Adaptive and Flexible Blocking for Record Linkage Tasks

Luiz Osvaldo Evangelista[1], Eli Cortez[1], Altigran S. da Silva[1], Wagner Meira Jr.[2]

[1] Universidade Federal do Amazonas, Brazil
luizevangelista@gmail.com {eccv,alti}@dcc.ufam.edu.br
[2] Universidade Federal de Minas Gerais, Brazil
meira@dcc.ufmg.br

Abstract. In data integration tasks, records from a single dataset or from different sources must often be compared to identify records that represent the same real world entity. The cost of this search process for finding duplicate records grows quadratically as the number of records available in the data sources increases and, for this reason, direct approaches, such as comparing all record pairs, must be avoided. In this context, blocking methods are used to create groups of records that are likely to correspond to the same real world entity, so that the deduplication can be applied to these blocs only. In the recent literature, machine learning processes are used to find the best blocking function, based on a combination of low cost rules, which define how to perform the record blocking. In this paper we present a new blocking method based on machine learning. Different from other methods, our method is based on genetic programming, allowing for the use of more flexible rules and a larger number of such rules for defining blocking functions, leading to a more effective process for the identification of duplicate records. Experimental results with real and synthetic data show that our method achieves over 95% of correctness when generating block of potential duplicate.

Categories and Subject Descriptors: H. Information Systems [**H.m. Miscellaneous**]: Databases

Keywords: Record Linkage, Blocking, Genetic Algorithms

1. INTRODUCTION

One of the main problems in data integration is to identify in the files to be integrated, records that match the same entity in the real world. In this context, an entity may be a company, individual or any other real world concept with well-defined meaning [Winkler 2006]. Once analyzed and compared, the records identified can be organized so as to form pairs of records that are considered duplicates. This process of comparison and identification of replicated records is known as *Record Linkage* (RL).

Through the RL process, records obtained from different data sources can be linked or duplicates in a single source can be identified [Winkler 2006]. The goal is the same in both cases: by associating records from different files or by identifying duplicated records from a same file are identified helps to improve the quality of data and facilitates access to information. An example of the *Record Linkage* process is shown in Figure 1.

As in the example illustrated, it can be observed that in practice, associating records that represent the same entity is not a trivial task, since the records usually provided for this type of operation do not have unique identifiers and therefore attribute values are used to determine duplicity in the midst of such data. This is the case for the attributes *Authors* and *Title* shown in Figure 1.

As happens in Figure 1, we note that, in general, attribute values do not have any pattern of representation (e.g., abbreviations, punctuation, etc.), hindering the identification of pairs of records from the same entity by means of exact match between the *string* obtained from each record. For this reason, approximately matching techniques for the identification of pairs are deployed. For example,

Copyright©2010 Permission to copy without fee all or part of the material printed in JIDM is granted provided that the copies are not made or distributed for commercial advantage, and that notice is given that copying is by permission of the Sociedade Brasileira de Computação.

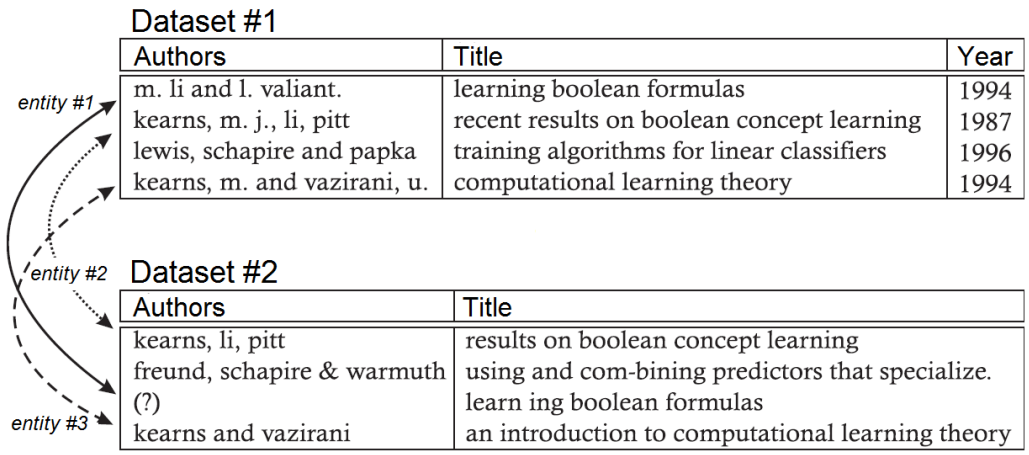

Fig. 1. Record Linkage using approximated matching.

techniques based on text similarity metrics, that consider similar terms instead of an exact match of terms.

Other problems may occur, even when these similarity metrics are used. For example, files that need to be processed often contain large amounts of records, and the number of pairs formed with these records can require long processing [Michelson and Knoblock 2006]. As a practical example, consider the case where two data files are composed of 10 thousand records each, and the number of pairs found with the records of these two files is 100 million. If each pair is evaluated at 0.01 seconds, it would require approximately 11 days to examine all pairs, which is considered to be a high processing time to deal with only 10 thousand records of each data set.

In [Winkler 1994] an approach is presented based on an approximate algorithm to reduce the number of candidate pairs. With this approach, the associations between records occur in a *one-to-one* manner, where each record in a first set is bound to a record with the highest degree of similarity found in a second set. The algorithm may not always produce good results, being more suitable in situations where each set of records has a small number of duplicates.

