Optimization Routines for Enforcing One-to-One Matches in Record Linkage Problems

by Diego Moretti, Luca Valentino and Tiziana Tuoto

Abstract Record linkage aims at quickly and accurately identifying if two records represent the same real world entity. In many applications, we are interested in restricting the linkage results to "1 to 1" links, that is a single record does not appear more than once in the output. This can be dealt with the transport algorithm. The optimization problem, however, grows quadratically in the size of the input, quickly becoming untreatable for cases with a few thousand records. This paper compares different solutions, provided by some R packages for linear programming solvers. The comparison is done in terms of memory usage and execution time. The aim is to overcome the current implementation in the toolkit RELAIS, specifically developed for record linkage problems. The results highlight improvements beyond expectations. In fact the tested solutions allow successfully executing the "1 to 1" reduction for large size datasets up to the largest sample surveys at National Statistical Institutes.

Introduction

Record linkage is a process that aims at quickly and accurately identifying if two (or more) records represent the same real world entity. A record linkage project can be performed for different purposes and the variety of the uses makes it a powerful instrument to support decisions in large commercial organizations and government institutions. In official statistics, the field in which this work is developed, the combined use of statistical survey, administrative data and other new data sources (the so-called Big Data) is largely widespread and strongly stimulates the investigation of new methodologies and instruments to deal with record linkage projects.

This work is developed in the field of official statistics: in this area, the combined use of statistical surveys, administrative data and other new data sources (the so-called Big Data) is largely widespread and strongly stimulates the investigation of new methodologies and instruments to deal with record linkage projects.

Since the earliest contributions to modern record linkage (Newcombe et al., 1959; Fellegi and Sunter, 1969) there has been a proliferation of different approaches, that make use also of techniques based on data mining, machine learning, soft computing and others. Record linkage can be seen as a complex process consisting of several distinct phases involving different knowledge areas.

A record linkage process becomes trivial if the input files share a common error-free unit identifier, but it can be quite complex when common identifiers are error prone or no common identifier exists at all and one has to rely on shared covariates, as is actually the case with real data (Hernández and Stolfo, 1998).

To effectively face the record linkage problem, the Italian National Statistical Institute (Istat) designed and developed a toolkit, RELAIS, that is the result of many experiences gained performing several integration processes in different contexts (Cibella et al., 2012). This software is configured as an open source project with the aim of facing the record linkage complexity by decomposing the whole problem in its constituting phases and dynamically adopting the most appropriate technique for each step. It is therefore possible to define the most suitable strategy for each linkage problem depending on the specific data requirements (Cibella et al., 2007). Software and related documentation can be downloaded from the Istat website (ISTAT, 2015) and the European website for sharing and reusing solutions for public administrations (JOINUP, 2015)

The "Selection of unique links" is a step of the record linkage that has not been thoroughly investigated by the statistical and IT communities. This work focuses on it. Traditionally, Jaro (1989) suggested to solve it with an optimization procedure, the simplex algorithm. The solution is constrained by the size of the data processed: the optimization algorithm must solve an input matrix that grows quadratically in the input files size. In this work, we analyse the solutions already proposed and implemented and compare them with the alternatives available in the R CRAN. It is worth noting that our intention is not to provide a comparison of algorithms or optimization solutions in a general context, our goal is focused on the specific solution of the record linkage phase.

The work aims at presenting alternative algorithms and their implementations for selecting unique links in record linkage problems as they are faced in official statistics. To ensure accessibility and reuse, we consider R as the environment for running the alternative algorithms, although, due to

the algorithm complexity, R often only provides a wrapper for other more efficient programming languages. The comparison is presented throughout real life examples, derived from long experiences in data linkage at NSIs. The provided experiments are designed to account for the several features and characteristics of real world data, in terms of size, accuracy of input data and expected output. The comparison is motivated by the need for improving the current optimizer so as to process larger datasets in a short time, without losing the linkage efficacy measured in terms of precision and recall.

The work is organised as follows: in next section we shortly introduce the most common formalization for record linkage problems (Fellegi and Sunter, 1969), and specifically the optimization problem we need to solve. We describe the current module for the selection of unique links in RELAIS. Then we propose a short section about alternative algorithms. We compare the alternatives throughout several use cases, based on applications of record linkage in NSIs. Once the best implementation has been identified, further enhancements are highlighted. Finally, in the last section, we resume results and concluding remarks.

