Record Linkage as a Multiobjective Optimization Problem solved by Evolutionary Computation

Francesco Cusmai
SAPIENZA – Univ. Roma, ITALY
cusmai.francesco@gmail.com

Carola Aiello
SAPIENZA – Univ. Roma, ITALY
carola.aiello@dis.uniroma1.it

Monica Scannapieco
ISTAT and SAPIENZA – Univ. Roma, ITALY
scannapi@istat.it

Tiziana Catarci
SAPIENZA – Univ. Roma, ITALY
catarci@dis.uniroma1.it

ABSTRACT

Record linkage is an important problem to be solved in all contexts in which data integration is to be performed under the hypothesis of data sources that can exhibit errors.

In this paper we formulate the record linkage problem as a multiobjective optimization problem and we propose an algorithm for solving it based on evolutionary computation. This computational paradigm has a stochastic, adaptive and intrinsically parallel nature that makes it particularly suitable for a peer-to-peer integration context.

Experiments on real datasets proved the effectiveness and efficiency of this novel approach.

1. INTRODUCTION

Peer data management systems (PDMSs) are distributed data management systems where peers interact according to a peer-to-peer paradigm. Differently from traditional data integration systems, no centralized global schema is present, but each peer maps its own schema to schemas of other peers on a local basis. These systems are characterized by large scale, high dinamicity and openness.

Query answering in these systems is realized by exploiting peer mappings that describe the semantic relationships between the schemas of pairs (or small sets of) peers. In order to specify such mappings, it is important to join the different sources on identifying attributes. In real integration scenarios, such attributes may be affected by errors, that is joins cannot be performed in an exact way, but some approximation must be introduced.

In order to implement approximate joins for the purpose of integrating data, a record linkage activity must be performed. Given two (or more records) record linkage (RL) has the purpose of comparing them on some common attributes and deciding if they are a match or not, on the basis of the result of such a comparison. The record linkage problem has been studied since more than five decades, but in the context of peer data management systems, it presents new research challenges.

More specifically, record linkage techniques can be classified into: probabilistic, knowledge-based and empirical [2]. The main reference for the probabilistic RL literature is the Fellegi and Sunter model [8]; many different methods have been proposed to estimate the parameters of such a model, most of them based on the Expectation-Maximization method [12, 22]. The probabilistic approaches have problems with respect to the estimation of the Fellegi and Sunter model's parameters with large data sets [23].

In knowledge-based techniques (e.g., [17]) the RL process is based on the use of rules. These rules represent the domain knowledge that is extracted from the data sources to be matched, and reasoning strategies are applied to make the process more effective. These techniques are typically based on the usage of training sets to learn rules, hence this is a scenario difficult to implement due to the dinamicity and openness features of a PDMS.

Empirical techniques (see e.g., [7]) focus mainly on the search space reduction problem and are typically used in conjunction with techniques of the cited paradigms.

To overcome these limitations, we propose to solve the record linkage in a PDMS setting by using a completely novel approach, based on evolutionary computation. Evolutionary algorithms have all the features desired for this specific setting: they are stochastic, adaptive and intrinsically parallel search algorithms. Indeed, the stochastic feature and the intrinsic parallelism can help to face problems deriving from large scale; adaptiveness is very important for the dinamicity and openness of PDMSs.

More specifically, in this paper, we first formulate the record linkage problem as a multiobjective optimization problem. Then, we extend an evolutionary algorithm named NSGA-II, used for solving multiobjective optimization problems, in order to solve our problem. We experimentally show that our algorithm has good performance, both in terms of effectiveness and efficiency.

The rest of the paper is organized as follows. After some background definitions, provided in Section 2, Section 3 describes two formulations of the record linkage problem. Then, in Section 4, we describe the details of the proposed algorithm. Finally, we present our experimental results in Section 5 and we draw some conclusions in Section 7. The related work is presented in Section 6.

2. DEFINITIONS

Before presenting the details of our algorithm we provide some background definitions. In general, multiobjective optimization problems can be defined in the following way [4, 3]:

$$Min/Max \quad f_m(x) \quad m = 1, 2, ..., M \tag{1}$$
$$g_j \geqslant 0 \quad j = 0, ..., J \tag{2}$$
$$h_k = 0 \quad k = 1, ..., K \tag{3}$$
$$x_i^L \leqslant x_i \leqslant x_i^U \quad i = 1, ..., n. \tag{4}$$

Where f is the function to optimize, *(2)* and *(3)* represent constraints to be satisfied and *(4)* is the interval of values that x_i can assume.
The differences between multiobjective optimization problems and the single-objective ones are:

- M objectives instead of a single one;
- beside the n-dimensional space of the admissible solutions (which is also present in single-objective optimization problems), there is a further M-dimensional space whose coordinates correspond to the M objectives of the problem;
- generation of a set of solutions.