To produce good results, regardless of the number of duplicates in each dataset, *Blocking* methods can be used, as a rapid strategy to group records according to some criteria of low processing cost at first, and then, after that, records that are likely to correspond to the same entity are further processed by *Record Linkage* methods. As a consequence, it is expected that, instead of a very large amount of pairs of records, only the most likely candidates are considered, thus, the *Record Linkage* operations are performed in less time.

In general, the blocking process can be used as a preparatory stage for *Record Linkage* processes. In this sense, blocking methods can also be viewed as information pruning techniques, discarding pairs that are least likely to correspond to the same entity.

In this paper, we present a new blocking method, called *BGP* (Blocking based on Genetic Programming), based on the learning of blocking expressions in the disjunctive normal form. The problem we deal with is to find the best blocking schema, given a set of pairs of records for analysis in a process of machine learning and a set of rules for findings with those pairs of records. It is noteworthy that this is the same scenario addressed by other methods of machine learning.

The *BGP* method is based on the genetic programming technique (GP) [Koza 1998], which is able to verify and match a greater number of rules, possibly larger than the amounts tested in other proposals in the literature [Bilenko et al. 2006; Michelson and Knoblock 2006]. More complex rules can also be

used in this case, increasing the chances that better results will be achieved. Without the restriction on the number and complexity of rules, several possibilities can be tried, such as the use of rules based on parameters.

The use of parameters can be considered a new aspect to the composition of blocking strategies. While previous methods only combine rules and attributes to define the blocking schemas, the *BGP* also chooses the best values of parameters that can be used in each rule, making the processes of machine learning occur with greater flexibility. As a result of using this new aspect, the chances of producing blocking schemas better adapted to the data is larger, increasing the blocking quality results.

To evaluate the quality and scalability of our blocking method we conducted experiments using the collections *Cora*, *CiteSeer* and *Evolbase*. The collections *Cora* and *CiteSeer* consist of real data and the collection *Evolbase* consists of synthetic data. These collections were used as datasets of previous studies. Collection *Cora* was used in [Bilenko et al. 2006] to evaluate the method *DNF Blocking*, and collections *CiteSeer* and *Evolbase*, in [Sarawagi and Bhamidipaty 2002] and [de Carvalho et al. 2006], respectively. In addition, we present experiments comparing the proposed method, *BGP* with the *DNF Blocking* method [Bilenko et al. 2006] and the *BSL Blocking* [Michelson and Knoblock 2006].

Preliminary results with our work on *BGP* were previously presented in [Evangelista et al. 2009], and here we present a number of extensions to this work including a new set of experiments and analyses of the proposed blocking method and further discussions comparing it with recent blocking method proposals in the literature.

This paper is organized as follows. In Section 2 we review the existing blocking methods in the current literature and the differences between the approaches used in these methods. In Section 3, *BGP* is presented as an alternative blocking method, describing how it was developed using the capabilities of genetic programming. Section 4 shows the results of experiments performed with the proposed method, comparing it with state-of-art baseline methods. Finally, in Section 5 the conclusions of this work are presented, as well as suggestions for future work.

2. RELATED WORK

Several studies on methods of blocking can be found in current literature. The first proposals were based on the use of similarity metrics and therefore may fail in several situations, e.g., when files have large amounts of duplicates [Bhattacharya and Getoor 2004]. Generally, methods of blocking can be classified into two groups according to the approach used to organize records in blocks: (1) *static methods* and (2) *dynamic methods*.

In the static methods of blocking, the process of grouping the records does not take into account the characteristics found in the data, and is conducted in the same way in all situations. Examples of methods with this feature are *Canopy Blocking* [McCallum et al. 2000] and *Soft TF-IDF* [McCallum et al. 2000]

The main idea of the *Canopy Blocking* [McCallum et al. 2000] is to group records efficiently in a two-step process: the first step is performed with a low processing cost and the second is intending to refine the results of the first step. Notice, however, that this approach may not work in all cases. For example, different entities may be present in similar records that may be wrongly associated to the same block. As a result of blocking failures like this, a greater number of false pairs of records can be formed and consequently the number of returned candidate pairs are increased unnecessarily.

Another approach for computing the similarity between records is the *Soft TF-IDF* [McCallum et al. 2000] method. This method was initially proposed as a metric of text similarity, enabling one to discover the degree of similarity between different records, considering one attribute at a time. This method requires similarity functions to be selected beforehand by an user, together with the definition

of likelihood parameters.

Latest blocking methods have been developed based on data features and for this reason rely on dynamic approaches to the blocking of records. One of such dynamic methods was proposed in [Yan et al. 2007]. The authors present the *Sorted Neighborhood* algorithm, which works by sliding an imaginary window, which involves records ordered according to a predefined critera. The sorting criteria is defined by a user, thus, records that potentially contain the same entity should become neighbors in the list covered by the imaginary window. In a next step, once the sorting phase is over, after every change of position of the imaginary window, records surrounded by the window are considered to be the same entity and for this reason they compose a block. In this approach, the window has a size that varies during the blocking process, according to the degree of relatedness observed between adjacent records.

Other methods are based on *Machine Learning* [Mitchell 1997] and are therefore also considered adaptive. Examples of such methods are *DNF Blocking* [Bilenko et al. 2006] and *BSL Blocking* [Michelson and Knoblock 2006], which aim at finding the best blocking function for the grouping of records. In these approaches, small clusters of records are used as samples in order to *train* the algorithms that are executed. Based on this samples, the algorithms *learn* about specific features from the data, producing results with higher quality than those observed in previous methods.

Instances of training for the learning process performed by *DNF Blocking* are pairs of records classified as true, if the records contain the same entity, or false, if the records contain different entities. These instances are used in training analysis for the rules choice that produce the best groups of records in the blocking. These rules are selected and combined to form expressions in the disjunctive normal form (DNF) called *blocking schemas*. These expressions define how records are grouped.

