

**EVOLUTIONARY APPROACHES TO DATA
INTEGRATION RELATED PROBLEMS**

MOISÉS GOMES DE CARVALHO
ADVISOR: ALBERTO HENRIQUE FRADE LAENDER

**EVOLUTIONARY APPROACHES TO DATA
INTEGRATION RELATED PROBLEMS**

Thesis presented to the Graduate Program
in Computer Science of the Federal University
of Minas Gerais in partial fulfillment of
the requirements for the degree of Doctor
in Computer Science.

Belo Horizonte

October 2009

Carvalho, Moisés Gomes de.

C331a Evolutionary Approaches to Data Integration
Related Problems. / Moisés Gomes
de Carvalho. — Belo Horizonte, 2009.
xxiv, 111 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas
Gerais. Departamento de Ciência da Computação.

Advisor: Prof. Alberto Henrique Frade Laender.

1. Genetic Programming - Teses. 2. Data Integration -
Teses. 3. Record Deduplication - Teses.
I. Orientador. II Título.

CDU 519.6*73(043)

MOISÉS GOMES DE CARVALHO
ORIENTADOR: ALBERTO HENRIQUE FRADE LAENDER

**ABORDAGENS EVOLUCIONÁRIAS PARA
PROBLEMAS RELACIONADOS A INTEGRAÇÃO
DE DADOS**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Belo Horizonte
Outubro de 2009

Carvalho, Moisés Gomes de.

C331a Abordagens evolucionárias para problemas relacionados a integração de dados. / Moisés Gomes de Carvalho. — Belo Horizonte, 2009.

xxiv, 111 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Prof. Alberto Henrique Frade Laender.

1. Programação genética - Teses. 2. Integração de dados - Teses. 3. Deduplicação de registros - Teses.
I. Orientador. II Título.

CDU 519.6*73(043)



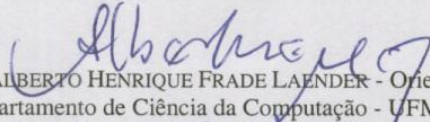
UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

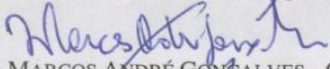
FOLHA DE APROVAÇÃO

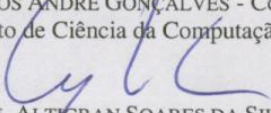
Abordagens evolucionárias para problemas relacionados a integração de dados

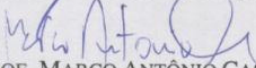
MOISÉS GOMES DE CARVALHO

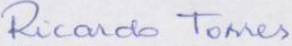
Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

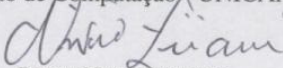

PROF. ALBERTO HENRIQUE FRAIDE LAENDER - Orientador
Departamento de Ciência da Computação - UFMG

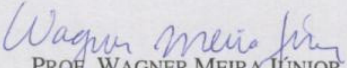

PROF. MARCOS ANDRÉ GONÇALVES - Co-orientador
Departamento de Ciência da Computação - UFMG


PROF. ALTIGRAN SOARES DA SILVA
Departamento de Ciência da Computação - UFAM


PROF. MARCO ANTÔNIO CASANOVA
Departamento de Informática - PUC - RJ


PROF. RICARDO DA SILVA TORRES
Instituto de Computação - UNICAMP


PROF. NÍVIO ZIVIANI
Departamento de Ciência da Computação - UFMG


PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 26 de outubro de 2009.

Aos meus pais.

Acknowledgments

Aos meus pais Aparecida e Francisco por todo o suporte, amizade e amor incondicionais.

Aos meus irmãos Felipe e Alice pelo carinho, incentivo e torcida.

Ao meu orientador, professor Alberto Laender, por sua paciência para comigo e por sua disposição durante todos os anos que trabalhamos juntos enquanto estive na UFMG.

Ao Marcos André e Altigran Soares pela paciência, pelos puxões de orelha e pela confiança depositada.

À Renata, Sheila, Sônia, Cida, Rosencler e todos da secretaria da pós-graduação pelo trabalho fantástico e por toda atenção e ajuda que recebi.

Aos meus colegas e amigos do LBD, Latin e Speed, entre os quais Christiano (o Gigante), Bárbara, Allan Jones (o Audaz!!), Rodrygo, Thiago, Gabriel, Anderson, Guilherme, Luana, Luiz Guilherme, Evandrino, Karla, Waister, Rafael, Marlos, Fabiano, Adriano, Leonardo, Coutinho, e Humberto pela amizade, coleguismo e companheirismo durante todos os anos que pudemos estar no mesmo barco e vivendo as mesmas experiências.

Ao Horácio e David, que além de trilharem o doutorado comigo, foram como irmãos que ganhei durante essa fase um tanto diferente da vida.

Ao amigos que fiz no vôo livre na Serra da Moeda, em especial o Patolino, os Morcegos Voadores e a Elisa Vignolo, que marcaram de forma muito especial suas presenças em minha vida.

Aos meus colegas e professores da UFAM, pela torcida e incentivo durante todo o

percurso.

Ao Haylo Neto, Daniel Figueiredo, Rodrigo Reis e ao Antenor Pessoa, da Wicked Garden, pelo companherismo e amizade de tantos anos e, claro, por estarem sempre prontos para tocar algo quando das minhas visitas à Manaus.

Ao Wladimir, ao Jan Val Ellam, ao Planeta X, ao Xenu e a todos os cabeções que vêm e vão, por terem deixado o peso do doutorado um pouco mais leve.

Novamente, obrigado a todos que citei e também aos que por ventura não tenha citado (provavelmente precisaria escreveria um outro volume para fazer justiça) por toda a ajuda que recebi, nas mais diferente formas, durate o processo de transformação que o doutorado exigiu. Sem todos e cada um de vocês, imagino que viver tudo isso não teria passado de uma experiência vazia e sem sentido.

Um grande abraço.

“A compreensão humana não é um exame desinteressado, mas recebe infusões da vontade e dos afetos; disso se originam ciências que podem ser chamadas “ciências conforme a nossa vontade”. Pois um homem acredita mais facilmente no que gostaria que fosse verdade. Assim, ele rejeita coisas difíceis pela impaciência de pesquisar; coisas sensatas, porque diminuem a esperança; as coisas mais profundas da natureza, por superstição; a luz da experiência por arrogância e orgulho; coisas que não são comumente aceitas, por deferência à opinião do vulgo. Em suma, inúmeras são as maneiras, e às vezes imperceptíveis, pelas quais os afetos colorem e contaminam o entendimento.”

(Francis Bacon, Novum organon(1620))

Resumo

Integração de dados tem como objetivo combinar dados de diferentes fontes (repositórios de dados tais como bibliotecas digitais e bancos de dados) por meio da adoção de um modelo de dados global e da detecção e resolução de problemas de conflito de esquemas e entre os dados armazenados, de modo a prover uma percepção/visão unificada ao usuário. Dois problemas específicos relacionados ao processo de integração de dados - deduplicação de registros e pareamento de esquemas - apresentam espaços de soluções muito vastos. Por esse motivo, explorar esses espaços da forma tradicional torna-se uma alternativa computacionalmente cara e tecnicamente inviável para se encontrar soluções. Além disso, as soluções para estes problemas exigem que objetivos múltiplos (e às vezes conflitantes) sejam atendidos simultaneamente. O objetivo desta tese é apresentar abordagens evolucionárias, como a programação genética, como ferramentas para solucionar tais problemas, levando a novas abordagens e métodos capazes de atender a todas essas exigências e ao mesmo tempo, prover soluções de alta eficiência e eficácia.

O primeiro trabalho apresentado nesta tese propõe uma abordagem, baseada em programação genética, para deduplicação de registros. Essa abordagem combina diferentes evidências extraídas dos dados armazenados para sugerir funções de deduplicação capazes de identificar quando dois registros são réplicas ou não. Como demonstrado pelos experimentos realizados, nossa abordagem consegue superar métodos na literatura até então considerados como o estado-da-arte. Além disso, as funções de deduplicação sugeridas são eficientes, exigindo menos processamento, pois utilizam menos evidên-

cias. Finalmente, essa abordagem evolucionária é capaz de adaptar automaticamente as funções de deduplicação a qualquer valor de limiar de identificação de réplicas, poupando o usuário do trabalho de escolher e ajustar o valor desse parâmetro.

A partir dos resultados obtidos pela abordagem anterior, também é proposta uma abordagem evolucionária para o problema de encontrar casamentos entre elementos de esquemas de repositórios de dados semanticamente relacionados (problema de pareamento de esquemas). O objetivo do nosso trabalho foi desenvolver uma abordagem capaz de encontrar casamentos de esquemas em uma situação adversa na qual informações sobre a estrutura do repositório não estão disponíveis. Esta abordagem é pioneira na tarefa de encontrar casamentos complexos usando somente os dados armazenados nos repositórios. Para encontrar casamentos complexos são utilizadas estratégias de busca, baseadas em técnicas de deduplicação de registros e de recuperação de informação, durante o processo evolucionário. Para demonstrar a eficácia de nossa abordagem, conduzimos uma avaliação experimental usando conjuntos de dados reais e sintéticos. Os resultados demonstram que a abordagem proposta é capaz de identificar casamentos complexos com grande precisão, apesar de fazer uso somente dos dados armazenados.

Abstract

Data integration aims to combine data from different sources (data repositories such as databases, digital libraries, etc.) by adopting a global data model and by detecting and resolving schema and data conflicts so that a homogeneous, unified view can be provided. Two specific problems related to data integration - schema matching and replica identification - present a large solution space. This space is computationally expensive and technically prohibitive to be intensively and exhaustively explored by traditional approaches. Moreover, the solutions for these problems usually require that multiple, sometimes conflicting, objectives must be simultaneously attended. This thesis aims to show that evolutionary-based techniques can be successfully applied to such problems, leading to novel approaches and methods that address all aforementioned requirements and, at the same time, provide efficient and high accuracy solutions.

In this thesis, we first propose a genetic programming approach to record deduplication. This approach combines several different pieces of evidence extracted from the actual data present in the repositories to suggest a deduplication function that is able to identify whenever two entries in a repository are replicas or not. As shown by our experiments, our approach outperforms existing state-of-the-art methods found in the literature. Moreover, the suggested function is computationally less demanding since it uses fewer evidence. Finally, it is also important to notice that our approach is capable of automatically adapting to a given fixed replica identification boundary, freeing the user from the burden of having to choose and tune this parameter

Based on the previous approach, we also devised a novel evolutionary approach

that is able to automatically find complex schema matches. Our aim was to develop a method to find semantic relationships between schema elements, in a restricted scenario in which only the data instances are available. To the best of our knowledge, this is the first approach that is capable of discovering complex schema matches using only the data instances, which is performed by exploiting record deduplication and information retrieval techniques to find schema matches during the evolutionary process. To demonstrate the effectiveness of our approach, we conducted an experimental evaluation using real-world and synthetic datasets. Our results show that our approach is able to find complex matches with high accuracy, despite using only the data instances.