In a multiobjective optimization scenario a different definition of optimum is needed. In the following we introduce the notion of Pareto optimum that is the one mainly adopted in these cases.

Definition - Pareto Optimality *A solution $x \in \Omega$ is Pareto optimal in Ω if and only if there is no $y \in \Omega \quad : v = F(y) = f_1(y), f_2(y), ..., f_k(y)$ dominates $u = F(x) = f_1(x), f_2(x), ..., f_k(x)$.*

To state if a solution x is better than a solution y we refer to the following definition:

Definition - Pareto Dominance *A vector $u = (u_1, ..., u_k)$ dominates a vector $v = (v_1, ..., v_k)$ $(u \preceq v)$ if and only if $\forall i \in \{1, .., k\} \quad u_i \leqslant v_i$ and $\quad \exists i \in \{1, .., k\}\ u_i < v_i$.*
With this ordering relation between the solutions an optimal set will be defined in the following way:
Definition - Pareto Optimal Set *Let us consider an optimization problem where $F(x)$ is the function to be optimized, the Pareto Optimal Set $\mathcal{P}$ is: $\mathcal{P} = \{x \in \Omega \quad | \quad \nexists y \in \Omega \quad F(y) \preceq F(x)\}$.*

For each decision variable belonging to the Pareto Optimal Set there is a correspondent vector in the space of the objective function as stated by the following definition:

Definition - Pareto Front *Let us consider an optimization problem with an objective function $F(x)$ and a Pareto Optimal Set $\mathcal{P}$, the Pareto Front $\mathcal{F}$ is defined as:*
$\mathcal{F} = \{u = F(x) \quad | \quad x \in \mathcal{P}\}$.

Multiobjective optimization problems can be solved with evolutionary algorithms [1]. Evolutionary computation is a paradigm inspired by the biological evolution process that is ruled by natural selection [13]; it includes techniques of genetic algorithms, evolution strategies and evolutionary programming [10]. The use of evolutionary algorithm to solve multioptimization problems is mainly due to the population-based nature of these algorithms which allows the generation of several elements of the Pareto Optimal Set in a single run. A formal definition for an evolutionary algorithm is provided in the following.

Definition - Evolutionary Algorithm *An evolutionary algorithm is defined as an 8-tuple*
$EA = (I, \Phi, \Omega, \mu, \lambda, s, \iota, \Psi)$
where:

- $I = A_x \times A_s$ is the space of the individuals with A_x e A_s arbitrary sets.
- $\Phi : I \longmapsto \mathbb{R}$ denotes the fitness function which assigns real values to the individuals.
- $\Omega = \{\omega_{\Theta i}, ..., \omega_{\Theta z} \quad | \quad \omega_{\Theta i} : I^\lambda \mapsto I^\lambda\} \cup \{\omega_{\Theta 0} : I^\mu \mapsto I^\lambda\}$ is a set of probabilistic genetic operators $\omega_{\Theta i}$, each one controlled by specific parameters contained in $\Theta_i \subset \mathbb{R}$. λ is the number of *son-individuals* and μ is the number of *father-individuals*.
- $s_{\Theta i} : (I^\lambda \cup I^{\mu+\lambda}) \mapsto I^\mu$ denotes the selection operator that can change the number of individuals from λ to $\mu + \lambda$ where $\lambda = \mu$ is allowed. A further set of parameters Θ_s can be used by the selection operator.
- Finally $\iota : I^\mu \mapsto \{true, false\}$ is an ending condition for the evolutionary algorithm and the transition function $\Psi : I^\mu \mapsto I^\mu$ describes the transformation process of a population P into the next one by applying the genetic operators and the selection:
  - $\Psi = s \circ \omega_{\Theta 1} \circ ... \circ \omega_{\Theta j} \circ \omega_{\Theta 0}$
  - $\Psi(\mathrm{P}) = s_{\Theta i}(\mathrm{Q} \bigcup \omega_{\Theta i_1}(...(\omega_{\Theta i_j}(\omega_{\Theta 0}(\mathrm{P})))...))$
  - $\{i_1, ..., i_j\} \subseteq \{1, ...z\} \quad \mathrm{Q} \in \{\emptyset, \mathrm{P}\}$

3. PROBLEM FORMULATION

Let us consider two files, say A and B, of dimension n and m respectively. Without loss of generality we assume that $n \leqslant m$.

In our work we formulate the RL problem as a multi-objective optimization problem, according to two distinct formulations, detailed in the following.

FORMULATION I

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} F(y_i^A, y_j^B) \cdot x_{ij} \tag{5}$$
$$\sum_{j=1}^{m} x_{ij} \leqslant 1 \tag{6}$$
$$\sum_{i=1}^{n} x_{ij} \leqslant 1 \tag{7}$$
$$F(y_i^A, y_j^B) = \begin{bmatrix} f_1(y_{i1}^A, y_{j1}^B) \\ f_2(y_{i2}^A, y_{j2}^B) \\ \cdot \\ \cdot \\ \cdot \\ f_k(y_{ik}^A, y_{jk}^B) \end{bmatrix} \tag{8}$$
$$x_{ij} \in \{0, 1\} \tag{9}$$

Variables x_{ij} are equal to 1 when record j of file B has been assigned/matched to record i of file A and 0 otherwise.