DNF Blocking presents better results in cases where static methods fail, because the dynamic and adaptive approaches capture the notion of duplication between records without necessarily considering the similarity between the values of its attributes. However, as stated earlier, an increase in the number of these rules may also imply on an increase on the time needed for the process of machine learning, impairing the use of the *DNF Blocking*. As an alternative to reduce the processing time, the *BSL blocking* method can be used.

BSL blocking is similar to *DNF Blocking* with respect to the use of blocking rules. The major differences are in how the training samples are used and the method of combining predicates to produce blocking strategies in the learning process.

In practice, *BSL Blocking* is usually used with samples containing small numbers of pairs of records of training, considering only the true pairs of the training samples. For that reason, it may take less time to execute, compared to the time of execution of the *DNF Blocking*. This advantage can be used in cases where short processing time is a critical requirement for the grouping of records.

3. BGP – BLOCKING BASED ON GENETIC PROGRAMMING

In this section we describe *BGP* (Blocking based on Genetic Programming), an adaptive blocking method which uses genetic programming (GP) as a basis for solving the problem of adaptive blocking. The *BGP* method, as well as the *DNF Blocking* and *BSL Blocking*, is based on blocking schemas formed from *boolean* predicates for identifying pairs of records that correspond to the same entity in the real world. *BGP* method also requires sample data to find good blocking schemas using machine learning to achieve this goal.

BGP differs from BSL and DBLF mostly in the way the blocking predicates are combined in the process of looking for the best blocking schema. To achieve this goal, those methods use iterative algorithms that analyze combinations of blocking predicates, which are recorded on a given *string*. In

BGP, an evolution process is deployed instead, which tries to maximize the quality of the resulting blocking predicate.

Another important distinction is related to the kind of rules used in these predicates. The rules used in previous methods were called *standard* rules and the rules of the second version of the *BGP* method, called *rules based on parameters*. Thus, the version of our method using *standard* rules is called *BGP-SR* and the version of rules based on parameters, is called *BGP-PR*.

3.1 Using Genetic Programming in Adaptive Blocking

The genetic programming approach can be used to solve adaptive blocking by considering that blocking strategies are computer programs that go through the process of evolution. This idea is developed below.

Representation of Blocking Schemas

In our approach, blocking schemas are represented as trees along the evolutionary process, and which are modified using genetic operators. Originally, however, blocking schemas are modeled as sequences of predicates[1] and *boolean* operators interleaved. An example of such an organization is presented in the following blocking schema:

```
({address, bigramsInCommon} &&
      {venue, termsInCommon} &&
      {publisher, threeFirstCoincidentCharacters}) ||
({title, termsInCommon})
```

For this expression, two records are considered to be related to the same entity, if they present at least a common term in the values of the *title* attribute, or, if this possibility does not arise, if *bigrams* in common are found in *address* attribute values at the same time that any term in common is found in the *venue* attribute and the *publisher* attribute present values with the three first coincident characters. The blocking schema of the previous example can be illustrated in a tree form, as shown in Figure 2.

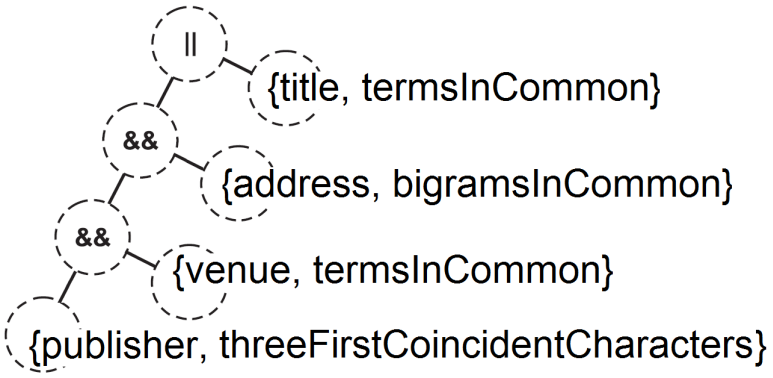

Fig. 2. Representation of a Blocking Schema.

[1]In the *BGP* method, the predicates are similar to those obtained with the *DNF Blocking* and *BSL Blocking* methods, consisting of a combination of attributes and rules of affinity.

In Figure 2, each terminal node corresponds to a blocking predicate and the other nodes are defined by the *boolean* operator *or* (represented by "||") or operator *and* (denoted by "&&"). Each edge represents the relationship between predicates and *boolean* operators, defining the associations between elements of these types for the representation of blocking schemas as trees. By using trees greater flexibility can be achieved for the extraction of segments of blocking schemas expressions, facilitating the use of genetic operators. Blocking schemas in disjunctive normal form are preferred. To identify the schemas with higher chances of producing good results, an efficient *fitness* function is deployed. In the following section, we describe the *fitness* function that we have used in the development of the *BGP* method.

Fitness Function

Blocking schemas are considered good when they help in the efficient identification of all or most of the pairs of duplicate records. In order to measure this quality, allowing comparisons between different schemas and selecting the best option among them during the learning process, some criteria based on coverage of pairs of records should be used, as in other adaptive blocking methods. Common criteria are the *genuine coverage of pairs* (*PC*) and *reduction ratio in the number of candidate pairs* (*RR*).

In the *BGP* method, to measure the quality of blocking strategies, the criteria *PC* and *RR* are used in a combined form, producing a single value to be used as a degree of *fitness*. Using this combination of values, the best blocking schema is the one that present the best performance for both quality criteria.