Contents

1	Introduction	1
1.1	Data Integration	2
1.2	Thesis Statement	5
1.3	Thesis Objectives	6
1.4	Contributions	7
1.5	Thesis Organization	8
2	Related Work	11
2.1	Record Deduplication	11
2.1.1	Domain Knowledge Approaches	12
2.1.2	Probabilistic Approaches	13
2.1.3	Machine Learning Approaches	14
2.2	Schema Matching	16
2.2.1	Schema-Level Approaches	16
2.2.2	Instance-Level Approaches	18
2.2.3	Hybrid Approaches	20
3	Overview of Genetic Programming	21
3.1	Genetic Programming	21
3.2	Solution Representation	23
3.3	Basic Genetic Operations	23
3.4	Evolutionary Process	26

3.5	Generalization Issues	30
4	A GP-based Approach to Record Deduplication	33
4.1	Modeling the Record Deduplication Problem with GP	35
4.2	Fitness Function	37
4.3	Complexity of the Training Phase	38
4.4	Experiments	39
4.4.1	Experimental Datasets	40
4.4.2	Experiments with User-Selected Evidence	43
4.4.3	Experiments with Automatically-Selected Evidence	46
4.4.4	Experiments with the Replica Identification Boundary	52
4.5	Remarks on the Results	60
5	Impact of the GP Parameter Setup on Record Deduplication	63
5.1	The Experimental Dataset	64
5.2	Population and Generation Size	65
5.3	Crossover-Reproduction Proportion and Pairing Methods	67
5.4	Selection Method and Mutation Rates	68
5.5	Initial Random Tree Creation Method	69
5.6	Remarks on the Results	70
6	An Evolutionary Approach to Schema Matching	73
6.1	Proposed Approach	76
6.1.1	Schema Match Solution Representation	77
6.1.2	Record-oriented Strategy	82
6.1.3	Attribute-oriented Strategy	83
6.1.4	Selecting the Best Matches	86
6.2	Experiments	87
6.2.1	Datasets	88
6.2.2	Basic Steps	91

6.2.3	Partially Overlapped Data Scenario	91
6.2.4	Fully Overlapped Data Scenario	93
6.2.5	Disjoint Data Scenario	95
6.3	Remarks on the Results	97
7	Conclusions	99
7.1	Summary of Results	100
7.2	Future Work	101
	Bibliography	105

List of Figures

1.1	Data Integration Approaches (Ziegler and Dittrich, 2004)	2
1.2	Data Integration Process	5
3.1	Example of a Function Mapped as a Tree	24
3.2	Random SubTree Crossover	24
3.3	Random SubTree Mutation	26
3.4	Steady State Algorithm: (1) Selecting parents (2) New offspring is created (3) Parents are replaced in the population	27
3.5	Generational Algorithm: (1) Selecting parents in the present population (2) New offspring is created (3) New individuals are inserted in the next population	29
4.1	Fittest Individual Effort - Generations X F1 Values - Cora Dataset	58
4.2	Fittest Individual Effort - Generations X F1 Values - Restaurants Dataset	59
4.3	Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 1	59
4.4	Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 2	60
4.5	Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 3	60
6.1	Semantic Relationships between Repository Schemas: Complex Matches	75
6.2	A Match Representation Example	78
6.3	Random Generated Individual with Two Schema Matches	80
6.4	Crossover Operation	81
6.5	Match M2 Fitness Evaluation using a String Similarity Measure	83

6.6	Match M2 Fitness Evaluation using the Cosine Similarity Measure	84
6.7	Selecting the Best Matches for the User Output List	87

List of Tables

4.1	Discriminative Attribute Selection - Marlin - Cora Database	44
4.2	Discriminative Attribute Selection - Marlin - Restaurants Database	44
4.3	GP Parameters: (1) Previously Fixed Evidence (2) Previously Fixed Evidence vs Not Previously Fixed Evidence using Synthetic Datasets	45
4.4	User-Selected Evidence - F1 Averages	45
4.5	Cora and Restaurant Datasets - Automatically-Selected Evidence Strategy	47
4.6	Training and Test Times for Cora and Restaurants Datasets - Automatically-Selected Evidence Strategy	47
4.7	Synthetic Datasets - Automatically-Selected Evidence Strategy	50
4.8	Training and Test Times for Synthetic Datasets - Automatically-Selected Evidence Strategy	50
4.9	Number of generations required to reach F1 above 95%	53
4.10	GP Parameters: (1) Previously Fixed Evidence (2) Previously Fixed Evidence vs Not Previously Fixed Evidence using Synthetic Datasets	53
4.11	F1 Results: Identification Boundary - Cora and Restaurant Datasets	54
4.12	F1 Results: Identification Boundary using Synthetic Repositories - Synthetic Datasets	54
4.13	Boundary Variation - Tree Adaptation during Training - Cora Dataset	55
4.14	Boundary Variation - Tree Adaptation during Training - Restaurants Dataset	55
4.15	Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 1	56

4.16	Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 2	56
4.17	Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 3	57
5.1	Standard GP Setup	65
5.2	Population and Generation Size - F1 Averages and Standard Deviations . .	66
5.3	Crossover–Reproduction Proportion and Pairing Method - F1 Averages and Standard Deviations	68
5.4	Selection Method and Mutation Rate - F1 Averages and Standard Deviations	68
5.5	Initial Tree Creation Method - F1 Averages and Standard Deviations . . .	69
6.1	Dataset Characteristics	90
6.2	GP Parameters: (1) Partially Overlapped (2) Fully Overlapped (3) Disjoint	90
6.3	Results: Partially Overlapped Data Scenario	92
6.4	Results: Fully Overlapped Data Scenario	94
6.5	Results: Disjoint Data Scenario	96

Chapter 1

Introduction

The increasing volume of information available in digital media has become a challenging problem for the administrators of existing data repositories. Usually built on data gathered from different sources, data repositories used by information systems such as digital libraries and e-commerce brokers may present records with disparate structure. Also, problems regarding low response time, availability, security, and quality assurance become more difficult to handle as the amount of data gets larger.

In this environment, the decision of keeping repositories with “dirty” data (i.e., with replicas, with no standardized representation, etc.) goes far beyond technical questions such as the overall speed or performance of the systems. The solutions available for addressing this situation require more than technical efforts, they need management and cultural changes as well (Bell and Dravis, 2006).

To avoid this situation, it is necessary to analyze the main causes for “dirt” in data repositories. The most common ones are problems found in traditional data input interfaces (e.g., free text data input fields) that do not apply data standardization and deficiencies in data integration. This last one particularly presents a great potential to cause errors and inconsistencies that require expensive and difficult correction.

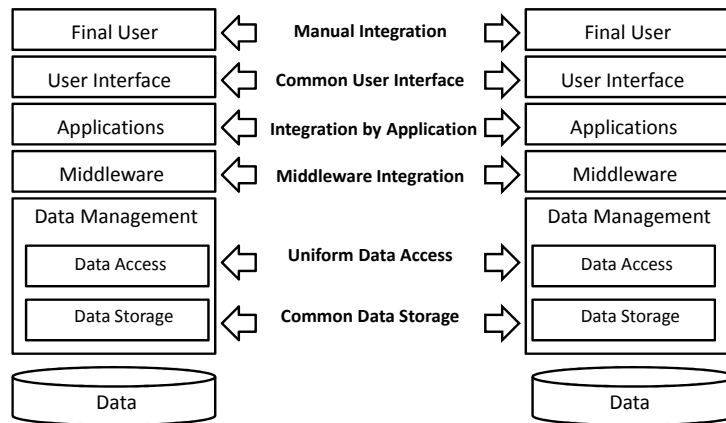


Figure 1.1. Data Integration Approaches (Ziegler and Dittrich, 2004)

1.1 Data Integration

In general, data integration aims to combine data from different sources (data repositories such as databases, digital libraries, etc.) by adopting a global data model and by detecting and resolving schema and data conflicts so that a homogeneous, unified view can be provided. This unified view aims at giving users the illusion of interacting with one single information system. Usually, there are two main reasons for data integration (Ziegler and Dittrich, 2004):

- Given a set of data sources, an integrated view eases data access and reuse through a single data access point.
- Given a certain information need, data from different complementary sources can be combined to provide a more comprehensive basis to satisfy this information need.

However, the path towards this unified view presents a serious problem: all this data should adopt the same abstraction principles, in other words, they should be accessed by the same semantic concepts in order to be fully used. Also, it is important to stress that most data repositories are not designed aiming at easing the integration process.

For these reasons, it is not rare to find several different ways of representing the same concepts, entities and ideas. This leads to different approaches for data integration.

As illustrated in Figure 1.1, detailed in (Ziegler and Dittrich, 2004), the data integration process may occur at several distinct layers, each one related to a specific functionality usually present in most systems. At the user layer, users access data and services through various interfaces that run on top of different applications. Applications may use a middleware - transaction processing (TP) monitors, SQL middlewares, etc. - to access data via a data access interface. The data itself is managed by a data storage system. Usually, database management systems (DBMS) are used to combine data access and storage functionalities.

Nonetheless, the integration process for both the data access and storage layers present some of the most defiant situations. This occurs because these layers require clean data repositories (i.e., repositories that are free from problems caused by flaws in semantic integration, such as lack of data standardization and replicas) in order to provide reliable services for the upper level layers.

Even for these two layers (data access and storage), it is possible to find different descriptions for their internal steps, that exist to attend specific needs and objectives. However, it is possible to describe a common sequence of steps for the integration process of these layers (Batini et al., 1986). Each step is generally related to common data integration subproblems. These steps represent data integration tasks that may be organized in two phases, as illustrated in Figure 1.2, whose main objectives are briefly described in the following:

1. Semantic Integration Phase

- a) Pre-Processing: involves structural and data type analysis in order to gather information about the modeling and concepts adopted in the data repositories that will be integrated.
- b) Schema Matching: comprises finding valid alternatives for matching and

posterior mapping between the concepts and modeling adopted in the data repositories.

- c) Post-Processing: comprises schema matching validation (or ranking, when there is more than one valid suggestion) and data mapping. In this step, additional processing can be required in order to prepare the repository for further steps or to present the results to the user.

2. Instance Integration Phase

- a) Cleaning and Standardization: comprises data cleaning and the establishing of patterns and standards that the data must obey, in order to ease later integration steps.
- b) Blocking: comprises similarity analysis and initial processing (to build blocks) in order to optimize the replica identification step. The main idea behind the blocking strategies is to create blocks whose records have the smallest possible probability of being matches of records on the other blocks.
- c) Replica Identification: involves comparison between repository records to find replicated entries, related to the same real-world entity. There are two main tasks that can be viewed as equivalent in replica identification: *Record Linkage* (or record association) which is the task of finding record entries that refer to the same real world entity, however, these records are stored in different data repositories. *Record Deduplication* which is the task of identify replicas of records found in the same data repository.
- d) Clustering: comprises identification of original and replicated records, and analysis of the differences found between the same record related entities found.