The record linkage and the optimization phase

Formalization of the probabilistic record linkage decision model

Fellegi and Sunter (1969) firstly defined record linkage as a decision problem: given two lists, say A and B, of size n_A and n_B, the linkage process can be viewed as a classification problem where the pairs in the Cartesian product $\Omega = ((a,b), a \in A \quad \text{and } b \in B)$ have to be assigned into two subsets M and U, independent and mutually exclusive, such that M is the set of Matches $(a = b)$ while U is the set of Non-Matches $(a \neq b)$.

In order to assign the pairs (a,b) either to the set M or U, k common attributes (the matching variables) are compared. The statistical model for record linkage is built upon the so called comparison vectors γ

$$\gamma_{(ab)} = \left(\gamma_{(ab)1}, \ldots, \gamma_{(ab)k}\right)$$

where, in the simplest setting,

$$\gamma_{(ab)j} = \begin{cases} 1 & \gamma_{(ab)j} = \gamma_{(ab)j} \\ 0 & \gamma_{(ab)j} \neq \gamma_{(ab)j} \end{cases}, \qquad j = 1, \ldots, k.$$

The comparison vectors $\gamma_{(ab)}$ are usually assumed to be independent and identically distributed random vectors with distribution given by a mixture of two different (unobserved) distributions: the former represents the pairs (a,b) which actually are the same unit, the m distribution; the latter represents the pairs (a,b) which actually belong to different units, the u distribution. The mixture weight p represents the marginal probability that a random pair of records (a,b) is a true match, i.e. it may be interpreted as the percentage of overlapping of the two data sets.

The estimation of the mixture weight p and the two distributions m and u requires the use of iterative methods, generally the EM algorithm or its generalizations. Also, in the standard setting, the matching variables are assumed independent of each other. Several extensions of this basic set-up have been proposed, mainly by introducing potential interactions among key variables, see for example (Winkler, 1995; Larsen and Rubin, 2001).

Once the two distributions $m(\gamma_{(ab)})$ and $u(\gamma_{(ab)})$ are estimated, a given pair should be allocated to M or U on the basis of the likelihood ratio, also called composite matching weight:

$$r_{(ab)} = \frac{\hat{m}(\gamma_{(ab)})}{\hat{u}(\gamma_{(ab)})}.$$

It is also possible to assign the pairs on the basis of a posterior probability that the pair is a match:

$$r^*_{(ab)} = \frac{\hat{p} * \hat{m}(\gamma_{(ab)})}{\hat{p} * \hat{m}(\gamma_{(ab)}) + (1 - \hat{p})\, \hat{u}(\gamma_{(ab)})}.$$

In general, we declare as matches the pairs of records with likelihood ratio r - or posterior probability r^* - above a fixed threshold. In practice, the choice of the threshold can be problematic, as illustrated, for example, in Belin and Rubin (1995). In this context, optimization techniques may be helpful to solve the multiple matches issue, that is the possibility that a single unit in data set A is linked with more than one unit in data set B. This will be discussed in the next subsection.

The optimization problem in record linkage

In several applications, the record linkage aims at recognizing exactly and univocally the same units and to establish only unique or "1 to 1" links. In other words, the linkage result must satisfy the constraint that one record on file A can be assigned to one and only one record on file B, and vice-versa. This kind of applications requires several constraints and is a complex problem of optimization. For instance, when comparing Population Census with Post Enumeration Survey, one is interested in "1 to 1" links in order to recognize people caught in the two occasions; moreover when linking tax register and income survey again "1 to 1" links are the expected output in order to enrich the available information in the input sources. On the other side, when hospital admissions forms are linked to a patient list, multiple linkages ("n to 1" links) are admissible. Finally "n to m" links are expected when linking, for instance, Employees and Employers files.

To achieve "1 to 1" links in the Fellegi-Sunter setting that considers the cross product of possible pairs, Jaro (1989) suggested to formulate it as a linear programming problem: once the matching weight r is assigned to each pair, the identification of "1 to 1" links can be solved maximizing the objective function given by the sum of weights for the link pairs, under the constraints given by the fact that each unit of A can be linked at most with one unit of B and vice-versa. According to Jaro (1989), this is a degenerate transportation problem, and the use of such a linear programming model represents an advance with respect to other ad hoc assignment methods. In order to formulate the problem, let r_{ab} be the matrix containing the composite weights for all pairs, the maximizing function is:

$$Z = \sum_{a=1}^{n_A} \sum_{b=1}^{n_B} r_{ab} X_{ab}$$

under the $n_A + n_B$ constrains:

$$\sum_{a=1}^{n_A} X_{ab} \leqslant 1 \qquad b = 1, 2, \ldots, n_B$$

$$\sum_{b=1}^{n_B} X_{ab} \leqslant 1 \qquad a = 1, 2, \ldots, n_A$$

where X_{ab} is a matrix with entries corresponding to indicator variables, equal to 1 if record a in A is linked with record b in B .