In information retrieval systems, the well-known *F-measure* [Ricardo Baeza-Yates 1999] is used to combine values of *recall* and *precision* through their *harmonic mean*. We here follow the same idea, but replace *precision* and *recall* by similar concepts *PC* and *RR*, respectively.

Let $R = \{r_1, ...r_n\}$ a set of records to deduplicate and $R_T \subset R$ a set of training records. We define a set of training pairs $P = \{P'_1, ...P'_m\}$, $P'_i = \{(r_j, r_k) \in R_T \times R_T \mid j \neq k\}$ and a set of labels $L = \{l_1, ...l_m\}$, $l_i \in \{0, 1\}$ corresponding to each $P'_i \in P$. In this definition, a pair $P_i \in P$ is called a true pair if $l_i = 1$ and a false pair if $l_i = 0$.

Using the harmonic mean, the number of true and false pairs correctly identified is used for computing the *fitness*, as defined in Equation 1.

$$f_{FIT} = \frac{2}{\frac{1}{PC} + \frac{1}{PF}} \tag{1}$$

By using *fitness* function presented in Equation 1, the blocking schemas considered as good are the ones that present a high genuine coverage of pairs (PC) and that can also achieve a low number of candidate pairs (RR).

However, the quality assessment of blocking schemas can be still be improved. As stated in Section 2, it is noteworthy that blocking schemas with larger numbers of conjunctions of affinity rules can cover larger numbers of true pairs of records. Thus, this idea can be incorporated into the fitness function, by defining the function shown in Equation 2.

$$f^*_{FIT} = f_{FIT} + \left[\frac{C}{100}\right] \tag{2}$$

where C represents the number of conjunctions found in the blocking schema evaluated and f_{FIT} represents the function presented in Equation 1.

The *fitness* function represented in Equation 2 was used for evaluating the blocking schemas during the steps of machine learning carried out in the experiments reported in this article.

GP Algorithm and Parameters Setup

The generic algorithm of GP described in [Koza 1998] adapted to solve the problem of blocking can be seen as the following sequence of steps:

—Randomly generate an initial set of blocking schemas (individuals);
—Evaluate the schemas, using the *Fitness* function;
—Create a new set of schemas (new generation):
  —Copy the best schemas for a new set;
  —Create new schemas by mutation;
  —Create new schemas through *crossover* (reproduction);
  —Evaluate the new schemas, associating them to values of *Fitness*.
—The best schema after a predetermined number of generations is the blocking solution.

When comparing the algorithm adapted to the generic algorithm of genetic programming, it can be noticed that the adaptation was made manly by incorporating the concept of blocking schemas to the structure of the generic algorithm, replacing the general concept of individual for the blocking schema concept.

In the algorithm, some parameters implicit in the structure of genetic algorithm can be observed. Among these parameters, the more important are the ones identified by: (1) number of generations, (2) maximum depth of the tree representing individuals, (3) number of individuals in each generation, (4) number of best individuals copied to the next generation (5) probability of mutation and (6) maximum depth in segments of mutation.

3.2 *BGP-SR* and *BGP-PR* Blocking Methods

The *BGP* method was implemented in two versions: the *BGP-SR*, which is based on *standard* rules, such as those used in *DNF Blocking* and *BSL Blocking*, and *BGP-PR*, which has been defined and experimented with using rules based on parameters.

The complete blocking process, from the selection of rules to the generation of the candidate pairs of record is illustrated in Figure 3. In this figure, we assumed that the choice of the type blocking rules, i.e., with or without parameters, defines the version of the *BGP* method to be used. Then, the blocking process has two-step. First, the genetic programming algorithm is used to produce the blocking schema to be used. For this purpose, samples presenting true and false pairs of records are used for training. In the second phase, the blocking itself is done in a single pass through the records by associating each record to its respective block according to the blocking schema generated.

3.3 Adaptive Blocking using *Standard* Rules - *BGP-SR*

The standard rules used in BGP-SR are the following: (1) perfect match, (2) terms in common, (3) bi-grams in common, (4) tetra-grams in common, (5) hexagrams in common, (6) first character in common, (7) first three characters in common and (8) first five characters in common. These rules are usual deployed in blocking schemas described in the literature [Bilenko et al. 2006; Michelson and Knoblock 2006].

Taking advantage of the random process used for the generation of blocking schemas, a greater number of rules can be used by genetic programming process. This increases the potential diversity of rules available for the analysis of pairs of records, and allows for the generation of blocking schemas that are better adapted to the data sets being processed. However, the use of larger sets of rules may lead to a more time consuming deduplication processes.

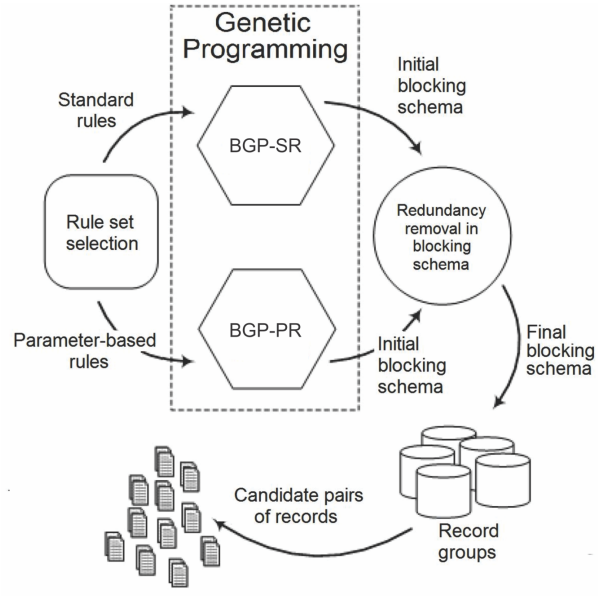

Fig. 3. Adaptive Blocking using the methods *BGP-SR* and *BGP-PR*.

