithm when applied to a real-world web sites database. Finally, Section V concludes with some discussion.

II. Preliminaries

This section provides the methodological and problem background used in this work. First, we review the operation mode of Subdue, and then describe some basics of multiobjective optimization (MOO) related to our work.

A. The Subdue Algorithm

Subdue [10], [11] is the first algorithm proposed in GBDM for discovering interesting and repetitive subgraphs in a structural database. The algorithm uses the minimum description length (MDL) principle [14] to discover frequent subgraphs, extract them, and replace them by a single node in order to compress the entire database. The maximization of the description length is considered and suggests how well a subgraph can compress the input graph. This measure is a combination of two objectives, support and complexity (or size), which are commonly used preferences in GBDM algorithms. The description length of a subgraph g in graph data G is computed as:

$$value_{\text{MDL}}(g, G) = I(G)/(I(g) + I(G|g)) \tag{1}$$

The description length of a graph is the necessary number of bits to completely describe the graph; $I(G)$ and $I(g)$ are the number of bits required to encode the graph G and the subgraph g respectively; $I(G|g)$ is the number of bits required to encode the graph obtained by compressing G with g, i.e. substituting each occurrence of g in G by a single node [15]. The extracted subgraphs represent structural concepts in the data. Subdue can be run several times in a sequence in order to extract different meta-concepts from the previously simplified database. After multiple Subdue runs on the database, we can discover a hierarchical description of the structural regularities in the data [16]. Subdue can also use background knowledge, such as domain-oriented expert knowledge, to be guided and to discover subgraphs for a particular domain goal. The algorithm performs the subgraph discovery in polynomial time. In the last fifteen years, it has been successfully applied for many real-world problems including, bioinformatics [17], counter-terrorism [2], web data mining [18], geology [19], and aviation and chemistry [20], among others.

```
1)  SUBDUE (Graph database $G$, BeamWidth, MaxBest, Limit)
2)  ParentList = {Vertex $v$ — $v$ has a unique label in graph}
3)  Evaluate all the unique vertices
4)  BestList = UpdateBestList(ParentList) //Restrict to size MaxBest
5)  ProcessedParents = 0
6)  while ProcessedParents $\leq$ Limit and ParentList $\neq \emptyset$ do
7)      ChildList = {}
8)      while ParentList $\neq \emptyset$ do
9)          Parent = RemoveHead(ParentList)
10)         CandidateList = ExtendSubgraph(Parent)
11)         Evaluate subgraphs in CandidateList
12)         ChildList = UpdateChildList(CandidateList) // Restrict to size BeamWidth
13)         ProcessedParents = ProcessedParents+1
14)     end while
15)     BestList = UpdateBestList(ChildList) //Restrict to size MaxBest
16)     ParentList = ChildList
17) end while
18) Return BestList // Discovered subgraphs
```

Fig. 1. The outline of Subdue algorithm.

Like many GBDM algorithms, Subdue models the search process in the lattice of all possible subgraphs. In order to conceptually arrange all the subgraphs in this lattice, it begins with the set of subgraphs (ParentList) containing all uniquely labeled vertices (that is, each subgraph represents one uniquely labeled vertex). To traverse the lattice, Subdue extends each subgraph from ParentList in all possible ways either by a single edge and a vertex or by a single edge only if both vertices are already in the subgraph. These subgraphs are kept on a queue (ChildList) and are ordered based on their ability to compress the graph, measured by the MDL index. This queue can grow exponentially as the search progresses. In order to avoid that exponential explosion, Subdue uses a variant of beam search [21] based on selecting and only storing the top *BeamWidth* subgraphs in ChildList. The search terminates either upon reaching a user specified limit on the number of subgraphs extended, or upon exhaustion of the search space. Once the search process terminates, the algorithm returns a list of the best subgraphs, BestList, which can store a maximum of *MaxBest* subgraphs. Fig. 1 shows the outline of the Subdue system. Subdue takes as input a graphical database G, and then three parameters to control the search process.

B. Multiobjective Optimization Basics

As follows, we describe some MOO basics and general notations used throughout the paper. A general MOO problem can be described as a vector function f that maps a tuple of l parameters (decision variables) to a tuple of o objectives [4], [5]. Formally:

$$\begin{aligned} \text{min./max. } & y = f(x) = (f_1(x), f_2(x), \ldots, f_o(x)) \\ \text{subject to } & x = (x_1, x_2, \ldots, x_l) \in X \\ & y = (y_1, y_2, \ldots, y_o) \in Y \end{aligned} \tag{2}$$