The function $F(y_i^A, y_j^B)$ calculates the distances between fields of record i of file A and fields of record j of file B and it is composed by the functions $f_1(y_{i1}^A, y_{j1}^B), ..., f_n(y_{in}^A, y_{jn}^B)$. These functions can be chosen among a set of comparison functions between strings such as Edit Distance, Jaro and others [16]. Note that a different function can be used for comparing different fields.

Constraints (6) and (7) assure that the match is one to one. Specifically, a record of the file A can be matched with at most one record of the file B and viceversa.

The goal is to maximize the total values for comparison functions.

Example.

Let us suppose to run a RL procedure between table 1 and table 2, shown in the following. As a first step we focus on the common attributes of file A and file B thus obtaining files A' and B' shown in tables 3 e 4.

Table 1: File A

N	Surname	Address	Income
1	Smith	Wasington str.	20000
2	Smith	Lincoln str. 16	10000
3	White	Lincoln str. 90	50000

Table 2: File B

N	Address	Name	Sex	Surname
1	Washington str. 16	Antony	M	Smith
2	Lincoln str.	Lucy	F	Smit
3	Lincoln str. 90	Mary	F	White
4	Kennedy str. 90	John	M	Whit

Table 3: File A'

N	Surname	Address
1	Smith	Wasington str.
2	Smith	Lincoln str. 16
3	White	Lincoln str. 90

Table 4: File B'

N	Surname	Address
1	Smith	Washington str. 16
2	Smit	Lincoln str.
3	Whit	Lincoln str. 90
4	White	Kennedy str. 90

The problem formulation for this input is:

$$\max \sum_{i=1}^{3} \sum_{j=1}^{4} F(y_i^{A'}, y_j^{B'}) \cdot x_{ij}$$
$$\sum_{j=1}^{4} x_{ij} \leqslant 1$$
$$\sum_{i=1}^{3} x_{ij} \leqslant 1$$
$$F(y_i^{A'}, y_j^{B'}) = \begin{bmatrix} EditDistance_1(y_{i1}^{A'}, y_{j1}^{B'}) \\ EditDistance_2(y_{i2}^{A'}, y_{j2}^{B'})) \end{bmatrix}$$
$$x_{ij} \in \{0,1\} \qquad i = 1,..,3 \quad j = 1,..,4$$

Notice that for both *Surname* and *Address* we have chosen the edit distance as comparison function.

An optimal solution for this case is obtained with the following assignment of the variables:

$$x_{11} = 1 \quad x_{22} = 1 \quad x_{33} = 1$$
$$\forall ij \quad (1 \leq i \leq 3 \wedge 1 \leq j \leq 4 \wedge i \neq j) \rightarrow x_{ij} = 0.$$

Formulation I does not take into account any threshold value when comparing different record fields on the basis of similarity functions. However, this is a common input to several record linkage procedures; therefore, we include these thresholds in a second problem formulation, detailed in the following.

FORMULATION II

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} (F(y_i^A, y_j^B) \cdot x_{ij}) \cdot t_{ij} \tag{10}$$
$$\sum_{j=1}^{m} x_{ij} \leqslant 1 \tag{11}$$
$$\sum_{i=1}^{n} x_{ij} \leqslant 1 \tag{12}$$

$$F(y_i^A, y_j^B) = \begin{bmatrix} f_1(y_{i1}^A, y_{j1}^B) \\ f_2(y_{i2}^A, y_{j2}^B) \\ . \\ . \\ . \\ f_k(y_{ik}^A, y_{jk}^B) \end{bmatrix} \quad (13)$$

$$\mathcal{V} = \begin{bmatrix} \alpha_1 \\ . \\ . \\ . \\ \alpha_k \end{bmatrix} \quad (14)$$

$$t_{ij} = 1 \quad iff \quad F(y_i^A, y_j^B) \geqslant \mathcal{V} \text{ and } x_{ij} = 1 \quad (15)$$

$$\alpha \in [0, 1]$$

$$t_{ij} \in \{0, 1\}$$

$$x_{ij} \in \{0, 1\}$$

The objective of this problem formulation is the maximization of the global similarity values of those match that satisfy the condition $F(y_i^A, y_j^B) \geqslant \mathcal{V}$ and $x_{ij} = 1$ where $\mathcal{V}$ is a vector of similarity thresholds that a match must exceed or at least be equal in order to be considered a true match. For this second formulation of the problem we developed an algorithm that uses a specific mutation genetic operator in order to focus the search. The genes of the individual of the population that is submitted to mutation are changed in a selective way. The mutated genes are those that correspond to matches that do not satisfy the condition $F(y_i^A, y_j^B) \geqslant \mathcal{V}$ and $x_{ij} = 1$.