The semantic integration phase, which is usually accomplished by the schema matching step, allows the implementation of services that are available in the data

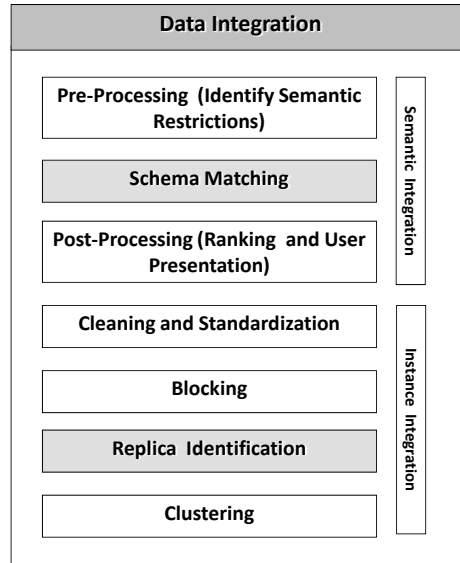


Figure 1.2. Data Integration Process

access layer. The instance integration phase, on the other hand, is responsible for maintaining the consistency of the data stored in the unified data layer.

In some specific environments, some of these steps are not performed or even required due to the characteristics of the data repositories, as well as to the objectives that must be fulfilled at the end of the integration process. In some scenarios, only the cleaning and standardization step may be required, in others, the replica identification step requires a further clustering analysis. We refer the reader to (Ziegler and Dittrich, 2004) for a more detailed discussion of the data integration process.

1.2 Thesis Statement

Two specific problems related to the data integration process - schema matching and replica identification - present a large solution space. This space is computationally expensive and technically prohibitive to be intensively and exhaustively explored by traditional approaches. Moreover, the solutions for these problems usually require that multiple, sometimes conflicting, objectives must be simultaneously attended. This

thesis aims to show that evolutionary-based techniques can be successfully applied to such problems, leading to novel approaches and methods that address all requirements and, at the same time, provide efficient and high accuracy solutions.

1.3 Thesis Objectives

The main objective of this thesis is to propose evolutionary approaches, mainly based on Genetic Programming (GP), to solve data integration related problems. The problems we address are record deduplication (which is central to the replica identification step) and schema matching, which are related, respectively, to the instance and semantic integration phases of the data integration process, as described in Figure 1.2.

Regarding record deduplication, we propose a GP-based approach that combines several different pieces of evidence extracted from the actual data stored in a repository to produce a deduplication function that is able to identify whether two or more entries are replicas or not.

Based on our solution to the record deduplication problem, we devised a novel evolutionary approach to schema matching, strongly inspired by GP, that aims at automatically finding complex matches between schema elements of two semantically related data repositories. This novel approach was developed since some subtleties found in the schema matching problem require an evolutionary technique that extends the “classic” GP, as we shall explain in Chapter 6.

The reasons why we have adopted evolutionary approaches to address these problems relies on the fact that they present the following characteristics:

1. Large search space: both problems present solutions that can be build by combining small subsolutions for smaller problems. The optimal combination of these subsolutions involves a large search space, that is computationally expensive and technically prohibitive to be intensively and exhaustively explored.

2. Multi-objective solutions: both problems require that several objectives must be attended simultaneously in order to provide useful valid solutions.
3. Feature selection: despite the problem of finding how to combine subsolutions (characteristic 1), sometimes there are situations in which it is necessary to identify when some of the subsolutions should or not be part of the combined (final) solution. Each subsolution represents a different dimension component of the final solution, but, in some cases, the optimal solution may not include (or require) all subsolutions available.

Evolutionary approaches comprise some techniques that are known for their capability to efficiently find suitable answers to a given problem, without having to explore the entire space of solutions and when there is more than one objective to be accomplished (Banzhaf et al., 1998). In fact, some of them, particularly GP, have been successfully applied to several problems related to information and data management, such as document and information retrieval (Bergström et al., 2000; Cummins and O’Riordan, 2006; Gordon, 1988), text classification (Hirsch et al., 2007; Masand, 1994), ranking for web search (Fan et al., 2004), content-based image retrieval (Torres et al., 2009), content target advertising (Lacerda et al., 2006), and record deduplication (de Carvalho et al., 2006).

1.4 Contributions

The main contributions of this thesis are:

1. A GP-based approach to record deduplication (de Carvalho et al., 2008a) that:
 - is competitive when compared with existing state-of-the-art methods found in the literature, and outperforms them as demonstrated by our experiments (de Carvalho et al., 2008a);

- provides solutions less computationally intensive, since it suggests solutions that use the available evidence more efficiently;
 - frees the user from the burden of choosing how to combine similarity functions and repository attributes (this distinguishes our approach from all existing methods, since they require user-provided settings);
 - frees the user from the burden of choosing the replica identification boundary value, since it is able to automatically adapt the similarity functions to a fixed value for this deduplication parameter.
2. A detailed experimental study of our GP-based approach to record deduplication (de Carvalho et al., 2008b) that:
- shows how the selection of GP parameters impacts the performance of the record deduplication task;
 - provides guidelines for setting the parameters of our approach.
3. A novel evolutionary approach to complex schema matching that:
- uses only the actual data in the repositories to find matches between schema elements - to the best of our knowledge, this is the only approach capable of finding complex matches (n-n and 1-n) in this restricted scenario;
 - exploits novel matching strategies based on record deduplication and information retrieval techniques to find the matches in several repository data scenarios (from repositories with overlapping data to those with no common data) with high accuracy;
 - can be integrated with or be used to improve the effectiveness of existing data integration systems.

1.5 Thesis Organization

The remaining of the thesis is organized in six chapters as follows:

-
- Chapter 2 presents a review of major work related to the two problems we address in this thesis - record deduplication and schema matching.
 - Chapter 3 presents an overview of genetic programming focusing on those concepts that are the basis of our approaches. Particularly, it describes how solutions are represented in GP, the basic genetic operations and the evolutionary process.
 - Chapter 4 describes our GP-based approach to record deduplication and presents the experiments we conducted to evaluate it.
 - Chapter 5 presents an experimental study on how the genetic programming parameters affect the results of our GP-based approach to record deduplication.
 - Chapter 6 describes our novel evolutionary approach to schema matching and presents the experiments we conducted to evaluate it.
 - Finally, Chapter 7 presents a summary of our results and discusses future work.

Chapter 2

Related Work

In this chapter, we present a review of major work related to the two problems we address in this thesis: record deduplication and schema matching.

2.1 Record Deduplication

Record deduplication is a growing research topic in database and related fields such as digital libraries. Today, this problem arises mainly when data is collected from disparate sources using different information description styles and metadata standards. Other common place for replicas is found in data repositories created from OCR documents. These situations can lead to inconsistencies that may affect many systems such as those that depend on searching and mining tasks.

To solve these inconsistencies it is necessary to design a deduplication function that combines the information available in the data repositories in order to identify whether a pair of record entries refers to the same real-world entity. In the realm of bibliographic citations, for instance, this problem was extensively discussed by Lawrence et al. (1999a; 1999b). They propose a number of algorithms for matching citations from different sources based on edit-distance, word matching, phrase matching, and subfield extraction.

As more strategies for extracting disparate pieces of evidence become available,

many works have proposed new distinct approaches to combine and use them. Elmagarmid et al. (2007) classify these approaches into the following two categories:

- **Ad-Hoc or Domain Knowledge Approaches:** This category includes approaches that usually depend on specific domain knowledge or specific string distance metrics. Approaches that make use of declarative languages (Elmagarmid et al., 2007) can be also classified in this category.
- **Training-based Approaches:** This category includes all approaches that depend on some sort of training – supervised or semi-supervised – in order to identify the replicas. Probabilistic and machine learning approaches fall into this category.

Next, we briefly comment on some works based on these two approaches (domain knowledge and training-based), particularly those that exploit the domain knowledge and those that are based on probabilistic and machine learning techniques, which are the ones more related to our work.

2.1.1 Domain Knowledge Approaches

The idea of combining evidence to identify replicas has pushed the data management research community to look for methods that could benefit from domain-specific information found in the actual data as well as for methods based on general similarity metrics that could be adapted to specific domains (Elmagarmid et al., 2007).

As an example of a method that exploits general similarity functions adapted to a specific domain, we can mention the work by Chaudhuri et al. (2003). There the authors propose a matching algorithm that, given a record in a file (or repository), looks for another record in a reference file that matches the first record according to a given similarity function. The matched reference records are selected based on a user-defined minimum similarity threshold. Thus, more than one candidate record may be returned. In such cases, the user is required to choose one among them indicating which is the closest one. Records matching on high weight tokens (strings) are more

similar than those matching on low weight tokens. The weights are calculated by the well-known IDF weighting scheme (Baeza-Yates and Ribeiro-Neto, 1999).

This weighting scheme is also used in WHIRL (Cohen, 2000), a database management system that supports similarity joins among relations that have free text attribute values. In (Carvalho and da Silva, 2003), the authors use the vector space model for computing similarity among fields from different sources and evaluate four distinct strategies to assign weights and combine the similarity scores of each field. As a result of their experiment, they found that using evidence extracted from individual attributes improves the results of the replica identification task.

2.1.2 Probabilistic Approaches

Newcombe et al. (1959) were the first ones to address the record deduplication problem as a Bayesian inference problem (a probabilistic problem) and proposed the first approach to automatically handling replicas. However, their approach was considered empirical (Elmagarmid et al., 2007) since it lacks a more elaborated statistical ground.

After Newcombe et al.’s work, Fellegi and Sunter (1969) proposed a more elaborated statistical approach to deal with the problem of combining evidence. Their method relies on the definition of two boundary values that are used to classify a pair of records as being replicas or not. Tools that implement this method, such as Febrl¹, usually work with two boundaries as follows:

1. Positive Identification Boundary – if the similarity value lies above this boundary, the records are considered as replicas;
2. Negative Identification Boundary – if the similarity value lies below this boundary, the records are considered as not being replicas.

For the situation in which similarity values stand between the two boundaries, the records are classified as “possible matches” and, in this case, a human judgment is

¹Freely Extensible Biomedical Record Linkage - <http://sourceforge.net/projects/febrl>

necessary.

2.1.3 Machine Learning Approaches

More related to our work are those proposals that apply machine learning techniques for deriving record-level similarity functions that combine field-level similarity functions, including the proper assignment of weights (Bilenko et al., 2003; Bilenko and Mooney, 2003; Cohen and Richman, 2002; Tejada et al., 2001). These proposals make use of a small portion of the available data for training. This training dataset is assumed to have characteristics similar to the test dataset. This allows learning techniques to generalize their solutions when dealing with unseen data that share these characteristics. The good results usually obtained with these techniques have empirically demonstrated that those assumptions are valid.

Bilenko et al. (2003) and Bilenko and Mooney (2003) use a machine learning technique to improve both the similarity functions that are applied to compare record fields and the way pieces of evidence are combined. In their work, extracted evidence is encoded as feature vectors, that are used to train a support vector machine (SVM) classifier to better combine them in order to identify replicas, in a system named Marlin. The main idea behind this approach is that, given a set of record pairs, the similarity between two attributes (e.g., two text strings) is the probability of finding the score of best alignment between them, so the higher the probability, the bigger the similarity between these attributes. They compare the Marlin system with several other effective learning methods and show that it outperforms all of them. The Marlin system is used in this thesis as our baseline since it is the current state-of-the-art method for the record deduplication problem.