where x is called the decision vector, X is the parameter space, y is the objective vector, and Y is the objective space. To compare any two solutions, we apply the well known concept of Pareto dominance: assume, without loss of generality, a maximization problem, and consider two solutions x^1 and x^2 with vector-valued objective functions y^1 and y^2 respectively. An objective vector y^1 is said to weakly dominate another objective vector y^2 ($y^1 \succ y^2$) if no component of y^1 is smaller than the corresponding component of y^2 and at least one component is greater. Accordingly, we can say that a solution x^1 is better to another solution x^2, i.e., x^1 dominates x^2 ($x^1 \succ x^2$), if $f(x^1)$ dominates $f(x^2)$. Mathematically, the concept of Pareto optimality is defined as follows:

$$\begin{aligned} & \forall i \in \{1, 2, \ldots, o\} : f_i(x^1) \geq f_i(x^2) \wedge \\ & \exists j \in \{1, 2, \ldots, o\} : f_j(x^1) > f_j(x^2) \end{aligned} \tag{3}$$

Hence, optimal solutions, i.e., solutions not dominated by any other solution, may be mapped to different objective vectors. In other words, there may exist several optimal objective vectors representing different trade-offs between the objectives.

In a typical MOO problem there is not usually a single optimal solution to solve the problem, (i.e., being better than the remainder with respect to every objective, as in single objective optimization) but a set of optimal solutions that are superior to the remainder (are not dominated by them) when all the objectives are jointly considered. These solution vectors are known as nondominated, efficient or Pareto optimal and constitute the so-called nondominated, efficient or Pareto optimal solutions set. Their set of objective vectors is called nondominated, efficient or Pareto front.

III. Multiobjective Subdue Structure and Operation Mode

This section justifies the need of handling several preferences simultaneously in any GBDM algorithm and later describes the implementation of Subdue for multiobjective subgraph discovery.

A. Justification for MOO in GBDM

The fact that the existing GBDM techniques apply a single objective search has many limitations such as we risk a huge (very few) subgraphs in the case of weak (strict) preference value. For example, the formulation of the search problem considering a single criterion like the usual subgraph occurrence frequency (i.e., the support) would normally result in the generation of small subgraphs with a large extent. These subgraphs may not be very informative and as a consequence of being very basic a user may not be able to uncover any new knowledge from them. Moreover, in real-life applications a user is generally interested in mining a graph-based repository with several preferences which are actually meaningful to her/him. These preferences are often conflicting in nature. For example, users are normally interested in the discovery of subgraphs with high support and complexity. These objectives are conflicting, as simpler descriptions are usually the most frequent ones and *vice versa*. One simple approach to apply an existing GBDM algorithm to the latter scenario is to combine several criteria into a single-objective function by any kind of objective aggregation scheme. However, the aggregation of two conflicting criteria (*e.g.*, subgraph support and size) in a single-objective scalar function would result in a similar behavior where only the specific subgraphs showing the specific trade-off between the two objectives, explicitly or implicitly specified in the aggregation function, would be retrieved. In view of the reasons stated above, any successful GBDM methodology should not only rely on the optimization of a single criterion but also consider simultaneously additional, conflicting criteria to extract better defined concepts based on the size of the subgraph being explained, the number of retrieved subgraphs, and their diversity. With that aim in mind, in this paper we propose the incorporating of Pareto-based search strategies to graph mining techniques in order to allow them to handle the simultaneous optimization of several conflicting goals representing different user preferences.

To do so, we have selected Subdue [10], [11], the first and the most classical graph-based knowledge discovery system, and have extended it to deal with the latter scenario by incorporating a Pareto-based evolutionary multiobjective component [22], [23] to the selection strategy of Subdue in order to decide the search direction to be explored, i.e., to the decision on which of the subgraphs in ChildList will compose ParentList in the next iteration (see Section II). In the next subsection we provide the novel MultiObjective Subdue (MOSubdue) proposal.

B. Proposal

Towards applying preferences in subgraph discovery using Subdue, every subgraph can be jointly evaluated considering two objective functions: (i) the support (the occurrence frequency of the subgraph g in the whole graph data G), and (ii) the complexity or size (the number of vertices and edges present in the subgraph g). These objectives can be calculated as:

$$\mathit{value}_{\text{support}}(g, G) = \#\text{subgraphs in } G \text{ matching } g \tag{4}$$

$$\mathit{value}_{\text{complexity}}(g, G) = \#\text{vertices}(g) + \#\text{edges}(g) \tag{5}$$

The higher the support and size of a subgraph, the larger its importance since more frequent subgraphs represent sounder insights while larger subgraphs are associated to more concise (and thus more difficult to uncover) descriptions. Therefore, the best possible subgraphs are the ones that are maximized both in support and size. Nevertheless, the problem is that both objectives are conflicting, as simpler descriptions are usually the most frequent ones and *vice versa*. Thanks to the proposed extension, we will be able to retrieve a Pareto set of nondominated, meaningful subgraphs from a structural database in a single run of the algorithm, showing different optimal trade-offs between support and complexity. Hence, MOSubdue will allow us to uncover cohesive subgraphs comprising even a moderate number of observations (and not only the most frequent ones, as usual) which describe the underlying phenomena from different angles, revealing novel information that otherwise would be concealed by uninformative frequent descriptions.