It is worth noting that the size of the X_{ab} matrix increases quadratically in the size of the input files, with dramatic effects on the memory usage and computation time. For instance, when both input files contain 100 records, the X matrix contains 10 thousand cells; when both input files consist of 1000 records, the matrix becomes 1 million entries. Traditionally, in record linkage problems, the computation issues related to the input size are managed via the so called blocking procedures (Gu et al., 2003; Baxter et al., 2003); in the optimization step, the complexity related to the input size is also managed restricting to the "most likely" pairs. Indeed, the matching weight r represents the ratio between the likelihood that a pair belongs to the set of Matches and the likelihood that the pair belongs to the set of Non-Matches. Similarly, r^* represents the posterior probability that a pair belongs to the set of Matches. It is clear that for most pairs, the matching weights r_{ab} and r^*_{ab} take very small values (close to zero), since considering two input files of 1000 records and the 1 million pairs they generate, at most 1000 pairs will be true matches with expected high values of r_{ab} and r^*_{ab}. So, a common practice for solving the "1 to 1" links is to reduce the complexity by eliminating from the optimization analysis the pairs with a value of r_{ab} (similarly, r^*_{ab}) below a certain threshold. A common choice of the threshold for r_{ab} is 1, meaning that we disregard the pairs for which the likelihood of belonging to M is lower than the likelihood of belonging to U. For r^*_{ab}, the most proper choice seems 0.5, as it is a posterior probability. The role of r_{ab} and r^*_{ab} will be further discussed in the experimental section.

It is worth mentioning that in the Bayesian approach to record linkage, the "1 to 1" constraint is solved directly in the model rather than in an ex-post adjustment (Tancredi and Liseo, 2013; Stoerts et al., 2017; Sadinle, 2017); however, to the best of our knowledge, the Bayesian record linkage is still affected by a certain lack of scalability, so we do not consider it in this analysis.

The Relais toolkit

The abovementioned Relais is a toolkit developed and used in Istat and in other NSIs to face record linkage problems. It is implemented in two programming languages, Java and R; moreover, the relational database MySql is used to manage datasets and temporary results. The R language is used for the key statistical phases of Relais: the estimation of the parameters p, m, and u of the probabilistic decision model, based on Fellegi-Sunter approach (Fellegi and Sunter, 1969), and the optimization algorithm to obtain the "1 to 1" linkage results.

The "1 to 1" reduction in Relais

The "1 to 1" reduction in RELAIS uses the linear programming problem approach proposed by Jaro (1989) and defined in the previous section. The R module uses the **LpSolve** package. In the following, the core of the current R source:

```
# pairs is the output of the decision model
# colnames(pairs) <- c("a", "b", "gamma", "r", "r*")


# command1: application of a preliminary filter to the input data
filtered=pairs[pairs[,5]>0.5,]

# command2: input preprocessing
# counting of unique identifiers of records
nA= length(unique(filtered[,1]))
nB= length(unique(filtered[,2]))
A=cbind(a=unique(filtered[,1]),A=1:nA)
B=cbind(b=unique(filtered[,2]),B=1:nB)
filtered =merge(B, filtered)
filtered =merge(A, filtered)
dat=t(filtered)

# command3: preparing constraint matrix
constr=array(rep(0,(nA+nB)*ncol(dat)), dim=c(nA+nB,ncol(dat)))
p=rbind(matrix(rep(dat[2,],nA),c(nA,ncol(constr)),byrow=TRUE),
        matrix(rep(as.numeric(dat[4,])+nA,nB),c(nB,ncol(constr)),byrow=TRUE))
constr [as.numeric(p)==row(constr)]=1

# command4: preparing other LP parameters
diseq=rep('<=',nA+nB)
ones=rep(1,nA+nB)
# target function
coeff=dat[6,]

# command5: LP execution
library("lpSolve")
ret=lp ("max", coeff, constr, diseq, ones)
# preparing the reduced set of pairs
reduc <- t(dat[,ret$solution>0.9])
```