The adaptive approach presented by Cohen and Richman (2002) consists of using examples for training a learning algorithm to evaluate the similarity between two given names, i.e., strings representing identifiers. This approach is applied to both clustering and pair-wise matching, achieving satisfactory experimental results.

Active Atlas (Tejada et al., 2001) is a system whose main goal is to learn rules for mapping records from two distinct files in order to establish relationships among them. During the learning phase, the mapping-rule and the transformation weights are defined. The process of combining the transformation weights is executed using decision trees. This system differs from the others in the sense that it tries to reduce the amount of necessary training, relying on user-provided information about the most relevant cases for training. Before Marlin, this system was the state-of-the-art solution for the record deduplication problem.

An approach distinct from the previous ones is presented by Guha et al. (2004). The main idea is to generate individual rankings for each field based on similarity scores generated. Then, the individual rankings are merged in order to obtain a global ranking such that the distance among the individual rankings is minimized. By doing so, the top records in this global ranking are considered to be the most similar to the input record. Notice that this approach requires no training. Unfortunately, the experiments conducted do not evaluate the quality of the global ranking with respect to the record matching effectiveness.

In (de Carvalho et al., 2006), we propose a GP-based approach to improve results produced by the Fellegi and Sunter’s method (1969). Particularly, we use GP to balance the weight vectors produced by the Fellegi and Sunter’s method, in order to generate a better evidence combination than the simple summation used by that method. Our experiments with real datasets show that our approach achieves improvements over 7% in precision.

In this thesis, we use GP to find the best evidence combination in a generic framework that is independent of any other technique (de Carvalho et al., 2008a). We use GP to pair attributes and similarity functions in order to combine the available evidence, and compare this strategy with the traditional human chosen evidence combination. Another important difference from our previous work (de Carvalho et al., 2006) is that we use only one identification boundary value, which prevents the existence of results

that would require a human judgment to determine whether selected record pairs are replicas or not.

2.2 Schema Matching

Schema matching is a critical step in any data integration process (Elmagarmid et al., 2007). The problem usually arises due to the many possible ways of expressing the same ideas and concepts when modeling data. For example, database designers may opt for different representations of a same real-world entity (as an entity type or an attribute) or even adopt different data modeling paradigms.

Rahm and Bernstein (2001) propose a taxonomy for schema matching approaches, in which they classify the existing approaches in the following categories:

- **Schema-Level:** This category includes approaches that usually use structural schema information as well as some knowledge on the application domain considered.
- **Instance-Level:** This category includes all approaches that use only the stored data instances as source of evidence.
- **Hybrid:** This category includes all approaches that combine information extracted from both the schema structure and the stored instances.

Next, we briefly comment on some representatives of these three approaches, focusing on those that exploit only the data available in the repositories (instance level) which are the ones more related to our work.

2.2.1 Schema-Level Approaches

Several works on schema matching use structural information, such as attribute names, to find the relationships among the schema elements of two repositories. Batini

et al. (1986), for example, propose a methodology for integrating schemas of relational databases into a global, unified schema, using the information available in the associated entity-relationship schemas (e.g., entities, relationships, and cardinalities). However, this methodology lacks scalability since it requires manual match identifications, being infeasible for large scale schemas.

Other works present broader alternatives. For instance, they use object labels from schemas modeled with the OO paradigm, entity descriptions, attribute types, etc. One common way of using all this information is to induce association rules to unearth relationships between the schemas, as in systems such as TranScm (Milo and Zohar, 1998) and ARTEMIS (Castano et al., 2001).

In the TranScm (Milo and Zohar, 1998) system, the authors use a schema translation process, based on relationships created from attribute information extracted from the schema, such as their data types. The process aims at finding how to transform one schema into another by matching the attribute characteristics from two schemas. The system usually requires user insights and manually provided examples in order to be used.

In ARTEMIS (Castano et al., 2001), the schema matching is accomplished by using a global similarity coefficient. This coefficient is calculated using a previously stored dictionary and thesauri information. The system uses the labels and the descriptions of the attributes to identify potential relationships between schema elements. Then, it builds affinity trees that estimate how related the attributes are. These trees are then compared using association rules, and the best ones are presented for the final user. Nonetheless, this proposal ignores the problems of complex matches, presenting only 1-1 matches as solutions.

Other works represent the schema structure found in the repositories as graphs in order to compare them. The main examples of this strategy are the Cupid (Madhavan et al., 2001) and Similarity Flooding (Melnik et al., 2002) systems.

The Cupid system (Madhavan et al., 2001), which may be classified as a hybrid

matcher², unifies several strategies that aim at analyzing the repository structures in order to find schema matches. One of the adopted strategies is based on linguistic analyses of attribute names and object labels. Cupid is also able to extract information from labels used by keys, view names and structural constraints, such as relationship cardinalities. The final result is the combination of all this information, which is presented to the user as a graphical model of the relationships found between the schemas. It must be noticed that Cupid makes extensive use of external resources such as dictionaries, thus, its use may be problematic depending on the repository domain if these resources are not available.

Similarity Flood (Melnik et al., 2002) presents a pairing algorithm that looks for “semantic” points (e.g., same words or similar expressions for the same concepts) that can be used as milestones for building associations between the candidate schemas. The algorithm input are the two graphs that represent the schema structures and its output is a match table for the elements that were connected. This final solution requires user adjustments and, for this reason, the authors describe Similarity Flood as a user aid tool. It helps finding a set of match suggestions for which the user has only to provide fine-grained adjustments.

2.2.2 Instance-Level Approaches

As explained before, instance-level approaches build their strategies to find matches between schema elements using only the data available in the repositories being processed. Some of the most effective systems in this category use probabilistic and machine learning techniques. For example, Li and Clifton propose a system for this task, named SemInt (Li and Clifton, 1994, 2000), that is based on neural networks. SemInt categorizes the repository attribute values into specific types and, then, trains a neural network to find the similarity between these values. In the most recent version of this system (Li and Clifton, 1994), additional information is exploited in the neural network

²Its first version was purely instance-level.

training. Some of this additional information is extracted from structural elements (i.e., name of the attributes). However, the system is still only capable of identifying 1-1 matches.

Another elaborated approach that uses machine learning techniques is proposed by Doan et al. (2000; 2001). In their work, the LSD (Learning Source Description) system utilizes a multi-level architecture that combines the results of different learning strategies. LSD aims at semi-automatically processing the schema matching task. The system requires initial examples of semantic mappings from the user and employs these examples to train each machine learning technique available. Using the candidate solutions provided for each learner, a meta-learner is used to combine them in a single final solution to be presented to the user.

The first version of LSD (Doan et al., 2000) relied on the analysis of repository data samples to provide user selected instance-level examples for training. However, its most recent version (Doan et al., 2001) received upgraded modules that use schema information to improve the results. Thus, its approach can now be classified as a hybrid one. Nonetheless, the LSD system only presents solutions for 1-1 matches and because of its dependence of external domain information, such as dictionaries, it can present problems (e.g., no dictionaries or thesauri available) in some data domains.

More recently, Warren and Tompa (2006) have described a specific multi-column string matching approach that, in fact, proposes data mappings (i.e., it identifies the operations that are required to transform an attribute value into another one) between schema elements. Their approach is able to discover several complex combinations of strings in order to find the mappings. However, as pointed out by the authors, they require some common data between the repositories in order to find the solutions. Additionally, the approach also needs some external information, such as a previously selected set of attributes, in order to solve complex mappings.

2.2.3 Hybrid Approaches

Systems and tools that are built based on hybrid schema matching approaches use both instance and structural information to find the relationships between schemas. As discussed by Rahm and Bernstein (2001), most of the current research are in fact focused on hybrid approaches. This can be explained since, separately, both instance and schema level approaches present deficiencies that may be solved when their information is combined.

An example of a hybrid schema matching system is Clio (Hernández et al., 2001; Miller et al., 2000; Yan et al., 2001), which employs a user-feedback technique and an analysis of the data stored in the repositories to suggest the best schema matches. The Clio system gathers information from the users in cycles, improving the solutions in each interaction. Both, information extracted from the schemas (e.g., attribute names) and from the instances, are refined in this process.

Another example is SKAT (Mitra et al., 1999, 2000), which provides a hybrid approach for repositories that are modeled using the OO paradigm. The system explores the cardinality of the associations between objects, as well as their labels.

A more elaborated system is iMAP (Dhamankar et al., 2004). This system comprises several modules, featuring different pre-processing data strategies, machine learning techniques, and schema processing approaches. All the information coming from these different modules is combined during a post-processing stage, and several options and suggestions of solutions are presented to the final user.

In this thesis, we propose a novel evolutionary instance-level approach that aims at automatically finding complex schema matches. Our major objective was to develop a method that provided useful information about existing semantic relationships between schema elements, in a restricted scenario in which only the data instances are available. To the best of our knowledge, our approach is the first instance-level approach that deals with complex matches.

Chapter 3

Overview of Genetic Programming

In this chapter, we present an overview of genetic programming focusing on those concepts that are the basis of our approaches. Particularly, we describe how solutions are represented, the genetic operations and the evolutionary process.

3.1 Genetic Programming

Evolutionary programming is based on ideas inspired on the naturally observed process that influence virtually all living beings, the *natural selection*. Genetic Programming (GP) (Koza, 1992; Banzhaf et al., 1998) is one of the best known evolutionary programming techniques. It can be seen as an adaptive heuristic whose basic ideas come from the properties of the genetic operations and natural selection system. It is a direct evolution of programs or algorithms used for the purpose of inductive learning (supervised learning), initially applied to optimization problems. GP, as well as other evolutionary techniques, is also known for its capability of working with multi-objective problems, that are normally modeled as environment restrictions during the evolutionary process (Banzhaf et al., 1998).

GP-based systems are also widely known for their good performance on searching over very large - possibly infinite - search spaces, where the optimal solution in many cases is not known, usually providing near-optimal answers (Koza, 1992; Banzhaf et al.,

1998). This happens because the GP algorithm is able to search, in parallel, several points on the problem state space with numerous search directions. As stated by Koza (1992), “the fact that the GP algorithm operates on a population of individuals, rather than a single point in the search space of the problem, is an essential aspect of the algorithm”. This can be explained since the population serves as the pool of the probably-valuable genetic material, which is used to create new solutions with probably-valuable new combinations of features.

The main aspect that distinguishes GP from other evolutionary techniques (e.g., genetic algorithms, evolutionary systems, genetic classifier systems) is that it represents the concepts and the interpretation of a problem as a computer program - and even the data is viewed and manipulated in this way. This special characteristic enables the GP computer program representation to model any other machine learning representation (Banzhaf et al., 1998).