Fig. 2 shows the outline of the MOSubdue algorithm. MOSubdue considers a Pareto-based multiobjective selection strategy on the opposite to the usual Subdue's single objective operation mode for subgraph discovery. Basically MOSubdue replicates all the processes of original Subdue such as initial parent generation, parent expansion, and child generation. Besides, MOSubdue applies Subdue's beam search in order to constrain the search space of subgraphs. As in classical Subdue, the beam search limits the size of ChildList using the parameter *BeamWidth*. However, in our case the selection of subgraphs in ChildList is performed by implementing the well known concept of Pareto dominance in order to guide the search towards discovery of the

```
1)  MOSUBDUE (Graph, BeamWidth, ParetoArchiveSize, Limit)
2)  ParentList = {Vertex $v$ — $v$ has a unique label in graph}
3)  Estimate support and complexity of all the unique vertices
4)  ParetoArchiveList = UpdateParetoArchive(ParentList) //stores only nondominated subgraphs
5)  ProcessedParents = 0
6)  while ProcessedParents $\leq$ Limit and ParentList $\neq \emptyset$ do
7)      ChildList = {}
8)      while ParentList $\neq \emptyset$ do
9)          Parent = RemoveHead(ParentList)
10)         CandidateList = ExtendSubgraph(Parent)
11)         Estimate support and complexity of subgraphs in CandidateList
12)         ChildList = UpdateChildList(CandidateList)
13)         ProcessedParents = ProcessedParents+1
14)     end while
15)     Rank subgraphs in ChildList using nondominated sorting
16)     ParetoArchiveList = UpdateParetoArchive(ChildList)
17)     Select BeamWidth subgraphs on queue ChildList using dominance rank
        and density estimation
18)     ParentList = ChildList
19) end while
20) Return ParetoArchiveList // Discovered nondominated subgraphs
```

Fig. 2. The outline of MOSubdue algorithm.

nondominated subgraphs, the Pareto set. After terminating the search, the algorithm reports a set of nondominated subgraphs stored in an external list, ParetoArchiveList, whose size is controlled by the parameter ParetoArchiveSize. ParetoArchiveList is updated at the end of each iteration using the nondominance criteria (eq. (3)).

The idea of calculating an individual's fitness in the population on the basis of Pareto dominance to achieve an efficient set of solutions has been very successful in the area of evolutionary multiobjective optimization (EMO) [22], [23]. The nondominated sorting genetic algorithm, NSGA-II [12], is one of the most popular EMO algorithm. NSGA-II makes use of Pareto dominance approach, where the population is divided into several fronts and the depth reflects to which front an individual belongs to. An individual is assigned a pseudo-dominance rank equal to the front number. The selection of an individual in the population is performed using this dominance rank value. In this work, our MOSubdue implementation utilizes this dominance depth approach to perform selection of subgraphs in ChildList.

MOSubdue implements Pareto dominance based fitness procedure in order to assign a scalar fitness value to all the subgraphs in ChildList. This is done by identifying different nondominated fronts in ChildList. Initially, for each subgraph we calculate two elements: a) domination count n_g, the number of subgraphs which dominate the subgraph g, and b) Z_g a set of subgraphs that the subgraph dominates. All the subgraphs with domination count zero compose the first nondominated front F_1. Then, for each subgraph with $n_g = 0$, we visit each member (q) of its set Z_g and reduce its domination count by one. In doing so, if for any member q the domination count becomes zero, we put it in a separate list Q. These members belong to the second nondominated front F_2. The above procedure is repeated with each member

```
1)  Begin : ChildList subgraphs with support and complexity values.
2)  for each subgraph $g \in$ ChildList do
3)      $Z_g = \emptyset$ and $n_g = 0$
4)      for each subgraph $g' \in$ ChildList, $g \neq g'$ do
5)          if $g \succ g'$ then // $g$ dominates $g'$
6)              $Z_g = Z_g \cup \{g'\}$ // Add $g'$ to the dominated set of $g$
7)          else if $g' \succ g$ then // $g'$ dominates $g$
8)              $n_g = n_g + 1$ // Increment domination counter of $g$
9)          end if
10)     end for
11)     if $n_g = 0$ then // $g$ belongs to the first front
12)         $g_{rank} = 1$ // Assign dominance rank to $g$
13)         $F_1 \cup \{g\}$ // Add $g$ to the first front
14)     end if
15) end for
16) $r = 1$
17) while $F_r \neq \emptyset$ do
18)     $Q = \emptyset$
19)     for each $g \in F_r$ do
20)         for each $g' \in Z_g$ do
21)             $n_{g'} = n_{g'} - 1$
22)             if $n_{g'} = 0$ then
23)                 $g'_{rank} = r + 1$
24)                 $Q = Q \cup \{g'\}$
25)             end if
26)         end for
27)     end for
28)     $r = r + 1$
29)     $F_r = Q$
30) end while
31) output : subgraphs in ChildList sorted into different nondominated fronts
```