The command1 is the preliminary filter useful to reduce the size of the input pairs: each pair is an entry of the constraint matrix. The filter is $r^* \geqslant 0.5$, as explained at the end of the previous section. Despite this simplification, the method may not work when processing medium/large sized datasets, i.e. when the input files A and B consist of more than 5000 records. In this case, the Relais tool offers an alternative algorithm (the greedy one) which, however, does not guarantee an optimal result. The size of the datasets that are treatable with the procedure depends on several causes. First of all, it depends on the workstation system 32-bit o 64-bit for Windows platform. Other relevant parameters are the size of the RAM, the version of the Operating System, the version of R, others software running on the workstation, the processor, etc. As shown in the next paragraph, we investigated the limits of this algorithm in two typical PC configurations (32-bit and 64-bit), furthermore we propose some improvements aimed at increasing its efficiency and performances.

Algorithms and R packages for LP solver

A recent analysis and comparison of different algorithms for linear programming is in Gearhart et al. (2013). This work is an inspiring starting point for our analysis aimed at investigating the best freely available open-source implementation in terms of performance and memory usage. Gearhart et al. (2013) compare four different open-source solvers facing a collection of linear programming problems; Gearhart et al. (2013) also consider the IBM ILOG CPLEX Optimizer (CPLEX), an industrial standard.

In this work, we compare linear programming solvers with the specific purpose of optimizing the identification of "1 to 1" links: to this end, we compare the current implementation available in Relais, the **lpSolve** (Berkelaar et al., 2013) R package, with the **Rglpk** (Theussl and Hornik, 2014) and

ROI.plugin.clp (Thieurmel, 2017) R packages. These two R packages are wrappers of C libraries, corresponding to the two methodologies, GLPK and COIN-OR respectively. The comparison of these packages in other contexts, with different optimization problems, is out of the scope of this work. At a very first stage, we also considered the **intpoint** (del Rio, 2012) R package, that has been discarded because of the low performance in memory management compared to the previous ones. We also developed a Java procedure that implements the Hungarian Method Karmarkar (Karmarkar, 1984) but it was discarded because it did not bring improvements.

The comparison of the proposed solvers is influenced by the several configuration parameters of the personal computers, the specific hand code for the input preparation, and other characteristics that have been fixed in the experimental settings, described in the following section.

Experiments

In this paragraph, we resume the experiments about the "1 to 1" reduction procedures with the aim of measuring their execution time and their ability to handle large size data. Moreover, some upgrades of the code are proposed and described in detail: the upgrades are evaluated by comparing their performances with the current version. As previously specified, the current Relais procedure successfully solves optimization problems when the input files are smaller than 5000 records each. We investigate improvements with the first objective of enlarging these sizes; in the set of solutions which enable to manage larger datasets, we evaluate the best performances in terms of execution time. As already mentioned, Relais also proposes another method, the greedy one, but this is not compared with the optimization algorithms because it follows a different rationale not aimed at global optimization of the result. In addition, currently in Relais the greedy algorithm is not implemented as an R module. In short, we have observed that when the greedy algorithm finds links other than those of the optimal algorithms, these links are not correct; however, this rarely happens, about 1 in 1000 proposed pairs.

Experiment setup

For the comparison of the algorithms, we used two typical PC configurations, the 32-bit and 64-bit R respectively. Details on the PC configurations are shown in table 1.

Config.	OS	System	Processor	RAM	R version
32-bit	Windows 7	32-bit	Pentium Dual-Core 2.7 Ghz	4 Gb	3.4.0
64-bit	Windows 7	64-bit	Intel 3.5 Ghz	16 Gb	3.4.2

Table 1: Experiment PC configurations

To evaluate the performances in terms of time and memory usage, we used ten different linkage exercises summarized in table 2. Each exercise is composed by two datasets to integrate, the entities object of the linkage can be people or companies. The first seven problems are quite standard, i.e. the size of the two datasets and the number the matches are balanced; on the other hand, the last three exercises are less common, i.e. the size of datasets are lopsided or there are few matches. In the first column of table (Exercise), we mark the exercise on the basis of the size of the datasets, with "L" as suffix for lopsided exercises; this mark is also used in the next tables. The datasets reproduce real data; the linking variables are the true ones either have been artificially generated mimicking true data, as reported in the second column. The size of datasets and the number of real matches vary across the exercises, as shown in the last two columns of table 2.