3.4 Adaptive Blocking using rules based on parameters - *BGP-PR*

To achieve a wider variety of predicates, the BGP-PR method uses rules based on parameters. This increases the number of duplicate records that can be predicted when good blocking schemas are generated in each machine learning step.

In the experiments with the *BGP-PR* method, rules based on parameters were defined by generalizing the rules *terms in common* and *n-grams in common*, which are similar to the rules *terms in common* and *bigrams/trigrams/hexagrams in common* from the set of standard rules used by the BGP-SR method. This rules shown in Table I.

Rules	Parameters
1. Terms in common	Number of consecutive terms in common
2. N-grams in common	Length of substrings in common

Table I. Rules based on parameters for the analysis of pairs of records.

The rule *terms in common* can be used to identify pairs of records from the same entity that contain a certain number of consecutive terms in common. In this case, the parameter set corresponds to the minimum number of terms that can be used to identify cases of duplicate records, as can be seen in the pair of records:

record#1: *title*:Models of Machines and Computation for Mapping in Multicomputers
and
record#2: *title*:Models of Machines for Mapping in Multicomputers,

where the terms found are *Models*, *of* and *Machines* in common and in that order. Besides these three consecutive terms, there are four other consecutive terms in common, and these terms are *for*, *Mapping*, *in* and *Multicomputers*. To cover these two cases, the rule *terms in common* can be used with values *three* and *four* for the parameter set. Using these parameter values, the blocking

predicates defined based on the attribute *title* can be represented as *{title, termsInCommon-3}* and *{title, termsInCommon-4}*, respectively.

Using a greater variety of rules, it is expected that the blocking processes will occur with greater flexibility, finding blocks most likely to produce duplicate records, since blocking schemas more suited to the data processed under these conditions can be found.

The results of experiments using the standard rules and based on parameters can be seen in Section 4, in which methods *DNF Blocking*, *BSL Blocking* and *BGP* are evaluated and compared based on the number of record pairs correctly identified and with respect to the number of candidate pairs returned as the result of the blocking processes.

4. EXPERIMENTS

We conducted experiments to evaluate the quality and the scalability of the blocking methods proposed. For this, we used collections *Cora*, *CiteSeer* and *Evolbase*. Collections *Cora* and *CiteSeer* consist of real data and collection *Evolbase* is composed of synthetic data. These collections were used as target datasets in previous studies. *Cora* was used in [Bilenko et al. 2006] to test the method *DNF Blocking*, and collections *CiteSeer* and *Evolbase* where used in [Sarawagi and Bhamidipaty 2002] and [de Carvalho et al. 2006] for deduplication experiments.

Setup

The metrics for measuring the coverage of pairs of duplicate records (PC) and the reduction ratio in the number of candidate pairs (RR) are defined as follows:

$$PC = \frac{||V_C||}{||V||}, \quad RR = 1 - \left[\frac{||C||}{||T||}\right] \tag{3}$$

where $||V_C||$ is the number of pairs correctly identified as duplicate pairs, $||V||$ is the total number of duplicate pairs, $||C||$ is the number of candidate pairs returned as the result of the blocking and $||T||$ is the number of pairs generated with all available records.

In the experiments of scalability, in addition to PC and RR, we also measure the costs due to the machine learning process, which serves as an indication of how efficient are the methods in generating a good solution. In these experiments, we consider the most effective methods as those which verify fewer combinations of predicates to return good blocking schemas as a result of processing at each step. The costs of the machine learning process verified in the experiments of scalability are calculated with the metric CAM, which is defined in Equation 4.

$$CAM = \frac{NCP}{NP} \tag{4}$$

In the metric CAM, NCP represents the number of combinations of predicates that occur during the execution of the blocking methods until a blocking schema is returned as the result in each case, and NP, the number of predicates available for the formation of blocking schemas.

While for the quality experiments we use collections *Cora*, *CiteSeer* and *Evolbase*, in scalability experiments, we only used *Evolbase* collection, since this is a synthetic collection and can be recreated with different numbers of records.

Collections

The main features of the collections used in the experiments of quality are presented in Table II.

Collection	Total number of records	Number of Training Samples		
		Percentage of records	Number of pairs	Average number of true pairs
Cora	1,295	5%	2,016	75
CiteSeer	154	30%	1,035	17
Evolbase	1,000	5%	1,225	5

Table II. Configuration of datasets used in the quality experiments with the collections *Cora*, *CiteSeer* e *Evolbase*.

In the data quality experiments, we used 5% of the total records to compose the machine learning samples. All collections were processed in this way, except the collection *CiteSeer*, where we used samples of 30% of the total amount of the records, since in samples of 5% a few pairs of duplicate records were found, a situation that would undermine the method *BSL Blocking*, since it is based only on examples of duplicate records.

In this data, samples present the averages of 75, 17 and 5 pairs of duplicate records from the collections *Cora*, *CiteSeer* and *Evolbase*, respectively. For verification of scalability, we extracted samples from 5% of total registrations, using data files with the number of records ranging from 1,000 to 8,000 records.

Genetic Programming Parameters

As stated in Section 3.1, the genetic programming algorithm uses a number of parameters that guide its procesing. In Table III we present the configuration of the parameters used by our method for the experiments here reported.