Fig. 3. Nondominated sorting in MOSubdue algorithm.

of Q and the third front is identified. This process continues until all the fronts are identified. The steps involved in identifying nondominated fronts in ChildList are shown in Fig. 3. Once the fronts are identified, each subgraph is assigned a scalar fitness (or rank) equal to its nondomination level and ChildList is sorted based on nondomination level in the ascending order of magnitude. The algorithm considers subgraphs with rank 1 are the best; subgraphs with rank 2 are the second-best; and so on.

Most state-of-the-art algorithms in EMO take into account density information in addition to the dominance criterion to obtain a well distributed set of Pareto optimal solutions. To guide the search towards a good spread of solutions in the Pareto set approximation, MOSubdue incorporates density information into the selection process of population individuals: an individual's chance of being selected is decreased according to the density of individuals in its neighborhood. In this study we use density information in addition to the dominance criterion to select subgraphs in ChildList in order to compose ParentList for the next extention step. The density information is calculated among the subgraphs belong to any particular nondominated front, i.e. having identical dominance rank. This additional density information represents the diversity of subgraphs belonging to that nondominated front and it is used to select the most diversified subgraphs. For this purpose, we employ the density estimation method proposed in the NSGA-II algorithm [12]. This method does

```
1)  Begin : Nondominated set $F$
2)  $n = |F|$ // Number of subgraphs in $F$
3)  for each subgraph $g$ do
4)      $F[g]_{\text{distance}} = 0$ // Initialize distance
5)  end for
6)  for objective $k \in \{\text{Support, Complexity}\}$ do
7)      $F = \text{sort}(F, k)$
8)      $F[1]_{\text{distance}} = F[n]_{\text{distance}} = \infty$ // Assign maximum distance value
9)      for $g = 2$ to $(n-1)$ do
10)         $F[g]_{\text{distance}} = F[g]_{\text{distance}} + (F[g+1].k - F[g-1].k)/(f_k^{\max} - f_k^{\min})$
11)     end for
12) end for
```

Fig. 4. Density estimation.

not require *any* user-defined parameter for calculating the diversity of subgraphs. The density of subgraphs surrounding a particular subgraph in ChildList is calculated as the average distance of the subgraphs on either side of this subgraph along both the preferences, i.e. support and complexity. This distance computation is done by sorting ChildList according to the first objective function (i.e. support) value in ascending order of magnitude. The subgraphs with smallest and largest objective function values are assigned an infinite distance value. All other intermediate subgraphs are assigned a distance value equal to the absolute normalized difference in the objective function values of two adjacent subgraphs. This calculation is then continued with the other objective function (i.e. complexity). The overall distance value is calculated as the sum of individual distance values corresponding to our two objectives. The objective functions are normalized before doing distance computation.

Fig. 4 gives the outline of density estimation in the set F. $F[g].k$, $k \in \{\text{support, complexity}\}$ refers to the k^{th} preference of subgraph g in set F and the parameters $f_k^{\max}$ and $f_k^{\min}$ the maximum and minimum values of k^{th} preference. After all the subgraphs in set F are assigned a distance metric, we can compare two subgraphs for their extent of proximity with other subgraphs. A subgraph with a smaller value of this distance measure is considered to have a greater density of subgraphs in its neighborhood and *vice versa*. To select subgraphs with good diversity in the set F, we prefer subgraphs with higher distance metric value. The ties between subgraphs with equal distance values are resolved arbitrarily.

IV. Experiments

In this section we analyze the behavior of MOSubdue algorithm by means of various unary and binary metrics proposed in the EMO literature [22], [23], [24], [25], and visual representations of the obtained Pareto fronts. Besides, we apply Subdue using different objective functions to produce an aggregated Pareto front which is then used for comparing the performance of MOSubdue. Firstly, some introductory subsections are included to define the experimental design by setting up parameters, describing the used database, and reporting the considered EMO metrics. Later, the whole experimental analysis is given.