Experiments report

The current procedure encounters two critical phases, in which the memory used risks exceeding the memory available for the R task, respectively commands 3 (preparing constraint matrix) and 5 (LP execution). In both cases, the issue is represented by the size of the constraint matrix X_{ab}. As mentioned before, when the sizes of the input files do not allow the execution of the R commands, currently in Relais the users are suggested to apply greedy techniques.

So, firstly we focused on modifying command 3 to overcome the memory problem. The most promising solution is to reformulate the constraint matrix as a vector of constraints, where only the non-zero values of matrix X_{ab} appear. This structure requires much less memory than the previous solution, especially when the size of the inputs increases. The `lp` function of the **lpSolve** package

Exercise	Linking variables	Entity	Datasets size	True matches
1K	Real	People	1,165x1,165	1,141
4K	Artificial	Companies	4,012x3,988	3,187
5K	Real	People	5,851x5,748	5,543
8K	Real	People	8,518x7,596	6,893
15K	Artificial	Companies	14,998x15,003	11,281
25K	Real	People	25,343x24,613	24,043
40K	Artificial	Companies	40,048x39,952	36,033
10KL	Artificial	Companies	9,963x1,018	1,015
20KL	Artificial	Companies	658x20,052	625
55KL	Artificial	Companies	55,210x5,083	2,189

Table 2: Experiment datasets

admits the `dense.const` parameter which allows us to use a vector instead of a matrix to express the constraints for our maximization problems. In this case commands 3 and 5 become as follows:

```
# command3.1: preparing constraint vector
constr.vec <- matrix(c(as.numeric(dat[2,]), as.numeric(dat[4,])+nA,
                       rep(1:ncol(dat),2), rep(1,(2*ncol(dat)))), ncol=3)


# command5.1: LP execution
library("lpSolve")
ret=lp ("max", coeff, , diseq, ones, dense.const=constr.vec)
# preparing the reduced set of pairs
reduc <- t(dat[,ret$solution>0.9])
```

Table 3 shows the results of the 10 exercises, in the two options. The column 'Matrix' shows the execution time of the current R module with constraints expressed by a matrix. The column 'Vector' shows the execution time of the R module with commands 3.1 and 5.1 in place of commands 3 and 5, i.e. constraints expressed by a vector. The 'KO' entry means that the execution is aborted due to memory error.

Config.	RAM	Exercise	Matrix	Vector
32-bit	4 Gb	1K	4.3	0.5
32-bit	4 Gb	4K	28.0	2.0
32-bit	4 Gb	5K	KO	5.7
32-bit	4 Gb	8K	KO	9.0
32-bit	4 Gb	15K	KO	32.9
32-bit	4 Gb	25K	KO	82.5
32-bit	4 Gb	40K	KO	213.3
32-bit	4 Gb	10KL	0.7	0.7
32-bit	4 Gb	20KL	1.9	0.8
32-bit	4 Gb	55KL	3.7	1.9
64-bit	16 Gb	1K	2.2	0.3
64-bit	16 Gb	4K	14.6	1.0
64-bit	16 Gb	5K	45.3	2.7
64-bit	16 Gb	8K	69.1	3.8
64-bit	16 Gb	15K	206.2	11.6
64-bit	16 Gb	25K	KO	27.6
64-bit	16 Gb	40K	KO	77.7
64-bit	16 Gb	10KL	0.2	0.2
64-bit	16 Gb	20KL	0.7	0.4
64-bit	16 Gb	55KL	1.4	0.8

Table 3: Use of the constraint matrix against the constraint vector

The results shown by table 3 fully meet our expectations. In fact, the current code can process only 1K, 4K and lopsided datasets using the 32-bit configuration and reaches up to 15K datasets in

the 64-bit configuration. The new code can process all datasets even in the worst configuration. In addition, there is also a great improvement in the execution times.

In a second phase, we concentrated our efforts on evaluating the use of the alternative packages identified and abovementioned, i.e. **ROI.plugin.clp** and **Rglpk**.

The two solvers accept the constraint parameter as vector, using the structure `simple_triplet_matrix` defined in the **slam** (Hornik et al., 2014) package.