Moreover, another advantage of GP over other evolutionary techniques is its applicability to symbolic regression problems, since the structures of the computer program representation are variable (i.e., they have no length limitations). According to Angeline and Kinnear (1996), “while GP does not mimic nature as closely as do genetic algorithms, it does offer the opportunity to directly evolve programs of unusual complexity, without having to define the structure or size of the program in advance”.

It is important to notice that most of the problems solved (or approximated) by GP are, in some ways, symbolic regression problems. This means that GP is able to discover the independent variables and their relationships with each other and with any dependent variables. Thus, GP can find the correct functional form that fits the data and discover the appropriate coefficients (Angeline and Kinnear, 1996). Solving this kind of problem is noticeably more complex than linear regression or polynomial regression problems.

3.2 Solution Representation

Usually, GP evolves a population of length-free data structures, also called *individuals*, each one representing a single solution to a given problem. For example, in our GP-based approach to record deduplication, the individuals are trees that represent arithmetic functions, as illustrated in Figure 3.1. In this representation, each variable (a numerical value) is represented by a leaf in the tree. The internal nodes represent operations that are applied to the leaves, such as simple mathematical functions (e.g., $+$, $-$, $*$, $/$, exp) that manipulate these values.

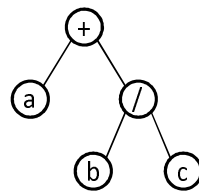
In GP, there are several ways of spawning the initial population of individuals that will participate in the evolutionary process. The most common way is to create random trees using the functions and terminals available. Some approaches also make use of the user input to suggest how to create the initial population. This is achieved by using the tree (problem solution) provided by the user for creating variants, for example, by applying mutations or crossing them with other random created trees.

Other ways of spawning the initial population are: *full depth* – all trees are created with the same maximum allowed depth; *grown* – the trees are created with a random number of nodes, but not exceeding the maximum allowed depth; *ramped half-and-half* – half of the trees is created using the full depth method and the other half, using the grown method.

3.3 Basic Genetic Operations

During the evolutionary process, the individuals are handled and modified by genetic operations such as *reproduction*, *crossover*, and *mutation* (Koza, 1992), in an iterative process that is expected to spawn better individuals (solutions to the proposed problem) in the subsequent generations.

Reproduction is the operation that copies individuals without modifying them. Usually, this operation is used to implement an elitism strategy (Koza, 1992), that is



$$\text{tree}(a,b,c) = a + (b/c)$$

Figure 3.1. Example of a Function Mapped as a Tree

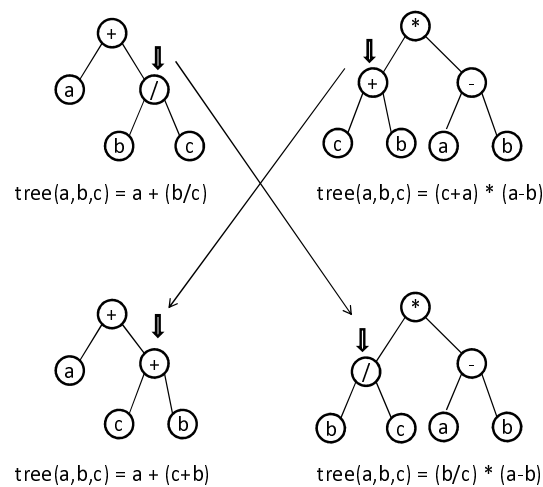


Figure 3.2. Random SubTree Crossover

adopted to keep the genetic code of the fittest individuals across the changes in the generations. According to Koza (1992), “the operation of reproduction for the genetic programming paradigm is the basic engine of Darwinian reproduction and survival of the fittest”. Thus, if a good individual is found in earlier generations, it will not be lost during the evolutionary process.

The crossover operation allows genetic content (e.g., subtrees) exchange between two parents, in a process that can generate two or more children. In GP, two parent trees are selected according to a matching (or pairing) policy and, then, a random subtree is selected in each parent. Child trees are the result from the swap of the selected subtrees between the parents, as illustrated in Figure 3.2.

One important concern is to discover how to balance the reproduction and crossover

rates. If the number of reproduced individuals is high, there will be no diversity among the population. Also, the evolutionary process may require more generations to achieve good solutions, since there are less new individuals that are created in each generation.

However, if the number of individuals that will suffer crossover is higher than necessary, the “pool” of good individuals will be smaller. This happens because the crossover operation, that is responsible for creating new individuals, also presents a destructive effect over good solutions (Banzhaf et al., 1998). A major consequence of this is that some good individuals may be lost during the process. This fact can be balanced by keeping some of the good individuals, that are found during the process, unaltered over the generations (using the reproduction operation).

Another concern is how to choose the most adequate way to create the pairs of individuals that will participate in the crossover. There are three well known strategies. The first one is the *random pairing* – the pairs of individuals that will exchange genetic materials are randomly selected. The second is the *ranking pairing* – the individuals are paired in crescent order of fitness. The third is the *mirror pairing* - the individuals are ordered by fitness and paired as $(1, n), (2, n - 1), (3, n - 3), \dots, ((n/2), (n/2) - 1)$, where n is the size of the population (which forces the exchange of genetic materials between good and bad individuals).

The mutation operation has the role of keeping a minimum diversity level of individuals in the population, thus avoiding premature convergence. Convergence is also known as *allele loss*. Allele¹ loss is the natural loss of traits in the gene pool during the evolutionary process. Thus, severe allele loss results in a population unable of solving a problem with the available gene pool. For these reasons, mutation is often responsible for preventing the evolutionary process from being stuck in minimum or maximum local values.

Every individual resulting from the crossover operation has an equal chance of suffering a mutation. In a GP tree scheme, a random node in the chosen tree is

¹In biology, the term allele refers to the range of values a particular gene can assume.



Figure 3.3. Random SubTree Mutation

selected and the pointed subtree is replaced by a new randomly created subtree, as illustrated in Figure 3.3. However, if the mutation rate is set to a higher value, it usually disrupts good solutions, slowing the evolutionary process. Like in nature, mutations are destructive and, for this reason, it corresponds to a parameter that is kept with lower values.

It is important to notice that, in our experiments (described in Chapters 4, 5, and 6) all operations for node replacements and insertions performed by the mutation and crossover operations consider equal (and constant) probabilities. All nodes have the same probability of being chosen during these operations, in order to guarantee the diversity of the individuals within the genetic pool.

3.4 Evolutionary Process

There are two ways to conduct the GP evolutionary process (Banzhaf et al., 1998): by applying a *steady state* algorithm that does not consider distinct generations, or a *generational* one that does.

A steady state algorithm copies the next generation of individuals into the same population from which the parents were previously selected. When the genetic op-

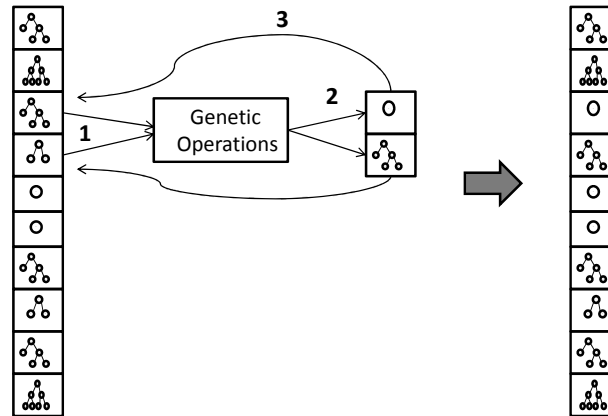


Figure 3.4. Steady State Algorithm: (1) Selecting parents (2) New offspring is created (3) Parents are replaced in the population

erations on the parents are completed, the new offspring takes the place of members of the previous generation within that population. In sum, the new individuals of the population are exchanged with the old members of the same population (Banzhaf et al., 1998), as illustrated in Figure 3.4. This process of creating the new population continues until no individuals of the previous generation remain or another termination criterion is established, such as processing time. Thus, in this case, the evolutionary process flows in a continuous run.

A generational algorithm, on the other hand, comprises well-defined and distinct generation cycles. The steps that comprise this kind of algorithm are the following:

1. Initialize the population (with random or user provided individuals).
2. Evaluate all individuals in the present population, assigning a numeric rating or fitness value to each one.
3. If the termination criterion is fulfilled, then execute the last step. Otherwise continue.
4. Reproduce the best n individuals into the next generation population.
5. Select m individuals that will compose the next generation with the best parents.

6. Apply the genetic operations to all individuals selected. Their offspring will compose the next population. Replace the existing generation by the generated population and go back to Step 2.
7. Present the best individual(s) in the population as the output of the evolutionary process.

The evaluation in Step 2 is done by assigning to an individual a value that measures how suitable that individual is to the proposed problem. In our GP experimental environment, individuals are evaluated on how well they learn to predict good answers to a given problem, using the set of functions and terminals available. The resulting value is also called *raw fitness* and the evaluation functions are called *fitness functions*.

The fitness value can be seen as a “direction” or “path” that a GP suggested solution takes in its evolutionary process. In other words, it represents an individual’s ability to overcome the difficulties within an environment for available resources. Thus, by using this fitness value, in Step 5 it is possible to select which individuals should be in the next generation.

The selection step applies a criterion for choosing the individuals that should be in the next generation. After the evaluation, each solution has a fitness value that measures how good or bad it is to the given problem. Using this value, it is possible to decide whether an individual should be in the next generation.

Strategies for the selection operation may use very simple or complex techniques, varying from just selecting the best n individuals to randomly selecting the individuals proportionally to their fitness. The mostly adopted strategies are: *roulette wheel* – the probability of selection is proportional to the individual’s fitness; *tournament* – for each available position in the next generation, two individuals are randomly chosen and the fittest one is selected; *random* – the selection is made randomly; *ranking* – all n available positions are occupied by the n first fittest individuals; *greedy* – a small group of fittest individuals have their chances of being selected improved over the others ones

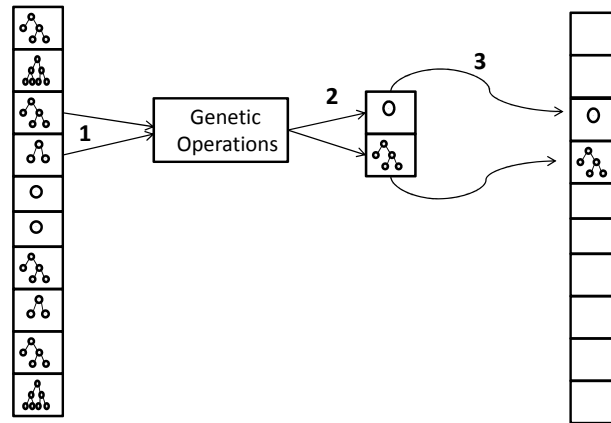


Figure 3.5. Generational Algorithm: (1) Selecting parents in the present population (2) New offspring is created (3) New individuals are inserted in the next population

in the population. For more details on each of these strategies, see (Banzhaf et al., 1998) and (Koza, 1992).

These selection strategies have different results on the balance of the number of fitted and unfitted individuals among the population. Since different problems may require different experimental setups, choosing the most suitable selection strategy for a specific problem may improve the quality of the suggested solutions and may require less resources.