A. Web sites analysis dataset

This database is available online at Subdue website[1]. The data were extracted from real World Wide Web pages and were transformed to labeled graphs using a web robot [13]. In this work, we have considered ProfStu graph data which was generated using professor and student web sites. The information these graphs contain is hyperlink structure and page's content. The data consist of 47 graphs, which are quite complex containing several unique vertex labels. We consider this real world database as a challenging way to illustrate our multiobjective approach. Instead of dealing with each of the huge graphs individually as done in [13], we selected five graphs (numbered 6, 19, 25, 43, and 45) from the ProfStu database and joined them into a single database with a complexity of 832 vertices, 885 edges, and 511 unique labels. For this database, the true Pareto set is not known due to its large complexity and real-world nature. For comparison purposes, we have used a pseudo-optimal Pareto set, which is obtained as a fusion of all the nondominated sets achieved by both the algorithms in different runs. This pseudo-optimal Pareto set contains 8 nondominated subgraphs that are distinct in the objective vector space.

B. Experimental setup

Here, we first provide the use of the original Subdue algorithm for generating a set of nondominated subgraphs in order to compare the performance with our MOSubdue proposal. Next the parameter values, and the considered performance metrics are presented.

1) Nondominated subgraphs using basic Subdue: The current version of Subdue[2] supports three subgraph evaluation metrics, viz. MDL, support and size. We run Subdue with each evaluation metric on the database that outputs BestList (last line, Fig. 1), whose size is set to *MaxBest* = 33. Later, we combine the three BestLists and remove any multiple copied subgraphs in order to produce the aggregated set. Finally, we apply the non dominance definition (see Eq. (3)) on the aggregated set to remove the dominated subgraphs and obtain a baseline nondominated solution set approximation generated from the single-objective Subdue algorithm.

2) Parameter values: MOSubdue algorithm has been run ten times with ten different seeds during 1000 seconds. Besides, original Subdue was run during 1000 seconds in which simulation of Subdue with each objective was performed for (333 seconds) 1/3 of the total running time. The maximum number of nondominated subgraphs that can be reported by MOSubdue was set to *ParetoArchiveSize* = 100. We used three different values of *BeamWidth*, equal to 5, 10, and 20 to analyze the behavior of our proposed MOSubdue approach.

[1]http://ailab.wsu.edu/subdue/datasets/webdata.tar.gz
[2]http://ailab.wsu.edu/subdue/software/subdue-5.2.1.zip

3) Metrics of performance: In this paper, we have considered the two usual kinds of multiobjective metrics [22], [23], [24], [25]:

- those which measure the quality of a nondominated solution set returned by an algorithm, and
- those which compare the performance of two different algorithms.

As regards the former group, we use the hypervolume ratio (HVR) and cardinality measure to compare the obtained Pareto set approximations. The hypervolume measures the volume enclosed by a Pareto set approximation with respect to a reference point. For two-dimensional objective vector, hyervolume is the summation of area covered by each member of the Pareto set. In the case of maximization problem, as ours, we define a reference point as $(0, 0)$. Here, we use the hypervolume ratio (HVR) [22]. HVR measures both diversity and closeness of the Pareto set and is calculated as:

$$\mathit{HVR} = \frac{H_1}{H_2} \tag{6}$$

where H_1 and H_2 are the volume of the Pareto set and the true Pareto set (or the pseudo-optimal Pareto set in case the latter is not known) respectively. In our case, we will use a pseudo-optimal Pareto set. A value of HVR equals to one represents that the Pareto front and the pseudo Pareto front are equal.

The cardinality is equal to the size of the obtained Pareto set approximation by an algorithm.

The unary metrics allow us to determine the absolute, individual quality of the obtained Pareto set approximation, but they cannot be used for comparison purposes. On the other hand, binary indicators have been proposed for comparing the Pareto set approximations obtained by different multiobjective algorithms. In this work, we have used the following two binary indicators C and I_ϵ to compare the Pareto set approximations two by two:

The coverage metric (*C-measure*) compares a pair of nondominated sets by computing the fraction of each set that is covered by the other [26]:

$$C(X', X'') = \frac{|\{\forall g'' \in X''; \exists g' \in X' : g' \succeq g''\}|}{|X''|} \tag{7}$$

where $g' \succeq g''$ indicates that the subgraph g' dominates or cover the subgraph g'' in a maximization problem. A value of $C(X', X'') = 1$ means that all the subgraphs in X'' are dominated or covered by the subgraphs in X'.

The I_ϵ-measure gives the minimum distance by which one set can be translated in each dimension in objective space such that the other set is weakly dominated [25]. In this paper, we consider the additive epsilon indicator, I_ϵ, given as:

$$\begin{aligned} I_\epsilon(X', X'') = \min_{\epsilon}\{ & f_i(g') + \epsilon \geq f_i(g''); \\ & g' \in X', g'' \in X'', \text{for } i = 1, \ldots, o \ \} \end{aligned} \tag{8}$$

where I_ϵ value is the minimal amount ϵ by which one needs to improve each objective, i.e., replace $f_i(g')$ by $f_i(g') + \epsilon$, such that it just dominates $f(g'')$, i.e., $f_i(g') + \epsilon \geq f_i(g'')$ for all $i = 1, \ldots, o$. A value of $I_\epsilon(g', g'') < I_\epsilon(g'', g')$ means that $f(g')$ dominates $f(g'')$ and $I_\epsilon(X', X'')$ indicates the minimal amount ϵ by which at least one subgraph in the Pareto set approximation X' dominates at least one subgraph in the Pareto set approximation X''.