4. EARL: AN EVOLUTIONARY ALGORITHM FOR RECORD LINKAGE

In this section we describe the algorithm we propose for record linkage, called EARL (Evolutionary Algorithm for Record Linkage). The aim of the algorithm is to generate a matching of records where the total distance between each attribute selected for the match is maximized, according to the given problem formulations.
The algorithm is composed by the phases shown in Figure 1, which are described in the following.

4.1 Phase I: Initialization and Creation of the First Population

The parameters that are input to EARL are:

- The dimension of the initial population, the number of generations and a parameter, named x, which is a negative integer that supports the creation of individuals (as better detailed in the following). All these parameters are set on the basis of the results we have obtained from tests on real datasets, detailed in Section 5.

- Matching attributes, that is those attributes of the input files that will be actually compared for taking the matching decision. These attributes are chosen by a domain expert.

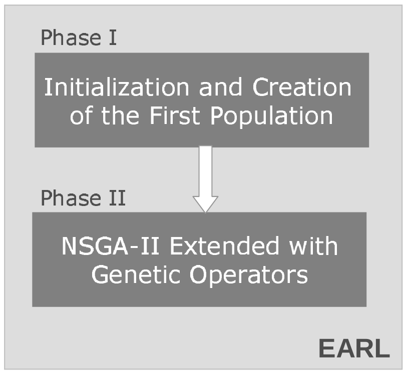
Phase I
Initialization and Creation of the First Population
Phase II
NSGA-II Extended with Genetic Operators
EARL

Figure 1: The phases of EARL

In the following, we will indicate as *donor file*, the file which has the highest number of records; instead, we will indicate as *receiving file* the other one. The donor and receiving files are read in two multidimensional arrays, with a number of rows equal to the records of each file and a number of columns equal to the number of attributes selected for the match. The first population is generated in a random way starting from the files which are input to the record linkage procedure. Let m be the dimension of the donor file and n be the dimension of the receiving file. Each individual of the initial population, say v, is represented by an array of objects with dimension n. Each object is a vector of k values corresponding to values of the comparison function function for each attribute in case of a match, it is vector of zero otherwise. The following pseudocode shows how each individual is created.

```
for $i$=1 to $n$
    $z$=randomGen(x,m)

    if $z$<0
            $[v_{i1} \dots v_{ik}] = [0, \dots, 0]$

    else if $z$=q>0

            $[v_{i1} \dots v_{ik}] = [f_1(y_{i1}^{don}, y_{q1}^{rec}), \dots, f_k(y_{ik}^{don}, y_{qk}^{rec})]$
```

The function `randomGen` generates a random value between `x` and `m`, where `x` is a parameter set in dependence of the value of `m` (see Section 5); the idea is to select randomly the index of the record of the donor file to match (or not) with the current index of the record of the receiving file. The variable z has a negative value in the case of a decision of not assigning a match for the corresponding records which have been compared; it has a positive value otherwise.
In order to satisfy the RL one-to-one constraint, when creating the new individual a preventive check is performed to avoid the creation of inadmissible solutions.

4.2 Phase II: NSGA-II extended with Genetic Operators

4.2.1 *NSGA-II*

NSGA-II is a multi-objective evolutionary algorithm that sorts the solutions according to the Pareto non-dominance principle. The steps of NSGA-II are the followings [4, 3, 5]:

```
$NSGA-II(N, g, f_k(x_k))$
$N$ number of individuals, $g$ number of generations, $f_k(x_k)$ function to be optimized

1. Initialization of a population $P_1$ of size $N$
2. Evaluation of the individuals for each objective
3. Assignment of a rank to each individual in function of a ranking based on the Pareto dominance principle
4. Generation of individuals from $P_1$ in order to obtain a population $P'_1$ of size $2N$
   (a) Extraction of individuals from $P_1$ with a selection
   (b) Crossover and mutation of the extracted individuals
5. $for \quad i = 1 \quad to \quad g$
   For each individual of population $P'_i$ with $i > 1$
      (a) According to the Pareto dominance principle, assignment of a rank to each individual and conforming partition of the population into non-dominated fronts $F_k$
      (b) Creation of a new population $P_{i+1}$ of size $N$ from $P'_i$
          for $j = 0 \quad to \quad j < k \; F_j.size < N - P_{i+1}.size$
             sorting the solutions of $F_j$ by the so-called crowding distance [1]
          $P_{i+1} = P_{i+1} \cup F_j$
      (c) If $N - P_{i+1}.size > 0$
             sorting of the solutions of $F_j$ according to the crowding distance
             adding of the first $N - P_{i+1}.size$ of $F_J$ to $P_{i+1}$
      (d) Creation of $P'_{i+1}$ through selection, crossover and mutation on $P_{i+1}$
```

The algorithm creates a population of individuals, it sorts each individual according to its level of non-domination then it applies the evolutionary operators to the population. The favorite individuals for the new population are those with the best rank and with the best crowding distance (elitism).