Then, command 3 becomes

```
# command3.2: preparing constraint parameter
constr <- simple_triplet_matrix(c(as.numeric(dat[2,]),as.numeric(dat[4,])+n),
               rep(1:ncol(dat),2), rep(1,(2*ncol(dat))), nrow=(n+m), ncol=ncol(dat))
```

In the case of **ROI.plugin.clp** solver, command 5 becomes:

```
# command5.2: LP execution
LP <- ROI::OP(as.numeric(coeff), ROI::L_constraint(L = constr, dir = diseq, rhs = ones),
                max = TRUE)
ret <- ROI::ROI_solve(x = LP, solver = "clp")
# preparing the reduced set of pairs
reduc <- t(dat[,ret$solution>0.9])
```

In the case of **Rglpk** solver, command 5 is:

```
# command5.3: LP execution
ret <- Rglpk_solve_LP(coeff,constrv,diseq,ones,types="I",max=TRUE)
# preparing the reduced set of pairs
reduc <- t(dat[,ret$solution>0.9])
```

Table 4 compares the execution times in seconds, required for the complete execution of the R module using the three different packages:

Config.	RAM	Exerc.	lpSolve	ROI.plugin.clp	Rglpk
32-bit	4 Gb	1K	0.5	0.1	0.4
32-bit	4 Gb	4K	2.0	0.2	1.0
32-bit	4 Gb	5K	5.7	0.3	2.7
32-bit	4 Gb	8K	9.0	0.3	3.5
32-bit	4 Gb	15K	32.9	0.6	9.6
32-bit	4 Gb	25K	82.5	0.7	27.7
32-bit	4 Gb	40K	213.3	1.3	73.2
32-bit	4 Gb	10KL	0.7	0.2	0.2
32-bit	4 Gb	20KL	0.8	0.2	0.3
32-bit	4 Gb	55KL	1.9	0.2	0.8
64-bit	16 Gb	1K	0.3	0.2	0.1
64-bit	16 Gb	4K	1.0	0.2	0.4
64-bit	16 Gb	5K	2.7	0.2	1.1
64-bit	16 Gb	8K	3.8	0.2	1.5
64-bit	16 Gb	15K	11.6	0.3	4.1
64-bit	16 Gb	25K	27.6	0.3	9.8
64-bit	16 Gb	40K	77.7	0.4	22.6
64-bit	16 Gb	10KL	0.2	0.1	0.1
64-bit	16 Gb	20KL	0.4	0.1	0.2
64-bit	16 Gb	55KL	0.8	0.2	0.3

Table 4: Reduction procedure using the three different packages

From table 4, the first remark underlines that the use of constraints as vectors allows all packages to manage all the tested datasets even in the worst configuration. Moreover, for this type of problem, it is quite clear that the **ROI.plugin.clp** solver guarantees the best performances, overtaking the other packages especially with large datasets. The gain with **ROI.plugin.clp** is more evident in the 32-bit configuration. The second best performer is **Rglpk**, however it is at least one order of magnitude slower than the previous one. Finally, **lpSolve** presents the worst performances, particularly in the 32-bit configuration, whilst the differences with **Rglpk** are reduced in the case of the 64-bit configuration.

These results are substantially valid for both balanced data sets and lopsided data sets. We note that in lopsided cases all solutions seem to perform better. In fact, the complexity of the problem is mainly due to the number of pairs proposed to the reduction algorithm rather than to the size of the input data. In typical "1 to 1" record linkage projects, the number of pairs depends more on the size of the smallest dataset than on the largest one. The **Roi.plugin.clp** greatly improves in the case of sparse matrices; the difference with **Rglpk** is reduced with lopsided data. In our opinion, a large part of the improvements with **Roi.plugin.clp** is due to the COIN-OR Optimization algorithm written in C and the use of a good wrapper for R; there is a part of the code where the R language communicates with a buffer with a procedure compiled in C language. **Rglpk** is also based on an algorithm written in C, but this is probably less powerful in this type of problem. Instead, **lpSolve** is written entirely in R, it makes an intensive use of the memory, and it is therefore less efficient than the other tested packages.