For a Pareto set approximation, the HVR measure is better when it tends to one, and the cardinality measure which is actually the size of the Pareto set, is better when it takes higher value. In the pair-wise comparison of Pareto set approximations the C measure is better when it tends to one, while the I_ϵ measure is better when it takes lower value.

TABLE I
COMPARISON USING HVR MEASURE.

Method	*BeamWidth* = 5	*BeamWidth* = 10	*BeamWidth* = 20
Subdue	0.5918 (-)	0.5020 (-)	0.4000 (-)
MOSubdue	0.9853 (0.0870)	0.6918 (0.0397)	0.4494 (0.0348)

TABLE II
COMPARISON USING CARDINALITY MEASURE.

Method	*BeamWidth* = 5	*BeamWidth* = 10	*BeamWidth* = 20
Subdue	5 (-)	13 (-)	11 (-)
MOSubdue	8.8 (1.6866)	16 (0.0)	22.8 (3.7059)

TABLE III
C-MEASURE VALUES.

	BeamWidth = 5		*BeamWidth* = 10		*BeamWidth* = 20	
Method	Subdue	MOSubdue	Subdue	MOSubdue	Subdue	MOSubdue
Subdue	-	0.375 (0)	-	0.4286 (0.0)	-	0.19 (0.0738)
MOSubdue	1.0000 (0)	-	1.0000 (0)	-	0.7500 (0.0)	-

TABLE IV
I_ϵ-MEASURE VALUES.

	BeamWidth = 5		*BeamWidth* = 10		*BeamWidth* = 20	
Method	Subdue	MOSubdue	Subdue	MOSubdue	Subdue	MOSubdue
Subdue	-	2.2109 (0.1306)	-	1.5714 (0.0)	-	1.5714 (0.0)
MOSubdue	1.0000 (0)	-	1.0000 (0)	-	1.0803 (0.0498)	-

C. Analysis of results

The results using the two unary metrics, HVR and cardinality are given in Tables I and II respectively. The values shown are the average and standard deviation of the ten performed runs for the MOSubdue approach. The two binary metrics C and I_ϵ calculate the dominance degree for a pair of the Pareto set approximations of two algorithms which is

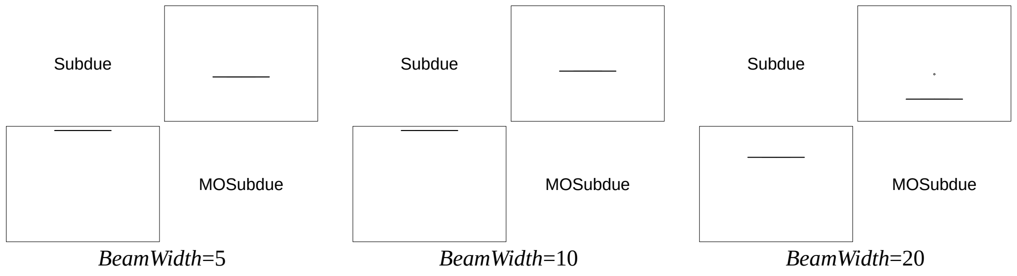

Fig. 5. Comparison using C-Measure (box-plots).

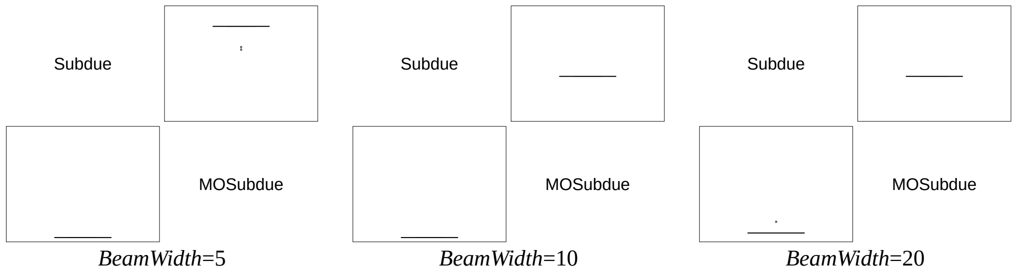

Fig. 6. Comparison using I_ϵ-Measure (box-plots).

plotted using the box-plots (see Figs. 5 and 6). Each box refers to algorithm A in the corresponding row and algorithm B in the corresponding column and gives the fraction of B covered by A, $C(A, B)$, in Fig. 5, and the value of the dominance preserving indicator $I_\epsilon(A, B)$ in Fig. 6. Besides the box-plot figures, all the values of the C and I_ϵ metrics are also provided in Tables III and IV respectively.