4.2.2 *Input Parameters to NSGA-II*

The algorithm NSGA-II is initialized in the following way:

- N is the initial population where each individual represents a linkage between the receiving file and the donor file;
- g is the number of generations;
- the function to be optimized is:

$$F(v) = \sum_{i=1}^{n} [v_{i1}, \ldots, v_{ik}] = [v_1, \ldots, v_k] \qquad (16)$$

  where v_1 denotes (a value for) the global similarity between the strings of the field that corresponds to the first matching attribute of the record of the receiving file and of the donor file of the individual v.
- the genetic operators defined in the following section.

4.2.3 *Definition of the Genetic Operators*

In this section we describe the genetic operators used in EARL. The selection operator is binary tournament, as typically used in NSGA-II.

The binary tournament genetic operator selects the individuals of the population in the following way:

1. a sorted array whose length is equal to the dimension of the population is created, each field of the array contains an integer that represents an individual of the population, the array can contain the same elements;
2. it takes the individual i and the individual $i + 1$, selected between the followings:
   - the individual with the best rank;
   - the individual with the best crowding distance, when having the same rank;
   - the individual is randomly selected, when having the same rank and the same crowding distance.

The crossover and mutation operators have been newly defined in order to fit our specific problem.
Our crossover operator takes as input two individuals and returns two individuals built with the best genes of the starting individuals. To perform this kind of crossover the genes of the starting individuals are compared one by one, if there exists a gene that dominates (according to the Pareto Dominance principle) the other, this gene is selected to create the new individual, otherwise, if none of the individuals dominate another one, the choice will be random.
Our mutation operator changes those genes that correspond to values of the similarity function that will probably lead to false-matching. Specifically the similarity functions assume values between 0 and 1 and if the value computed between any two attributes of two records is 0 then with a high probability the two records don't represent the same entity.

The genes of the individual submitted to mutation are changed in part according to the described mutation operator and in part randomly. The described mutation is mainly performed in the first generations and becomes less frequent in the next generations when the ratio between non-good genes and the total genes is less significant. On the other hand the random mutation, that is always applied to the same part of the population during all the generations, avoids the risk of a too premature convergence and the risk of falling into local maxima of the function.
In order to satisfy the RL one-to-one constraint, when creating the new individual a repairing action is performed.

[1]The crowding distance is computed for each objective i in the following way:
- sorting of the solutions according to the objective i;
- computation of the distance of the individual s as the difference between the values of f_i assumed by the previous and by the next individual of s, this value is normalized with the difference that the first and the last individuals assume for objective i;
- he crowding distance of individual s will be the sum of these distances obtained for each objective i.

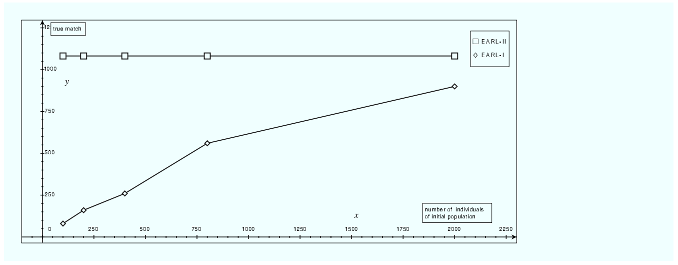

Figure 2: Number of true matches vs. dimension of initial population

5. EXPERIMENTS

This section describes a set of experiments performed with the objective of testing the effectiveness and efficiency of EARL.

We have used real data sets about personal data related to Rome citizens, owned by an Italian public administration. Specifically, we have run experiments on two datasets that were populated by different processes, each of size of about 8000 records. On these datasets, in the following dataset A and dataset B, a record linkage activity had already been performed in the context of a project lasted several months and involving both deterministic and probabilistic procedures. A significant human intervention was also necessary in order to decide the status of matching of records. We used these already available results as a benchmark for evaluating the effectiveness of EARL.

As far as the technological platform, we ran experiments on a PC with CPU of 2 Ghz and RAM of 1 GB, with OSX operating system. EARL was developed in Java. A tool called jmetal [14] supported the development of the evolutionary algorithm, and an open source java library was used for string comparison functions [19].

5.1 Effectiveness and Efficiency Experiments

In order to better study the behavior of EARL, we extracted 3 subsets of data from datasets A and B, respectively of sizes 300, 600, and 1000.

First of all we studied how to set up the dimension of the initial population.