Notice that a generational algorithm uses two populations at the selection stage, as illustrated in Figure 3.5. One population contains the parents to be selected (the “present population”) and the other population holds their descendents (the “next generation”). The steady state algorithm uses the same population for both parents and their descendents.

In this thesis, the GP evolutionary process is guided by a generational algorithm (Banzhaf et al., 1998). We adopted this approach since it captures the basic idea behind several evolutionary strategies and, also, because the generational algorithm is known to be better than the steady state algorithm when dealing with small populations (e.g., less than 100 individuals), as reported in (Kinnear, 1993). In smaller

populations, the advantages of the steady state algorithm are outweighed by “the negative effects of genetic drift²” (Kinnear, 1993).

Regarding to the generational algorithm, it is important to point out that the termination criterion on Step 3 depends on the particular application. In the case of our proposed approaches, we stop after a fixed number of generations is processed, in order to limit processing time. However, it is also possible to relate the termination criterion to the fitness (e.g., if the fitness value reaches a previously fixed boundary, the evolutionary process stops).

3.5 Generalization Issues

Machine learning techniques, such as GP, use valid or useful solutions to find other ones with the same characteristics of those given as example. However, it is possible that these techniques also find some solutions that are only useful in specific situations, such as when small datasets are used.

Usually, the main goal of such techniques is to look for solutions that are able to attend all requirements for both the controlled (when previously known solutions are given as examples) and real-case situations/scenarios. For this reason, machine learning techniques usually make use of a two-phase strategy to assure that the final solutions can endure both scenarios: the training and the test phases.

The training phase aims at “teaching” the learning technique to identify the relevant characteristics of good solutions (that are given as guiding examples). In most situations, this training is accomplished by the use of positive examples (i.e., valid or acceptable solutions), negative examples (i.e., invalid or unacceptable solutions) or both at the same time. From these examples, used as the *training set*, it is expected that the technique will successfully identify good solutions in new sets of examples.

The test phase, also referred to as “evaluation” phase, is required to verify that the solutions found in the training phase are useful. For this, the individuals are

²Genetic drift is the change in the relative frequency with which an allele occurs in a population.

tested against a collection of examples, whose behavior (or set of characteristics) is equal or similar to that of the training set, which is also called the *evaluation example set*. This procedure is necessary as a means to assure that the solution behavior in a representative sample of a population is the same (or valid) for other samples of this same population, in other words, to assure that the solution is generalizable.

When using GP (and other machine learning techniques), there are two major points that must be observed to assure generalizable solutions: sampling and overfitting. Sampling is a impacting factor, since bad sampling techniques will lead to a subset of the population with different characteristics from the ones found in the superset. For this reason, sampling must be done with attention in order to assure valid training and evaluation sets. Otherwise, the learning techniques will fail to provide useful solutions. Overfitting occurs when a suggested solution successfully works in the training set, but its behavior in the evaluation set is very different, usually providing poor results.

With respect to overfitting. Banzhaf et al. (1998) points out three major causes for it:

1. Complexity of the suggested solution: the more simple the solution, the higher the probability that it is generalizable.
2. Training effort and time: in neural networks, it is very important to stop the learning process before the overfitting occurs. However, in GP (and other evolutionary techniques), it is not clear when it occurs. This happens because finding the right moment to end the training is a complex and still open problem.
3. Size of the training set: the smaller the training set (i.e., not representative), the less trustful will be the solutions obtained from this set.

In our experiments, described in Chapters 4, 5, and 6, we provide detailed information on the sampling techniques and, also, the results achieved with both the training and test sets. This has been done in order to show that the results are statistically equivalent in both sets.

Chapter 4

A GP-based Approach to Record Deduplication

Record deduplication is the task of identifying, in a data repository, records that refer to the same real world entity or object in spite of misspelling words, typos, different writing styles or even different schema representations or data types. There has been a large investment from private and government organizations in the development of methods for removing replicas from data repositories (Wheatley, 2004; Bell and Dravis, 2006). This is due to the fact that clean and replica-free repositories not only allow the retrieval of higher-quality information but also lead to a more concise data representation and to potential savings in computational time and resources to process this data.

In this chapter, we propose a genetic programming (GP) approach to record deduplication. Our approach combines several different pieces of evidence extracted from the data content to produce a deduplication function that is able to identify whether two or more entries in a repository are replicas or not. Since record deduplication is a time consuming task even for small repositories, our aim is to foster a method that finds a proper combination of the best pieces of evidence, thus yielding a deduplication function that maximizes performance using a small representative portion of

the corresponding data for training purposes. Then, this function can be used on the remaining data or even applied to other repositories with similar characteristics. It is worth noticing that this (arithmetic) function, which can be thought as a combination of several effective deduplication rules, is easy and fast to compute, allowing its efficient application to the deduplication of large repositories.

A function used for record deduplication must accomplish distinct but conflicting objectives: it should efficiently maximize the identification of record replicas while avoiding making mistakes during the process (e.g., to identify two records as replicas when they are not).

As shown in our experiments, our GP-based approach achieves better results than a state-of-the-art method (Bilenko and Mooney, 2003), based on Support Vector Machines, using less evidence to support it. Using less evidence to compute the similarity between two records improves the deduplication efficiency, since this reduces the number of attribute comparisons required.

In this thesis, we generalize our previous results (de Carvalho et al., 2006) by showing that our GP-based approach is also able to automatically find effective deduplication functions, even when the most suitable similarity function for each record attribute is not known in advance. This is extremely useful for the non-specialized user, who does not have to worry about selecting these functions for the deduplication task.

In addition, we show that our approach is also able to adapt the suggested deduplication function to changes on the replica identification boundary used to classify a pair of records as a match or not (de Carvalho et al., 2008a). This releases the user from the burden of having to choose and tune these values.

4.1 Modeling the Record Deduplication Problem with GP

When using GP (or even some other evolutionary technique) to solve a problem, there are some basic requirements that must be fulfilled, which are based on the data structure used to represent the solution (Koza, 1992). In our case, we have chosen a tree-based GP representation for the deduplication function, since it is a natural representation for this type of function (i.e., combination of numerical values). Based on this assumption, these requirements (Banzhaf et al., 1998) are the following:

1. All possible solutions to the problem must be represented by a tree, no matter its size.
2. The evolutionary operations applied over each individual tree must, at the end, result into a valid tree.
3. Each individual tree must be automatically evaluated.

For Requirement 1, it is necessary to take into consideration the kind of solution we intend to find. In the record deduplication problem, we look for a function that combines pieces of evidence.

In our approach, each piece of evidence (or simply “evidence”) E is a pair $\langle \textit{attribute}, \textit{similarity function} \rangle$ that represents the use of a specific similarity function over the values of a specific attribute found in the data being analyzed. For example, if we want to deduplicate a database table with four attributes (e.g., *forename*, *surname*, *address*, and *postalcode*) using a specific similarity function (e.g., the Jaro function Koudas et al. (2006)), we would have the following list of evidence: $E_1 \langle \textit{forename}, \textit{Jaro} \rangle$, $E_2 \langle \textit{surname}, \textit{Jaro} \rangle$, $E_3 \langle \textit{address}, \textit{Jaro} \rangle$, and $E_4 \langle \textit{postalcode}, \textit{Jaro} \rangle$.

For this example, a very simple function would be a linear combination such as $F_s(E_1, E_2, E_3, E_4) = E_1 + E_2 + E_3 + E_4$ and a more complex one would be $F_c(E_1, E_2, E_3, E_4) = E_1 * ((E_2^{E_3})/E_4)$.

To model such functions as a GP tree, each evidence is represented by a leaf in the tree. Each leaf (the similarity between two attributes) generates a normalized real number value (between 0.0 and 1.0). A leaf can also be a random number between 1.0 and 9.0, which is chosen at the moment that each tree is generated. Such leaves (random numbers) are used to allow the evolutionary process to find the most adequate weights for each evidence, when necessary. The internal nodes represent operations that are applied to the leaves. In our model, they are simple mathematical functions (e.g., $+$, $-$, $*$, $/$, exp) that manipulate the leaf values.

To enforce Requirement 2, the trees are handled by sub-tree atomic operations (Banzhaf et al., 1998) to avoid situations that could affect the integrity of the overall function. There cannot be neither a case where the value of a leaf node is replaced by the value of an internal node nor one where the value of an internal node is replaced by the value of a leaf node (Koza, 1992).

According to Requirement 3, all trees generated during a GP evolutionary process are tested against pre-evaluated data repositories where the replicas have been previously identified. This makes feasible to perform the whole process automatically, since it is possible to evaluate how the trees perform in the task of recognizing record pairs that are true replicas.

The tree input is a set of evidence instances, extracted from the data being handled, and its output is a real number value. This value is compared against a replica identification boundary value (see Section 2.1.2 for more details) as follows: if it is above the boundary, the records are considered replicas, otherwise, the records are considered distinct entries. It is important to notice that this classification enables further analysis, especially regarding the transitive properties of the replicas¹. This can improve the efficiency of clustering algorithms, since it provides not only an estimation of the similarity between the records being processed, but also a judgment of whether they are replicas or not.

¹If A is duplicate of B and B of C, then A should be a duplicate of C.

During the evolutionary process, a small sample of the repository data is used by our GP-based approach for training purposes. Each individual tree is evaluated on how it deduplicates the training data (by means of the combination of evidence it represents/encodes). The better the way the evidence available is combined, the more efficient the individual tree represents a suitable deduplication function.

However, there are several ways of combining evidence, thus leading to a huge search space for our GP-based approach to find the best solutions. For example, it is possible that two individual trees contain the same pieces of evidence, but lead to deduplication functions that produce different results. On the other hand, it is also possible that individual trees containing different pieces of evidence may lead to different deduplication functions that produce similar results.

At the end of the evolutionary process a set of individual trees is chosen after the evaluations at the last generation. This set of trees is tested against another sample of the repository data, in order to evaluate and verify their deduplication efficiency. The best individual tree, that represents the best deduplication function, is returned as the final solution.

In order to evaluate all individual tree evaluations (for both training and testing), we use a fitness function² that is described in the following section.

4.2 Fitness Function

As explained earlier, each individual tree represents a function that is used to find replicas in a repository. This function is applied to all record-to-record comparisons that take place during the deduplication. After doing these comparisons for all record pairs, the total number of correct and incorrect identified replicas is computed. This information is then used by the most important configuration component in our approach: the fitness function.

²A fitness function is an objective function that is used to evaluate the individuals along the evolutionary process (Banzhaf et al., 1998; Koza, 1992).

In the experiments presented in this chapter, we have used the F1 metric as our fitness function, since it harmonically combines the traditional precision and recall metrics commonly used for evaluating information retrieval systems (Baeza-Yates and Ribeiro-Neto, 1999), as follows:

$$Precision = \frac{NumberOfCorrectlyIdentifiedDuplicatedPairs}{NumberOfIdentifiedDuplicatedPairs}$$

$$Recall = \frac{NumberOfCorrectlyIdentifiedDuplicatedPairs}{NumberOfTrueDuplicatedPairs}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Here, this metric is used to express, as a single value, how well a specific individual performs in the task of identifying replicas. In summary, our GP-based approach tries to maximize these fitness values by looking for individuals that can make more correct decisions with fewer errors.