The results show that the best performance is corresponding to the parameter *BeamWidth* equal to 5, while the performance worsens for increasing value of the *BeamWidth* (= 10 and 20), regardless the algorithm. This is probably due to the fact that there is a fixed run time which makes the larger width search with *BeamWidth* = 10 and 20 less productive as there is not enough depth search to get high quality solutions.

Our analysis based on the various performance evaluation metrics clearly demonstrates that the MOSubdue approach is able to achieve better nondominated solution set than obtained by Subdue for the web domain database. We will analyze the results obtained from both unary and binary performance evaluation metrics as follows:

The values of HVR reported in Table I show that the Pareto set approximations obtained by the MOSubdue algorithm cover a wide range of values in the objective space and clearly outperform those of the original Subdue. MOSubdue has obtained the best Pareto front approximation with an average HVR value of 0.9853, which is higher than the HVR value of 0.5918 for that produced by Subdue for *BeamWidth* = 5. The cardinality measure is based on the size of the Pareto set approximation obtained by an algorithm. From Table II the MOSubdue approach has produced the Pareto set approximations with higher cardinality value than those

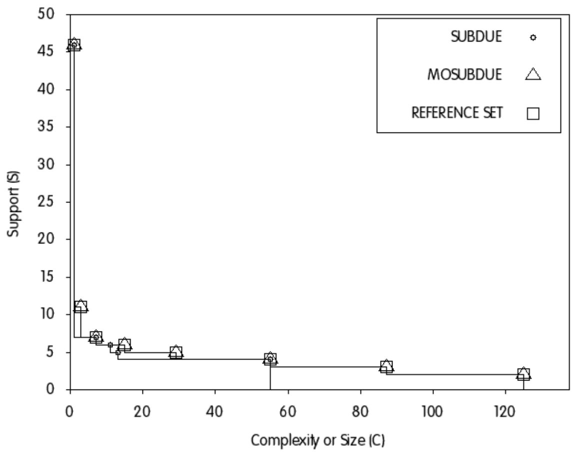

Fig. 7. Representation of the nondominated sets.

by Subdue, regardless the *BeamWidth* value.

The analysis of the binary indicators, C and I_ϵ in Tables III and IV, and Figs. 5 and 6 also highlights the better performance of the MOSubdue method. The values of the C and I_ϵ metrics show that the MOSubdue algorithm outperformed Subdue. For *BeamWidth* = 5, the Pareto set approximations obtained in ten different runs by the proposed approach cover all the members of the Pareto set approximation produced by the Subdue method.

Finally, we also compare the performance of our algorithm by means of the graphical representation of the Pareto front approximations for *BeamWidth* = 5 and the pseudo Pareto front of the database, as shown in Fig. 7. The approximation of the Pareto front obtained by MOSubdue is a fusion of the ten different runs. Let us call P_j the nondominated front obtained in the j^{th} run by MOSubdue, the aggregated Pareto front $P = P_1 \cup P_j \cup \ldots \cup P_{10}$ is the union of the ten different nondominated sets. Fig. 7 shows that the pseudo Pareto set dominates two out of four solution in the Pareto set approximation of Subdue, while the fusioned nondominated set of MOSubdue is identical to the pseudo Pareto set.

V. Concluding remarks

We have proposed a Pareto-based multiobjective search strategy for subgraph mining in structural databases. The approach is an extension of the Subdue algorithm called MOSubdue. The performance of MOSudue has been analyzed using a real-world web analysis domain taking the basic Subdue implementation as the baseline.

In view of the obtained results, we can conclude that MOSubdue is able to discover a set of good quality nondominated subgraphs in a single run. The analysis of MOSubdue was based on ten independent runs. Each of the ten obtained Pareto set approximations showed good diversity and closeness to the pseudo-optimal Pareto set according to four performance metrics commonly used in EMO. It is evident from the results that our proposal clearly outperformed the basic Subdue implementation when generating nondominated subgraphs for the problem.

Several ideas for future developments arise from this work. On the one hand, we should notice that, at the start of the search process, the MOSubdue algorithm builds a large number of subgraphs belonging to the first nondominated front whose support decreases as the search progresses. This is due to the start of the search process (line 2, Fig. 2) with the largest support subgraphs and the application of a constructive search. As MOSubdue implements Subdue's beam search with a constant *BeamWidth* parameter, the algorithm may produce suboptimal approximation sets by missing out some subgraphs that might allow it to explore other regions of the search space. One approach to solve the latter drawback may be to use an adaptive *BeamWidth* which will take a high value initially and will be decreased using some adaptation scheme as the search progresses. This will allow us to explore several subgraphs at the beginning of the search process.