Figure 2 shows how the number of true matches increases with the dimension of the initial population for the two problem formulations we have provided. Specifically, algorithm EARL 1 corresponds to the first problem formulation: the curve consistently grows up with number of individuals. Algorithm EARL 2 corresponds to the second problem formulation, in which thresholds on the matching attributes have all been fixed to 0.8. As it is shown, the EARL 2 curve is constant when varying the number of individuals of the initial population, which is a quite interesting finding. In order to understand the effect on the time behavior of this result, we have studied the execution times, when varying the number of individuals of the population. In Figure 3, we report the execution time for the dataset of 1000 records: obviously the time increases with the dimension of the initial population, though quite slowly, confirming the substantial independence of the EARL 2 algorithm from the dimension of the initial population.

A second set of experiments was made in order to consider how to fix the number of generations. Such experiments

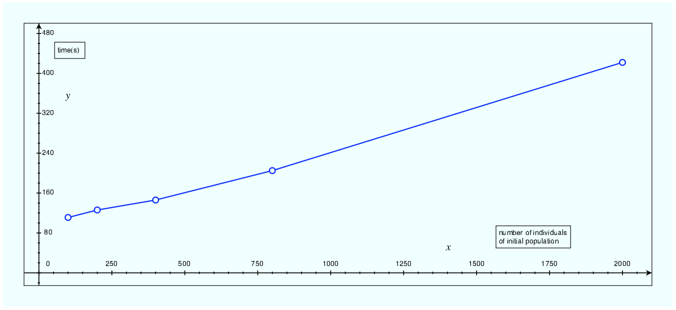

Figure 3: Execution time vs. dimension of initial population

(that we do not report for lack of space) allowed to study the number of *evaluations*, defined as the product between the number of individuals and the number of generations. The experiments showed:

- A few number of generations was necessary for the convergence of EARL 1. [2]
- how to fix the number of generations of EARL 2, in dependence of the behavior of the number of evaluations when varying the dimensions of the input datasets.

A third set of experiments allowed us to choose a value for the `x` parameter (see Section 4). We have observed that the parameter has an impact on how fast the algorithm converges, and hence on effectiveness and efficiency. More specifically, high values of `x` leads to a low effectiveness, remaining constant the number of generations and the dimension of the population. Hence, a rule of thumb for the choice of `x` is to consider quite low values, between 2% and 5% of the dataset with the highest dimension.

Finally, in Figure 4, we report the effectiveness performance of EARL 1 and EARL 2 that must be compared to the real performance figures we had for our benchmark data. The number of true matches is reported for EARL 1, EARL 2 and the real data. We can notice that:

- the performance of EARL 1 are quite good for datasets of 300, 600 and 1000 records, while become dramatically low for the 8000 record data set. This is mainly due to the fact that for such data set it was not possible, due to memory limitations of the experimental environment, to increase the number of individuals of the initial population in order to have better performance.
- the performance of EARL 2 (which are independent on the number of individuals of the initial population) are instead very good, even when compared with exact methods for record linkage, being the percentage of the true matches always higher than 80% of the real true matches with peaks of more than the 90%.

In synthesis, algorithm EARL 1 is really independent on the data, and does not need to fix parameters like threshold values that require an accurate analysis of the data at hand.

[2]Actually we verified that the behavior of EARL 1 can lead to a premature convergence to local maxima independent on the number of generations.

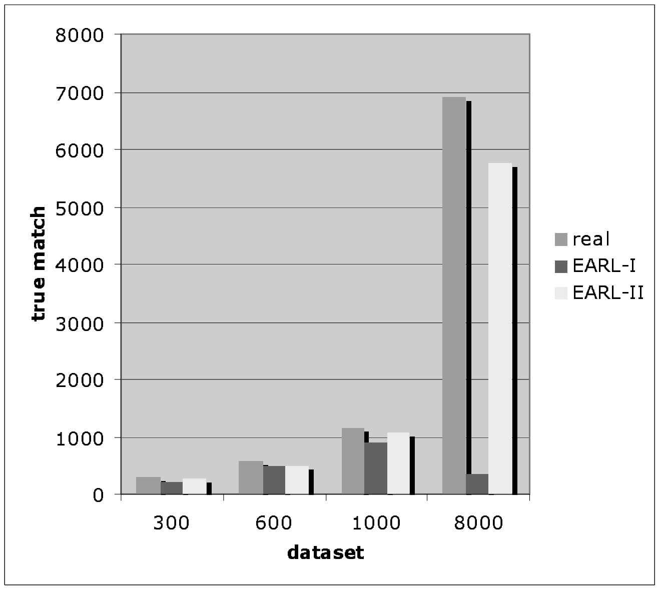

Figure 4: True matches for EARL 1, EARL 2 compared the real true matches

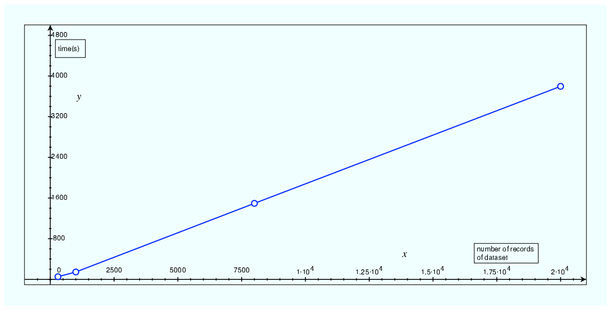

Figure 5: Execution time vs. dimensions of input datasets

On the other hand it can have performance that are not acceptable (but can be refined by appropriate tuning some parameters). Algorithm EARL 2 exhibits very good performance but threshold values must be fixed for the matching attributes.