Further improvements and concluding remarks

The vector representation of the constraints and the adoption of the **ROI.plugin.clp** package provide the expected important improvements in the module's performance, i.e. the size of the managed pairs, memory usage and the execution speed. The achieved results encourage us to overcome the current preliminary filter that allows processing only pairs (ab) with posterior probability $r^*_{ab} > 0.5$, see command 1. This filter was included to overcome the previous performance issue in terms of memory usage, however, from a statistical perspective, it risks compromising the results of the statistical model by deterministically removing possible matches. Obviously, the more restrictive the filter, the more likely it is that possible links are missing. So, to partially reduce this drawback, we intend to apply the filter $r_{ab} > 1$ instead of $r^*_{ab} > 0.5$. In fact, $r_{ab} > 1$ is less restrictive than $r^*_{ab} > 0.5$. The filters $r_{ab} > 1$ and $r^*_{ab} > 0.5$ are "theoretically" justified, as they represent, respectively, the likelihood ratio and the posterior probability of a pair to be a match.

We resume the 10 exercises proposed in the previous section and apply the "1 to 1" reduction with the proposed preliminary filters. In each run we measure the execution time and the quality of the output in terms of recall. The recall is a standard quality measure used in record linkage that compares the number of true matches identified by the linkage process with the number of true matches. The closer the value is to 1, the more effective the procedure is. The following table 5 compares the obtained measures. We only report the values obtained from the **ROI.plugin.clp** package in the 32-bit configuration. As above clarified, our first interest is to evaluate the improvements in the recall and to verify that memory problems do not recur.

Exercise	Filter $r^*_{ab} > 0.5$		Filter $r_{ab} > 1$	
	Time	Recall	Time	Recall
1K	0.1	0.985	0.1	0.997
4K	0.2	0.943	0.2	0.976
5K	0.3	0.966	0.3	0.971
8K	0.3	0.944	0.3	0.950
15K	0.6	0.927	0.7	0.980
25K	0.7	0.698	0.7	0.710
40K	1.3	0.688	7.3	0.945
10KL	0.2	0.956	0.3	0.983
20KL	0.2	0.944	0.5	0.958
55KL	0.2	0.888	0.6	0.938

Table 5: Recall measure using different filters

Table 5 firstly shows that, with the filter $r_{ab} > 1$, there is always a small improvement in the recall. Secondly, the algorithm is always successfully executable and the execution time doesn't generally increase significantly. The exercise 40K is an exception: the execution time goes from 1.7 to 7.3 seconds. In this case, we have verified that the modification of the preliminary filter has a significant impact: in fact, the number of considered pairs increases from about 26,000 to over 100,000, with remarkable effects on the recall. The 40K experiment proves, on one hand, that this adjustment can have important positive effects for the linkage process and, on the other hand, ensures that the procedure can be successfully executed also with a large number of input pairs.

```
# command1: application of a preliminary filter to the input data
filtered=pairs[pairs[,6]>1,]
```

```
# command2: input preprocessing
# counting of unique identifiers of records
n= length(unique(filtered[,1]))
m= length(unique(filtered[,2]))
A=cbind(I=unique(filtered[,1]),A=1:n)
B=cbind(J=unique(filtered[,2]),B=1:m)
filtered =merge(B, filtered)
filtered =merge(A, filtered)
dat=t(filtered)

# command3: preparing constraint parameter
constr <- simple_triplet_matrix(c(as.numeric(dat[2,]),as.numeric(dat[4,])+n), rep(1:ncol(dat),2),
                                rep(1,(2*ncol(dat))), nrow=(n+m), ncol=ncol(dat))

# command4: preparing other LP parameters
diseq=rep('<=',m+n)
ones=rep(1,m+n)
# coefficients for the target function
coeff=dat[6,]

# command5: LP execution
LP <- ROI::OP(as.numeric(coeff),
                        ROI::L_constraint(L = constr,  dir = diseq,  rhs = ones), max = TRUE)
ret <- ROI::ROI_solve(x = LP, solver = "clp")
# prepare the reduced set of pairs
reduc <- t(dat[,ret$solution>0.9])
```

To conclude, the main advantages of the proposed improvements relate to the successful execution of the "1 to 1" reduction for large datasets, as well as the gain in the execution time. The above reported new code for the implementation of the optimization step will replace the previous one in the new release of Relais. We are mostly satisfied with the achieved improvements, as they will simplify and enhance the future linkage strategies. For instance, with the improvements studied in this paper, we will be able to easily manage the linkage between statistical registers and social sample surveys. In particular, we will be able to manage the "1 to 1" optimization step for the current social surveys involving about 40,000 units, as in the experiment 40K. Moreover, we will also manage within its main spatial domains, the largest Italian sample survey, the Labour Force Survey (LFS). In fact, currently, the Italian LFS involves up to 25,000 units, in the NUTS2 domain.