Parameter	Value
Number of generations	5
Maximum depth of the tree representing individuals	4
Number of individuals in each generation	100
Number of best individuals copied to the next generation	4
Probability of mutation	95%
Maximum depth in segments of mutation	3

Table III. Parameters values used with the methods *BGP-SR/PR*.

Quality Results

In Table IV, we present the results of experiments conducted to verify quality of the proposed methods with collections *Cora*, *CiteSeer* and *Evolbase*.

Evaluated Methods	Cora		CiteSeer		Evolbase	
	PC (%)	RR (%)	PC (%)	RR (%)	PC (%)	RR (%)
BGP-SR	90.49	85.54	93.31	87.95	99.18	98.97
BGP-PR	94.72	93.54	91.02	93.02	98.47	98.46
DNF Blocking	92.20	48.57	91.47	82.42	84.08	99.81
BSL Blocking	86.51	39.74	85.15	65.55	92.87	88.59

Table IV. Blocking quality results for the collections *Cora*, *CiteSeer* and *Evolbase*.

In Table IV, we verify the accuracy rate achieved in the detection of pairs of duplicate records. As reported in this table, the lowest percentage of coverage of duplicate pairs (*PC*) were obtained with the *BSL Blocking* method, possibly due to the limited use of examples of pairs of duplicate records and, therefore, a low number of instances to be used by the machine learning process. Even in this case, the percentage of pairs of duplicate records identified was around 85% to 92% in the experiments

performed with the collection *Evolbase*. Methods *BGP-SR*, *BGP-PR* and *DNF Blocking* achieved the best results. In particular, methods *BGP-SR* and *BGP-PR* showed the better performance when evaluated by the reduction of candidate pairs. This is due to the greater efficiency of the genetic programming algorithm when processing a combination of blocking predicates.

Escalability Results

In addition to the rates of correct answers observed in the experiments, we analyzed the costs of machine learning, as well as the costs of grouping the records. We computed the cost of grouping records by verifying the average number of conjunctions found in each blocking schema evaluated, since large amounts of conjunctions can make long-time clustering. Similarly, large amounts of conjunctions can produce larger quantities of blocks of records. Next, we examined the costs of the machine learning processing using as a reference the number of blocking schemas assessed at each stage of the blocking task. In this second form of assessment, we considered the better methods to be the ones that evaluated smaller numbers of combinations of predicates to form blocking schemas during the steps of machine learning. The results of this evaluation are presented in Table V.

Collection	Blocking methods	Learning cost (average of combinations of evaluated predicates in each execution)	Number of considered predicates				Cost per blocking precicate
			Rules (R)	Attributes (A)	Parameters (P)	Number of predicates $(R) \times (A) \times (P)$	
Cora	*BGP-SR*	500	8	12	–	96	5.2
	BGP-PR	500	2	12	10	240	2.1
	DNF Blocking	3,520.8	8	12	–	96	36.7
	BSL Blocking	1,276.8	8	12	–	96	13.3
CiteSeer	*BGP-SR*	500	8	7	–	56	8.9
	BGP-PR	500	2	7	10	140	3.6
	DNF Blocking	675	8	7	–	56	12.1
	BSL Blocking	441.8	8	7	–	56	7.9
Evolbase	*BGP-SR*	500	8	14	–	112	4.5
	BGP-PR	500	2	14	10	280	1.8
	DNF Blocking	4,742.9	8	14	–	112	42.4
	BSL Blocking	394.6	8	14	–	112	3.3

Table V. Machine learning costs for the collections *Cora*, *CiteSeer* and *Evolbase*.

In Table V, we consider the learning cost as represented by the average of the blocking schemas evaluated in the steps of the experiments. These costs were fixed for methods *BGP-SR BGP-PR*, once they evaluated 500 blocking schemas, in which 100 schemas were randomly created in each of the 5 iterations of the genetic programming algorithm. The result for the costs of the methods *DNF Blocking* and *BSL blocking* were obtained by averaging the schemas evaluated in each steps of the performed experiment.

In these results, we also notice that the number of blocking rules was different in the case of the *BGP-PR* method, once this had been tried with different blocking rules from those used in other methods. These rules were parameters based on numerical values ranging from 1 to 10, as well as 10 times the number of predicates available for the formation of blocking schemas. The other methods were evaluated with eight blocking rules, without the use of parameters, since these rules do not require this feature.

The numbers of predicates used in the verification were calculated by multiplying the number of attributes for each collection with the number of rules in each case and then multiplying this result by

the number of parameters available when applicable. The number of available blocking predicates was used in the final calculus, dividing the original learning cost to produce the cost of learning for each predicate available for data processing. In this context, it can be considered that the best methods are the ones that make the lower machine learning costs while assessing the greatest amount of predicates for producing results.

As a result of these experiments, we note that the methods *BGP-PR* and *BGP-SR* had lower costs or close to those of other methods, once they rated the lowest amounts of blocking schemas in almost all stages of experiments, while they considered bigger sets of blocking predicates or ones with the same size when compared to the sets used in other methods during the stages of the experiments.

In a similar analysis to verify the machine learning costs, we also tried to verify costs of grouping the records. These results can be seen in Table VI.

Collection	Method	Number of conjunctions
Cora	*BGP-SR*	11.4
	BGP-PR	10.6
	DNF Blocking	3
	BSL Blocking	8.9
CiteSeer	*BGP-SR*	11.9
	BGP-PR	13.6
	DNF Blocking	2.9
	BSL Blocking	5.9
Evolbase	*BGP-SR*	5.8
	BGP-PR	5.7
	DNF Blocking	2.4
	BSL Blocking	1

Table VI. Average number of blocking schema conjunctions in the collections *Cora*, *CiteSeer* and *Evolbase*.