Finally, we ran an experiment with the objective to test the time efficiency of the two algorithms (EARL 1 and EARL 2 of course have the same time performance, implementing the same computational steps). In Figure 5, we show the time necessary for the execution versus the dimensions of the input datasets, to which we added a 20000 record datasets, obtained synthetically from the 8000 record one. We can observe that the such performance are really good: for the maximum size datasets, that is 20000 records each, the time is of slightly more than one hour.

6. RELATED WORK

In this section, we briefly review some works of the area of multiobjective optimization through evolutionary algorithms.

In 1985 Schaffer [20] implemented VEGA (Vector Evaluated Genetic Algorithm), the first multiobjective genetic algorithm to find a non-dominate solution set. On one hand this algorithm has the advantage of an easy implementation, on the other the algorithm could, with an high probability, prefer in the selection step solutions that are near the optimum only for one single objective. In 1993 Hajela e Lin [18] proposed a genetic algorithm based on weights (WBGA). Each objective function is multiplied by a certain weight; the sum of these weighted objective functions represents the fitness function. The algorithm has a low complexity, but it is difficult to obtain a uniform diversity of non-dominated solutions, because a uniform diversity of the weights doesn't necessarily correspond to a uniform diversity of solutions. Fonseca and Fleming in 1993 [11] introduced a multiobjective genetic algorithm (MOGA) that uses non-domination population classification. At each individual of the population a rank is assigned. This rank is calculated as a function of the number of individuals that dominate it. In this way non-dominated individuals have the same rank. In particular cases, the algorithm can produce an unexpected bias in the direction of some solutions in the search space. In 1989 Goldberg proposed to use the concept of non-dominated sorting for genetic algorithms. This idea was implemented by Srinivas and Deb in 1994 [21] in Non-dominated Sorting Genetic Algorithm (NSGA). The algorithm performance is strongly dependent on the minimum distance that two solutions must have in order that one doesn't influence the other's fitness value. In 1998 Zitler e Thiele [25] proposed an elitist evolutionary algorithm (SPEA). In each iteration the external population is populated with the best individuals. In case that the insertion of new individuals makes the external population exceed a certain dimension, it will be reduced with clustering techniques. Fitness assignment procedure is easy to calculate and a clustering algorithm ensures a good spread but a large external population increases the selection pressure for the elites and a small external population cancels the effect of elitism. Moreover there is a bias in fitness assignment dependent on the exact population and densities of solutions in the search space. In 2000 Knowles e Corne [15] proposed Pareto-Archived Evolution Strategy (PAES). The approach consists in keeping a small/limited archive of non-dominated solutions. The archive is updated in the next generations in order to keep non-dominated solutions with a good degree of diversity. The problem of this algorithm is the definition of the archive's size: if too small it can loose optimal solutions, if too big it can make the algorithm inefficient. The algorithm used in this paper, namely NSGA-II [5, 6], implements a specific strategy in order to keep the best individuals, and a mechanism for preserving the diversity among the different solutions. We have chosen this algorithm as a basis for EARL because it allows to have a Pareto ordering of the individuals in the population, without creating a unique, artificial macro-objective as a linear combination of the single objectives. This feature fits perfectly with the nature of our problem. Moreover, with respect to other algorithms similarly using Pareto-based ordering, a better efficiency is obtained by NSGA-II .

As far as we know, genetic algorithms haven't been used expressly for the RL process, but some of the approaches proposed for database merging provided a useful contribution to such a usage. In [9] a genetic algorithm is used for the constrained statistical matching[3]. In this work the con-

[3]Statistical matching is a procedure used to link two files or

strained statistical matching is reduced to the balanced linear transportation problem. The function to be minimized is the sum of the squares of the distances between the fields of the matched records. This approach is similar to WBGA where each objective has equal weights. For this reason it's hard to know between two solutions with equal fitness which solution is better in which objective. Zhu and Ungar [24] propose a technique for database merging that uses the dynamic programming to compute the distance between a couple of strings. This approach has a good accuracy after 150 generations, but it needs to define, for each domain, a dictionary in order to standardize data, otherwise the accuracy decreases significantly.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented an original formulation of the record linkage problem as a multiobjective optimization problem, and we have proposed an evolutionary algorithm, EARL, for solving it. The context in which we see the major application of this algorithm is the one of peer-to-peer management systems. In its current implementation, EARL has the following advantages if used in this context:

- first, it is a *one-shot* algorithm. Specifically, record linkage is typically implemented as a process with different algorithms for each of its phases, such as a search space reduction phase which involves sorting algorithms, a phase where the appropriate comparison functions must be chosen, a decision phase with related algorithms, etc. Instead, EARL is a single algorithm that can be easily run even in complex contexts like PDMSs.