4.3 Complexity of the Training Phase

The time complexity of the training phase, based on our modeling, is $O(N_g \times N_i) \times T_e$, where N_g is the number of evolution generations, N_i is the number of individuals in the population pool, and T_e is the fitness evaluation complexity of an individual.

In our problem, the fitness evaluation complexity of an individual is the complexity of a single deduplication process given by $O(N_t)$, where N_t is the number of training pairs. A single deduplication process requires an all-against-all comparison of all records available in the repository being processed. Thus, the complexity of the training is given by $O(N_g \times N_i \times N_t)$. This is the worst case scenario for the training phase.

It is important to recall that the training time is not as important as the time to perform the actual deduplication with the suggested function, since the training phase is likely to be performed only once, and if eventually required, it can be done

off-line. The application of them suggested function is usually fast as it only requires the computation of simple arithmetic functions. Finally, it is worth noticing that in a real setting this training time can be even reduced with the application of *blocking* techniques (Elmagarmid et al., 2007).

4.4 Experiments

In this section, we present and discuss the results of the experiments performed to evaluate our proposed GP-based approach to record deduplication. There were three sets of experiments:

1. GP was used to find the best deduplication function for previously user-selected evidence, i.e., $\langle \textit{attribute}, \textit{similarity function} \rangle$ pair combinations specified by the user for the deduplication task. The use of user-selected evidence is a common strategy adopted by all previous record deduplication approaches (Bilenko and Mooney, 2003; Fellegi and Sunter, 1969; Tejada et al., 2002). Our objective in this set of experiments was to compare the evidence combination suggested by our GP-based approach with that of a state-of-the-art SVM-based solution, the Marlin system, used as our baseline.
2. GP was used to find the best deduplication function with automatically-selected evidence. Our objective in this set of experiments was to discover the impact on the result when GP has the freedom to choose the similarity function that should be used with each specific attribute.
3. GP was tested with different replica identification boundaries. Our objective in this set of experiments was to identify the impact on the resulting deduplication function when different replica identification boundary values are used with our GP-based approach.

4.4.1 Experimental Datasets

In our experiments, we used two real datasets commonly employed for evaluating record deduplication approaches (Tejada et al., 2002; Bilenko and Mooney, 2003), which are based on real data gathered from the Web. In addition, we created three additional datasets using a synthetic dataset generator.

The first real dataset, the Cora Bibliographic dataset, is a collection of 1295 distinct citations to 122 computer science papers taken from the Cora research paper search engine. These citations were divided into multiple attributes (*authornames*, *year*, *title*, *venue*, and *pagesandotherinfo*) by an information extraction system.

The second real dataset, hereafter named the Restaurants dataset, contains 864 entries of restaurant names and additional information, including 112 duplicates, that were obtained by integrating records from Fodor and Zagat’s guidebooks. We used the following attributes from this dataset: (restaurant) *name*, *address*, *city*, and *specialty*.

The synthetic datasets were created using the Synthetic Dataset Generator (SDG) (Christen, 2005) available in the Febrl package. Since real data is not easily available for experimentation, due to privacy and confidentiality constraints, we resorted to make use of synthetic datasets in order to conduct a larger set of experiments to evaluate our approach. SDG can create datasets containing names (based on frequency tables for given names and surnames), addresses (based on frequency tables for localities, postcodes, streets numbers, etc.), phone numbers, and personal number ids (like the social security number).

The attributes available from SDG are similar to those frequently found in personal medical data records. Accordingly, we have used it to create datasets containing only original records. Using the original records, SDG randomly generates duplicates of these records by modifying them (inserting, deleting or substituting characters, and swapping, removing, inserting, splitting or merging whole words), based on real error characteristics, and then inserting them in the original dataset. Some of these replicas are also created using look-up tables containing real world spelling variations for names

and addresses.

For our experiments, we created the synthetic datasets with the following attributes: *forename*, *surname*, *street number*, *address1*, *address2*, *suburb*, *postcode*, *state*, *date of birth*, *age*, *phone number*, and *social security number*. Each one of these datasets contains 2000 records in the total. They all present a different proportion between the number of original and replicated records, as well as a limit on the maximum number of replicas that an original record is allowed to have. Also, a certain number of modifications were applied to the candidate replicated records to simulate commonly found errors, such as typos. These errors are controlled at record and attribute levels. The datasets present the following characteristics:

- Dataset 1 – This dataset contains four files of 500 records (400 originals and 100 duplicates), with a maximum of five duplicates based on one original record (using a Poisson³ distribution of duplicate records), and with a maximum of two modifications in a single attribute and in the full record.
- Dataset 2 – This dataset contains four files of 500 records (350 originals and 150 duplicates), with a maximum of five duplicates based on one original record (using a Poisson distribution of duplicate records), and with a maximum of two modifications in a single attribute and four in the full record.
- Dataset 3 – This dataset contains four files of 500 records (300 originals and 200 duplicates), with a maximum of seven duplicates based on one original record (using a Poisson distribution of duplicate records), and with a maximum of four modifications in a single attribute and five in the full record.

In the first set of experiments, our intent was to present a comparison between the results of our GP-based approach with Marlin which is, as already mentioned, a state-of-the-art SVM-based system for record deduplication (Bilenko et al., 2003). The

³We adopted the Poisson distribution since the number of records in each file is relative small (Christen, 2005).

Marlin system has been implemented using an RBF kernel, tailored to the deduplication problem. We have adopted it as our baseline because, from the many possible SVM-based solutions for this problem, it is, for the best of our knowledge, the only concrete implementation that has been experimentally implemented and tested. As we are discussing here, the choice of the learning method is not the only issue determining the relative performance of a deduplication strategy; the way the problem is modeled within the learning framework chosen might even be more important. We also used the same real datasets presented in (Bilenko and Mooney, 2003) to compare the effectiveness of our GP-based approach with Marlin.

The second and third sets of experiments used the synthetic datasets. The use of synthetic datasets in these experiments made it possible to better evaluate the impact on the quality of the final solutions as a result of the changes in the record deduplication parameter setup we suggest (the pieces of evidence and identification boundaries), since the existing errors and their characteristics are known in advance.

The content of the real datasets used in our experiments was shuffled and divided into separate files. The Restaurant and Cora datasets were divided in four files. All experiments involved the following two steps:

1. The GP framework chooses one file for training purposes.
2. The GP framework tests the results of the training step in all remaining files.

The identification boundary value for the first experiment was empirically chosen after initial tuning tests and its value was set to 3.0 for both real datasets, since it appeared to ease the GP effort during the evolutionary process. The maximum F1 values reported in the first and second experiments were calculated as in (Bilenko et al., 2003). For the remaining experiments, we present the mean and standard deviation values after ten runs, using a 4-fold cross validation. All comparisons consider a t-test with a p-value of 0.05.

It is important to notice that, in all experiments conducted, we do not use any blocking technique (Sarawagi and Bhamidipaty, 2002). Blocking techniques are used to deal with the high computational cost of the comparisons between the records that are required by the deduplication task. We adopted this experimental strategy to avoid any distortions in the recall value that could occur since blocking techniques split the datasets into smaller blocks. These smaller blocks may not have all replicas and originals of one record grouped in the same block, since the blocking method may fail in the grouping task, thus affecting the recall measure. By doing this, we intend to present an evaluation of our GP-based approach that is free of any errors originated from prior steps such as cleaning, standardization and blocking (Sarawagi and Bhamidipaty, 2002) in the record deduplication process.

In all sets of experiments, we used a workstation with the following hardware and software configuration: Pentium Core 2 Duo Quad (2 Ghz) processor with 4 GB RAM DDR2 memory and 320 GB SATA hard drive, and running a 64-Bits FreeBSD 7.1 Unix-based operational system.

4.4.2 Experiments with User-Selected Evidence

In this first set of experiments, we adopted a user-selected evidence strategy, i.e., we combined each dataset attribute with a previously user-selected similarity function. As mentioned before, using user-selected similarity functions for each attribute is a strategy commonly adopted by all previous works. The similarity functions chosen were string distance (Levenshtein distance) and cosine similarity (SoftTFIDF similarity), since we intended to compare our results with the ones presented in (Bilenko and Mooney, 2003), that use these same similarity functions. We refer the reader to (Koudas et al., 2006), for more information on the similarity functions used in this chapter.

Additionally, we also conducted experiments with the Marlin system to evaluate the discriminative characteristics of each possible attribute combination of the Cora and Restaurants datasets using the cosine similarity (SoftTFIDF similarity) function,

Table 4.1. Discriminative Attribute Selection - Marlin - Cora Database

Attributes	F1 (σ)	Attributes	F1 (σ)	Attributes	F1 (σ)
A	0.496 \pm (0.069)	AVO	0.561 \pm (0.074)	TVYO	0.878 \pm (0.038)
AO	0.781 \pm (0.233)	AVYO	0.612 \pm (0.100)	TY	0.853 \pm (0.049)
AT	0.769 \pm (0.069)	AY	0.373 \pm (0.069)	TYO	0.878 \pm (0.039)
ATO	0.821 \pm (0.056)	AYO	0.547 \pm (0.101)	V	0.480 \pm (0.068)
ATV	0.488 \pm (0.113)	O	0.653 \pm (0.066)	VO	0.576 \pm (0.060)
ATVO	0.838 \pm (0.049)	T	0.847 \pm (0.046)	VY	0.350 \pm (0.045)
ATVY	0.823 \pm (0.057)	TY	0.866 \pm (0.040)	VYO	0.397 \pm (0.070)
ATY	0.796 \pm (0.063)	TV	0.790 \pm (0.093)	Y	0.048 \pm (0.010)
ATYO	0.836 \pm (0.053)	TVO	0.867 \pm (0.042)	YO	0.376 \pm (0.054)
AV	0.484 \pm (0.067)	TVY	0.836 \pm (0.081)	ATVYO	0.861 \pm (0.046)

Table 4.2. Discriminative Attribute Selection - Marlin - Restaurants Database

Attributes	F1 (σ)	Attributes	F1 (σ)	Attributes	F1 (σ)
A	0.567 \pm (0.067)	CS	0.007 \pm (0.008)	NAS	0.897 \pm (0.042)
AC	0.561 \pm (0.065)	N	0.791 \pm (0.045)	NC	0.806 \pm (0.053)
ACS	0.877 \pm (0.212)	NA	0.896 \pm (0.046)	NCS	0.802 \pm (0.052)
AS	0.596 \pm (0.078)	NAC	0.900 \pm (0.042)	NS	0.789 \pm (0.046)
C	0.003 \pm (0.002)	NACS	0.898 \pm (0.041)	S	0.003 \pm (0.003)

since it was the most effective similarity function on both datasets. The F1 average and standard deviation (σ) results (from 30 runs) are presented in Tables 4.1 and 4.2. In Table 4.1, in the Attributes column, A stands for *authornames*, Y for *year*, T for *title*, V for *venue*, and O for *pagesandotherinfo*. In Table 4.2, N stands for *name*, A for *address*, C for *city*, and S for *specialty*.