In Table VI, we can notice that the largest number of conjunctions were found in the experimental results of the methods *BGP-SR* and *BGP-PR*, especially when the collections *Cora* and *CiteSeer* were used, meaning that increased costs were observed in such cases.

We can justify these results, considering that the collections *Cora* and *CiteSeer* which present the greatest difficulties in identifying duplicate records. This idea can be enhanced by what we can observe in the results obtained from the collection *Evolbase*, since the duplicate records can be found with less effort in the midst of such data. In this condition, the methods *BGP-SR* and *BGP-PR* presented schemas with smaller amounts of conjunctions.

However, we notice, that blocking schemas with large amounts of conjunctions do not always have the higher cost of processing. We can verify this by taking as an example 5 blocking schemas verified during the stages of the performed experiments, and the schemas formed by 3, 6, 9, 10 and 17 conjunctions. We measured the time of grouping the records using these schemas, as can be seen in Figure 4.

In Figure 4, we observed that the quantities of predicates were increasing in the order that the blocking schemas were verified. However, the measured times increased until the third assessed schema, only, and declined when we checked the last two schemas, which have larger numbers of conjunctions. After a detailed verification, we noticed that the of evaluation schemas times followed the average variation in each conjunction of predicates, as can be seen in Table VII.

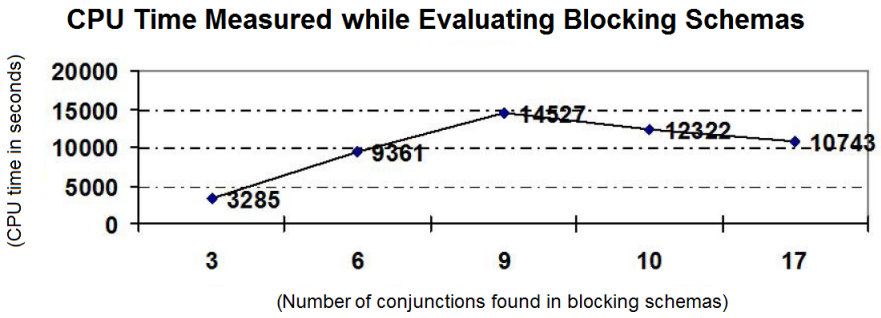

Fig. 4. Evaluation time of blocking schemas.

Blocking schemas	Predicates number in each conjunction	Average of predicates per conjunction	Time in seconds
#1 (3 conjunctions)	{2,2,3}	2.3	3,285
#2 (6 conjunctions)	{1,3,1,3,3,3}	2.8	9,361
#3 (9 conjunctions)	{3,2,2,3,2,3,3,3,3}	3.0	14,527
#4 (10 conjunctions)	{3,1,1,2,3,3,3,3,3,3}	2.5	12,322
#5 (17 conjunctions)	{4,2,2,1,1,1,1,3,1,1,2,1,1,1,1,2,6}	1.8	10,743

Table VII. Average number of Predicates in blocking schema conjunctions.

We may note in Table VII that the average times increased until the third blocking schema was evaluated, coinciding with the increase of the average of verified predicates in these three cases. Then there was a decrease in the measured times for the last two schemas, which showed a lower average by conjunction of predicates, suggesting that smaller times of grouping records than expected may occur in a situations like this.

In Figures 5 (A) (B) (C) and (D), the results of experiments of scalability are presented. As we can see in Figure 5(A), the methods *BGP-SR* and *BGP-PR* remained in good cover of duplicates pairs of records during all steps. Under the same conditions, the *DNF Blocking* showed satisfactory results only when 4,000 or more records were used in the experiments.

Iterative methods such as *DNF Blocking* may require too many samples to find good blocking schemas. In the case of *DNF Blocking*, each iteration of the main algorithm, returned a conjunction of blocking predicates. Therefore, the blocking schemas discovered with this method can be viewed as sequences of conjunctions. If few records are used in the machine learning training set , schemas showing few conjunctions can be formed, possibly failing to detect pairs of duplicate records at later times.

In Figure 5(B), we can verify the reductions achieved in terms of number of candidate pairs of records. As in previous checks, the *BSL Blocking* method was less stable, possibly due to the usage of smaller quantities of samples compared to other tested methods. In the experiments we also measure the machine learning costs, which are presented in Figure 5(C) as a way to identify the methods that had the lowest implementation costs. In these reviews the methods of genetic programming were those with the lowest cost, as the populations were kept small, with 500 blocking schemas tested at each stage of machine learning.

In the inspection of processing cost, the *DNF Blocking* method showed a decrease in costs as the