Bibliography

R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. *ACM SIGKDD*, 3:25–27, 2003. [p3]

T. R. Belin and D. B. Rubin. A method for calibrating false- match rates in record linkage. *Journal of the American Statistical Association*, (90):694–707, 1995. [p2]

M. Berkelaar et al. *lpSolve: Interface to Lpsolve v. 5.5 to Solve Linear Integer Programs*, 2013. URL http://CRAN.R-project.org/package=lpSolve. R package version 5.6.7. [p4]

N. Cibella, M. Fortini, R. Spina, M. Scannapieco, L. Tosco, and T. Tuoto. Relais: An open source toolkit for record linkage. *Rivista di Statistica Ufficiale*, (2-3):55–68, 2007. [p1]

N. Cibella, M. Scannapieco, L. Tosco, T. Tuoto, and L. Valentino. Record linkage with relais: Experiences and challenges. *Estadística española*, 54(179):311–328, 2012. [p1]

A. Q. del Rio. *Intpoint: Linear Programming Solver by the Interior Point Method and Graphically (Two Dimensions)*, 2012. URL http://CRAN.R-project.org/package=intpoint. R package version 1.0. [p5]

I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, (64): 1183–1210, 1969. [p1, 2, 3]

J. L. Gearhart, K. L. Adair, J. D. Detry Richard J.and Durfee, K. A. Jones, and N. Martin. Comparison of open-source linear programming solvers. *Sandia National Laboratories Report*, pages 4–62, 2013. [p4]

L. Gu, R. Baxter, D. Vickers, and C. Rainsford. Record linkage: Current practice and future directions. *CSIRO Mathematical and Information Sciences Technical Report*, 3(83), 2003. [p3]

M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge purge problem. *Journal Data Mining and Knowledge Discovery*, 2(1):9–37, 1998. [p1]

K. Hornik, D. Meyer, and C. Buchta. *Slam: Sparse Lightweight Arrays and Matrices*, 2014. URL http://CRAN.R-project.org/package=slam. R package version 0.1-32. [p7]

ISTAT. Download RELAIS on Istat Website, 2015. URL https://www.istat.it/en/methods-and-tools/methods-and-it-tools/process/processing-tools/relais. [p1]

M. A. Jaro. Advances in record linkage methodologies as applied to matching the 1985 census of tampa, florida. *Journal of American Statistical Society*, 84(84):414–420, 1989. [p1, 3, 4]

JOINUP. Download RELAIS on Joinup Website, 2015. URL https://joinup.ec.europa.eu/solution/relais-record-linkage-istat. [p1]

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. [p5]

M. D. Larsen and D. B. Rubin. Iterative automated record linkage using mixture models. *Journal of the American Statistical Association*, (79):32–41, 2001. [p2]

H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130 (130):954–959, 1959. [p1]

M. Sadinle. Bayesian estimation of bipartite matchings for record linkage. *Journal of the American Statistical Association*, 112(518):600–612, 2017. [p3]

R. Stoerts, R. Hall, and S. Fienberg. A bayesian approach to graphical record linkage and de-duplication. *Journal of the American Statistical Association*, pages 1660–1672, 2017. [p3]

A. Tancredi and B. Liseo. A hierarchical bayesian approach to record linkage and population size problems. *The Annals of Applied Statistics*, pages 1553–1585, 2013. [p3]

S. Theussl and K. Hornik. *Rglpk: R/GNU Linear Programming Kit Interface*, 2014. URL http://CRAN.R-project.org/package=Rglpk. R package version 0.6-0. [p4]

B. Thieurmel. *ROI.plugin.clp: 'Clp (Coin-or Linear Programming)' Plugin for the 'R' Optimization Interface*, 2017. URL https://CRAN.R-project.org/package=ROI.plugin.clp. R package version 0.4. [p5]

W. E. Winkler. Matching and record linkage. *Business Survey Methods (B.G. Cox et al, ed.)*, pages 355–384, 1995. [p2]

Diego Moretti
Istat - Italian National Institute of Statistics
via C. Balbo 16
Italy
dimorett@istat.it

Luca Valentino
Istat - Italian National Institute of Statistics
via C. Balbo 16
Italy
luvalent@istat.it

Tiziana Tuoto
Istat - Italian National Institute of Statistics
via C. Balbo 16
Italy
tuoto@istat.it