On the other hand, as already seen, a Pareto-based search strategy and a diversified subgraph selection can actually benefit the algorithm in achieving good nondominated set approximations. However, MOSubdue keeps only applying a constructive search (line 10, Fig. 2) to expand the subgraphs, as done by Subdue. That fact causes a bias in the search process towards one objective (support) during the exploration. This indicates an implementation of a pure EMO algorithm in GBDM could be more useful since the evolutionary algorithm would perform the exploration of different tree levels in the subgraph lattice at a given generation, thus not biasing the exploration towards any particular objective.

References

[1] T. Washio and H. Motoda, "State of the art of graph-based data mining," *ACM SIGKDD Explor Newsl*, vol. 5, no. 1, pp. 59–68, 2003.

[2] L. B. Holder and D. J. Cook, "Graph-based data mining," in *Encyclopedia of Data Warehousing and Mining, Vol. II*, J. Wang, Ed. Hershey: Information Science Reference, 2005, pp. 943–949.

[3] D. Cook and L. Holder, Eds., *Mining Graph Data*. London: Wiley, 2007.

[4] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making Theory and Methodology*. Amsterdam: North-Holland, 1983.

[5] T. Gal, T. Stewart, and T. Hanne, Eds., *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory and Applications*, ser. International Series in Operations Research and Management Science. Springer Verlag, 1999, vol. 21.

[6] H. Abbass, "Speeding up back-propagation using multi-objective evolutionary algorithms," *Neural Comput*, vol. 15, pp. 2705–2726, 2003.

[7] R. de A. Teixeira, A. Braga, R. Takahashi, and R. Saldanha, "Improving generalization of MLP with multi-objective optimization," *Neurocomputing*, vol. 35, pp. 189–194, 2000.

[8] Y. Jin and B. Sendhoff, "Pareto-based multi-objective machine learning: An overview and case studies," *IEEE T Syst Man Cy C*, vol. 38, pp. 397–415, 2008.

[9] Y. Jin, Ed., *Multi-Objective Machine Learning*. New York: Springer-Verlag, 2006.

[10] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," *J Artif Intell Res*, vol. 1, pp. 231–255, 1994.

[11] D. Cook and L. Holder, "Graph-based data mining," *IEEE Intell Syst*, vol. 15, pp. 32–41, 2000.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans Evol Comput*, vol. 6, pp. 182–197, 2002.

[13] J. Gonzalez, "Empirical and theoretical analysis of relational concept learning using a graph based representation," Ph.D. dissertation, Department of Computer Science & Engineering, University of Texas at Arlington, USA, 2001.

[14] J. Rissanen, *Stochastic Complexity in Statistical Inquiry Theory*. River Edge: World Scientific Publishing Co., Inc., 1989.

[15] L. Holder, D. Cook, J. Coble, and M. Mukherjee, "Graph-based relational learning with application to security," *Fundam Inf*, vol. 6, pp. 83–101, 2004.

[16] I. Jonyer, D. J. Cook, and L. B. Holder, "Graph-based hierarchical conceptual clustering," *J Mach Learn Res*, vol. 2, pp. 19–43, 2001.

[17] J. Kukluk, L. Holder, and D. Cook, "Learning node replacement graph grammars in metabolic pathways," in *Proc Int Conf Bioinformat & Comput Biol (BIOCOMP-07)*, 2007, pp. 44–50.

[18] A. Rakhshan, L. Holder, and D. Cook, "Structural web search engine," *Int J Art Intell Tools*, vol. 13, pp. 27–44, 2004.

[19] J. Gonzalez, L. Holder, and D. Cook, "Structural knowledge discovery used to analyze earthquake activity," in *Proc 13th Ann Florida Art Intell Res Symp (FLAIRS)*, 2000, pp. 86–90.

[20] R. Chittimori, J. Gonzalez, and L. Holder, "Structural knowledge discovery in chemical and spatio-temporal databases," in *Proc 16th Nat Conf Art Intell & 11th Conf Innovat App Art Intell (AAAI/IAAI)*, 1999, p. 959.

[21] B. T. Lowerre, "The HARPY speech recognition system," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, 1976.

[22] C. A. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-objective Problems*. Berlin: Springer, 2007.

[23] K. Deb, *Multi-objective Optimization using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

[24] E. Zitzler, L. Thiele, and K. Deb, "Comparison of multiobjective evolutionary algorithms: Empirical results," *IEEE Trans Evol Comput*, vol. 8, pp. 173–195, 2000.

[25] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans Evol Comput*, vol. 7, pp. 117–132, 2003.

[26] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans Evol Comput*, vol. 3, pp. 257–271, 1999.