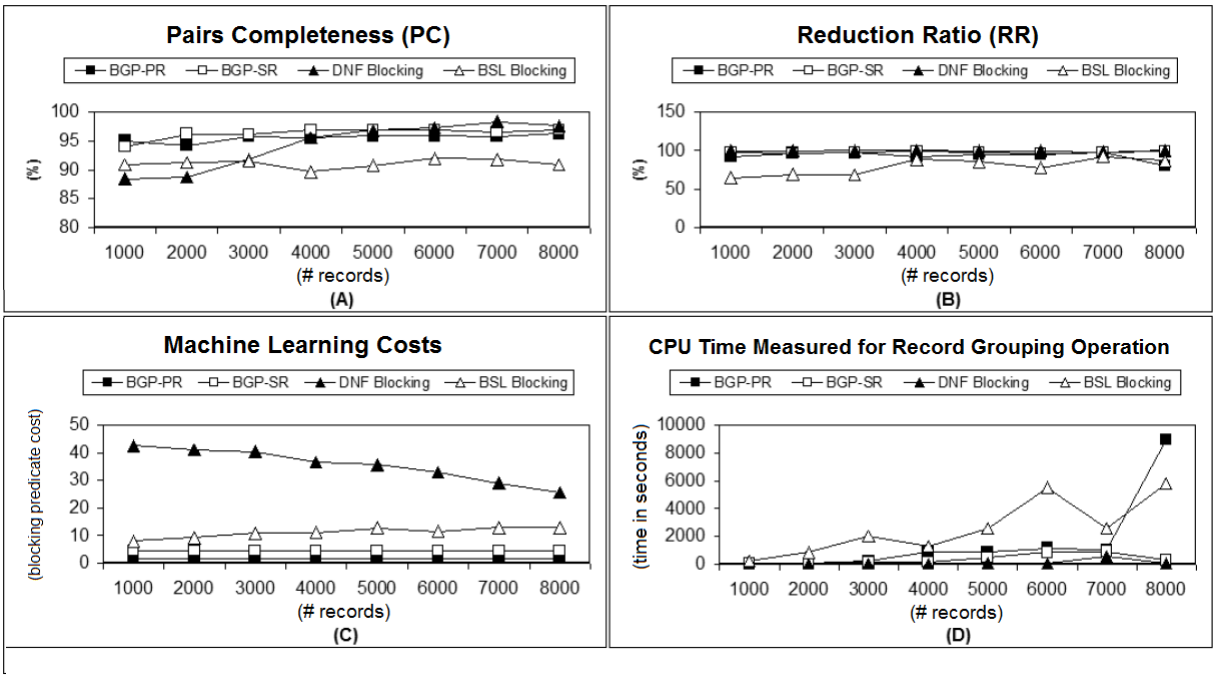

Fig. 5. Evaluating scalability aspects of Blocking Methods.

number of records were increased. We observed this decrease and noticed that it was due to predicates that were discarded at the beginning of each stage of execution.

For the scalability experiments, we also measured the processing times in seconds of CPU when the blocking schemas were used to group records. Those times had significant differences and therefore we do not regard them as a reference for verifying the scalability of the methods of blocking.

We can observe these times in Figure IV(D), as finding time for the grouping of records in each situation. We can note that the best methods were the *DNF Blocking*, *BGP-SR* and *BGP-PR* up to the step with 7000 records. In step with 8000 records, the method *BGP-PR* spent more time performing the grouping of records because of blocking schemas formed by a larger number of predicates based on rules of *n-grams*. These rules may delay the process, since a larger number of blocks can be produced when the data is processed, generating many similar blocks and hence many pairs of duplicate records are repeated. Situations like this can be justified in cases where the difficulty of detection of duplicate records is a difficult goal to be reached.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented two the adaptive blocking techniques *BGP-SR* and *BGP-PR*, both based on genetic programming. We experimentally compared these methods with state-of-art adaptive blocking methods in the literature and concluded that our methods are efficient and scalable to deal with a large number of blocking predicates. Using larger numbers of predicates is an advantage, since the blocking processes can be performed with more flexibility and the identification of duplicate records have greater chances of success. We have shown that by using genetic programming satisfactory results can be achieved without analyzing a very large number of blocking combinations of predicates, resulting in a significant reduction in machine learning costs. In future work, processing costs can still be reduced through active learning, once we start to use smaller samples of data in machine learning processes, and thus reducing the time required to check each combination of predicates.

REFERENCES

Bhattacharya, I. and Getoor, L. Iterative record linkage for cleaning and integration. In *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*. Paris, France, pp. 11–18, 2004.

Bilenko, M., Kamath, B., and Mooney, R. J. Adaptive blocking: Learning to scale up record linkage. In *Proceedings of the IEEE International Conference on Data Mining*. Hong Kong, China, pp. 87–96, 2006.

de Carvalho, M. G., Gonçalves, M. A., Laender, A. H. F., and da Silva, A. S. Learning to deduplicate. In *Joint Conference on Digital Libraries*. Chapel Hill, USA, pp. 41–50, 2006.

Evangelista, L. O., Cortez, E., da Silva, A. S., and Jr., W. M. Blocagem adaptativa e flexível para o pareamento aproximado de registros. In *Proceedings of the Brazilian Symposium on Databases*. Fortaleza, Brazil, pp. 61–75, 2009.

Koza, J. R. *On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1998.

McCallum, A. K., Nigam, K., and Ungar, L. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the International Conference On Knowledge Discovery and Data Mining*. Boston, USA, pp. 169–178, 2000.

Michelson, M. and Knoblock, C. A. Learning blocking schemes for record linkage. In *Proceedings of the National Conference on Artificial Intelligence*. Boston, USA, 2006.

Mitchell, T. M. *Machine Learning*. McGraw-Hill, New York, 1997.

Ricardo Baeza-Yates, B. *Modern Information Retrieval*. ACM Press, 1999.

Sarawagi, S. and Bhamidipaty, A. Interactive deduplication using active learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, pp. 269–278, 2002.

Winkler, W. Advanced methods for record linkage, 1994.

Winkler, W. E. Overview of record linkage and current research directions. Tech. rep., Statistical Research Division, U.S. Census Bureau, Washington, DC, 2006.

Yan, S., Lee, D., Kan, M.-Y., and Giles, L. C. Adaptive sorted neighborhood methods for efficient record linkage. In *Proceedings of the ACM/IEEE joint conference on Digital libraries*. Vancouver, BC, Canada, pp. 185–194, 2007.