- Second, as it is based on an evolutionary paradigm, it depends just on data, that is EARL automatically adapts to data at hand with limited human knowledge required.

However, it still needs an accurate design in order to be deployed on a P2P platform. We plan to address the issues regarding such design in the future. These issues include:

- the integration of the algorithm within the query answering process;

- the design of distributed mechanisms for enacting the algorithm in a peer to peer context;

- the physical distributed implementation of the algorithm.

8. REFERENCES

[1] Thomas Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Oxford, UK, 1996.

[2] C. Batini and M.Scannapieco, *Data quality: Concepts, methods and techniques*, Series on Advanced Database Applications, Springer, 2006.

[3] Gary B. Lamont Carlos A. Coello Coello and David A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Springer,2007.

datasets where each record from one of the files is matched with a record from the second file that generally does not represent the same unit, but does represent a similar unit.

[4] K. Deb, *Multi-objective optimization using evolutionary algorithms*, Wiley-Interscience Series in Systems and Optimization, John Wiley & Sons, 2001.

[5] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*, Proc. of the Parallel Problem Solving from Nature VI Conference (Paris, France), 2000.

[6] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: Nsga-ii*, IEEE Trans. Evolutionary Computation **6** (2002), no. 2.

[7] A.K. Elmagarmid, G.I. Panagiotis, and S.V. Verykios, *Duplicate record detection: A survey*, IEEE Trans. on Knowledge and Data Engineering **19** (2007), no. 1.

[8] Ivan P. Fellegi and Alan B. Sunter, *A theory for record linkage*, Journal of the American Statistical Association **64** (1969), no. 328.

[9] Giovanni A. Flores and Eliezer A. Albacea, *A genetic algorithm for constrained statistical matching*,(2007).

[10] D.B. Fogel, *Evolutionary computation. towards a new philosophy of machine intelligence*, The Institute of Electrical and Electronic Engineers, New York, 1995.

[11] Carlos M. Fonseca and Peter J. Fleming, *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*, Genetic Algorithms: Proc. of the Fifth International Conference, 1993.

[12] M.A. Jaro, *Advances in record linkage methodologies as applied to matching the 1985 census of tampa,florida*, JASS-84, 1985.

[13] J.H.Holland, *Adaptation in natural and artificial systems*, The MIT Press, Cambridge, 1992.

[14] jmetal, http://mallba10.lcc.uma.es/wiki/index.php/JMetal.

[15] Joshua D. Knowles and David W. Corne, *Approximating the nondominated front using the pareto archived evolution strategy*, Evol. Comput. **8** (2000), no. 2, 149–172.

[16] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava, *Record linkage: similarity measures and algorithms*, SIGMOD Conference, 2006.

[17] Wai Lup Low, Mong Li Lee, and Tok Wang Ling, *A knowledge-based approach for duplicate elimination in data cleaning*, Inf. Syst. **26** (2001), no. 8, 585–606.

[18] P.Hajela and C.Y.Lin, *Genetic search strategies in multi-criterion optimal design*, Structural Optimization, 1992.

[19] Sam's String Metrics, http://www.dcs.shef.ac.uk/ sam/stringmetrics.html.

[20] J. David Schaffer, *Multiple objective optimization with vector evaluated genetic algorithms*, Proc. of the 1st International Conference on Genetic Algorithms (Mahwah, NJ, USA), 1985.

[21] N. Srinivas and Kalyanmoy Deb, *Multiobjective optimization using nondominated sorting in genetic algorithms*, Evolutionary Computation **2** (1994), no. 3.

[22] W.E. Winkler, *Using the EM Algorithm for Weight Computation in the Fellegi and Sunter Modelo of Record Linkage*, Proceedings of the Section on Survey Research Methods, American Statistical Association, 1988.

[23] W.E. Yancey, *Improving em algorithm estimates for record linkage parameters*, RESEARCH REPORT SERIES, Statistics no.2004-01.

[24] J. J. Zhu and L. H. Ungar, *String edit analysis for merging databases*, Proceedings of the KDD-2000 Workshop on Text Mining, 2000.

[25] Eckart Zitzler and Lothar Thiele, *An evolutionary algorithm for multiobjective optimization: The strength pareto approach*, Tech. Report 43, CH-8092 Zurich, Switzerland, 1998.