Observing Table 4.1, the Marlin system found that the most discriminative attribute set for the Cora dataset comprises all attributes but one, *authornames*. However, this result (TVYO) is statistically equivalent to that using all available attributes (ATVYO). Regarding the Restaurants dataset, we can see in Table 4.2 that the most discriminative attribute set includes *name*, *address* and *city* (NAC), i.e., all attributes but one, *specialty*. Similarly to the former experiment, the most discriminative attribute set presents results statistically equivalent to the one using all attributes (NCAS).

Column “Value (1)” in Table 4.3 shows the parameter setup applied to our GP framework for this first set of experiments. This parameter setup was chosen after initial tuning experiments - it provided good results and, at the same time, avoided problems with overfitting and extensive training time requirements. Table 4.4 shows

Table 4.3. GP Parameters: (1) Previously Fixed Evidence (2) Previously Fixed Evidence vs Not Previously Fixed Evidence using Synthetic Datasets

Parameter	Value (1)	Value (2)
Max Number of Generations	20	30
Population	100	60
Initial Random Population Method	Full Depth	Full Depth
Reproduction Rate	20%	30%
Selection	Roulette Wheel	Roulette Wheel
CrossOver Rate	80%	70%
CrossOver Method	Random SubTree Exchange	Random SubTree Exchange
Mating (Pairing)	Random	Random
Mutation Rate	2%	2%
Mutation Operation	Random SubTree Insertion	Random SubTree Insertion
Max Random Tree Depth	5	5

Table 4.4. User-Selected Evidence - F1 Averages

Dataset	Similarity Metric	GP F1 (σ)	Marlin F1 (σ)
Cora	String Distance	0.900 \pm (0.010)	0.820 \pm (0.020)
	Cosine Similarity	0.880 \pm (0.004)	0.870 \pm (0.030)
Restaurants	String Distance	0.980 \pm (0.010)	0.900 \pm (0.020)
	Cosine Similarity	0.980 \pm (0.020)	0.900 \pm (0.020)
Synthetic 3	String Distance	0.960 \pm (0.020)	0.960 \pm (0.030)
	Cosine Similarity	0.970 \pm (0.010)	0.960 \pm (0.020)

the results obtained for the three datasets, Cora, Restaurants, and Synthetic 3, the last one the “dirtiest” of the synthetic datasets we created for our experiments. We present the F1 average and the standard deviation (σ) for both approaches. As we can see, the deduplication function suggested by our GP-based approach outperformed Marlin (the baseline) in the Cora dataset by 9.76% when we used the string distance function and there was a statistic tie when we used the cosine similarity function. In the Restaurants dataset, the function suggested by our GP-based approach outperformed the baseline by 8.88% when we used both functions. The results with the Synthetic 3 dataset were statistically equivalent for both approaches using both similarity functions. Improvements in the F1 values beyond the results presented here are difficult, since these datasets are reasonably easy to handle. Considering that most of the replicas present a well-manned pattern of errors when compared with the originals, both approaches have no difficulty to find the same replicated entries, thus leading to similar performances.

The best tree (the best individual extracted from the last generation) for the ex-

periment with the Cora dataset, when we used the string distance similarity function, was

$$GPCoraStrDist() = ((e + d) * d) + (c + (a * 2)) + d$$

where a , c , d , and e correspond to evidence defined on the attributes *authornames*, *title*, *venue*, and *pagesandotherinfo*, respectively.

In this function, the string distance similarity function was applied to four out of five attributes. Only the attribute *year* was not used. This happened because the cosine similarity function, when applied to dates, is not able to properly measure the “distance” between them. By analyzing the function, we can say that the attributes *venue* (used in evidence d) and *authornames* (used in evidence a) are those that play the most important role for identifying replicas in the Cora dataset, since they have higher weights, i.e., evidence a is multiplied by 2 and evidence d appears three times.

The best tree for the experiment with the Restaurants dataset, when we used the cosine similarity function, was

$$GPRestaurantCosine() = ((b + (b + d)) * a) + 2$$

where a , b , and d correspond to evidence defined on the attributes *name*, *address*, and *specialty*, respectively.

In this function, we notice that only the attribute *city* was not used as evidence. It is also important to notice that *address* (used in evidence b) was the attribute which received more weight in the function. This may be explained because the cosine similarity function is more discriminative when applied to long multiple string attributes, like addresses, than when applied to short strings (e.g., dates or short names).

4.4.3 Experiments with Automatically-Selected Evidence

In this second set of experiments, the evidence is not user-selected, since each attribute could be freely paired with any of the following similarity functions: string distance (Levenshtein distance), cosine similarity (SoftTFIDF similarity), SortWinkler, Winkler or Jaro. This is one of the main contributions of our work that distinguishes it from

Table 4.5. Cora and Restaurant Datasets - Automatically-Selected Evidence Strategy

Dataset	Method	Training F1 (σ)	Test F1 (σ)
Cora	GP User-Selected StrDist	0.900 \pm (0.010)	0.890 \pm (0.020)
	GP User-Selected CosSim	0.880 \pm (0.010)	0.880 \pm (0.020)
	GP Automatically-Selected	0.880 \pm (0.030)	0.870 \pm (0.020)
	GP Automatically-Selected+	0.900 \pm (0.010)	0.910 \pm (0.010)
Restaurants	GP User-Selected StrDist	1.000 \pm (0.000)	0.982 \pm (0.020)
	GP User-Selected CosSim	1.000 \pm (0.000)	0.981 \pm (0.020)
	GP Automatically-Selected	0.990 \pm (0.001)	0.970 \pm (0.020)
	GP Automatically-Selected+	1.000 \pm (0.000)	0.980 \pm (0.010)

Table 4.6. Training and Test Times for Cora and Restaurants Datasets - Automatically-Selected Evidence Strategy

Dataset	Method	Training (min) (σ)	Test (min) (σ)
Cora	GP User-Selected StrDist	1209 \pm (10)	49 \pm (1)
	GP User-Selected CosSim	685 \pm (3)	30 \pm (1)
	GP Automatically-Selected	1028 \pm (9)	43 \pm (2)
	GP Automatically-Selected+	2383 \pm (10)	45 \pm (3)
Restaurants	GP User-Selected StrDist	276 \pm (5)	7 \pm (1)
	GP User-Selected CosSim	201 \pm (4)	6 \pm (1)
	GP Automatically-Selected	266 \pm (5)	5 \pm (1)
	GP Automatically-Selected+	302 \pm (5)	7 \pm (1)

all previous proposals in the literature, which use user-selected evidence. For this, we compared the F1 levels achieved with the best results obtained in the previous experiment using user-selected evidence. The GP parameters were the same used in the first set of experiments (see column “Value (1)” in Table 4.3) to provide a standard setup for the comparison between the different strategies (user-selected evidence and automatically-selected evidence).

Table 4.5 shows the results of the training and test phases (F1 averages and standard deviations). The training and test running times are presented in Table 4.6. In both tables, *GP User-Selected StrDist* means our user-selected evidence strategy using the string distance function, *GP User-Selected CosSim* our user-selected evidence strategy using the cosine similarity function, and *GP Automatically-Selected* our automatically selected evidence strategy. As we can see, the F1 levels achieved by *GP User-Selected StrDist*, *GP User-Selected CosSim* and *GP Automatically-Selected* are similar, and all our strategies outperformed our baseline in these two datasets, as in the previous set of experiments.

Also, it is important to observe that the training and test results are very similar. This shows that our random data sampling, used to create the training data folds, was able to capture the characteristics of the test data. Moreover, this also shows that the suggested functions successfully generalized for the test data in these experiments and that the provided solutions were not overfitted to the training data.

Regarding the training and test running times presented in Table 4.6, we would like to point out that the training is performed using a small part of the available data and the test is executed using the remaining. Also, as explained in the beginning of this section, we are not adopting any blocking technique to avoid distortions in the recall values and, consequently, in the F1 values. With respect to the training and test times, the Cora dataset demands more time for both phases, since it has more entries and, also, more raw data (long string chains). The Restaurants dataset is smaller than the Cora one and also contains small strings. Thus, it requires less processing time, particularly when using highly demanding similarity functions, such as that based on the Levenshtein algorithm.

For the experiments with our GP-based approach, the best tree for the experiment with the Cora dataset, when we used the automatically-selected evidence strategy, was

$$GPCoraMix() = (m) + ((p) + (2 * g)).$$

where m , p , and g correspond, respectively, to the following list of evidence: $\langle venue, Winkler \rangle$, $\langle pagesandotherinformation, Jaro \rangle$, and $\langle year, string\ distance \rangle$.

In this function, only three out of the five attributes were used. One possible explanation for this is the flexibility given to the evolutionary process to suggest the best combination based on the most suitable evidence. Like in the first set of experiments, the attribute *venue* also composes this function. However, although the attribute *year* is not part of the function suggested in the previous experiment, in the function *GPCoraMix* the evidence $\langle year, string\ distance \rangle$ received more weight than any other one.

The best tree for the experiment with the Restaurants dataset, when we used the

automatically-selected evidence strategy, was

$$GPRestaurantMix() = 4 * d$$

where d corresponds to the evidence $\langle name, SortWinkler \rangle$.

In this function, evidence based only on the attribute *name* was considered sufficient to deduplicate the Restaurants dataset. The weight given to this evidence may be interpreted as a way to raise the function value in order to fit the identification boundary.

We notice that using the automatically-selected evidence strategy, the search space was much larger due to several possible evidence combinations, making it more difficult to find better solutions. However, we kept the same GP parameter setup in this experiment to allow a fair comparison between the results of the two strategies.

On the other hand, to compensate for the larger search space, we run the automatically-selected evidence strategy with an enhanced GP parameter setup: increasing the population size (including 20 new individuals) and raising the number of generations (adding 10 more cycles) during the training phase. The results of the experiment using our automatically selected strategy with this setup (labeled as *GP Automatically-Selected+* in Table 4.5) was able to match the results of the previous experiments using user-selected evidence strategy.

In order to confirm this behavior observed with real data, we conducted additional experiments using our synthetic datasets. The user-selected evidence setup used in this experiment was built using the following list of evidence: $\langle forename, jaro \rangle$, $\langle surname, jaro \rangle$, $\langle street\ number, string\ distance \rangle$, $\langle address1, jaro \rangle$, $\langle address2, jaro \rangle$, $\langle suburb, jaro \rangle$, $\langle postcode, string\ distance \rangle$, $\langle state, jaro \rangle$, $\langle date\ of\ birth, string\ distance \rangle$, $\langle age, string\ distance \rangle$, $\langle phone\ number, string\ distance \rangle$, $\langle social\ security\ number, string\ distance \rangle$. This list of evidence, using the jaro similarity function for free text attributes and a string distance function for numeric attributes, was chosen since it required less time to be processed in our initial tuning tests.

The setup of the automatically-selected evidence used is suggested by our GP-

Table 4.7. Synthetic Datasets - Automatically-Selected Evidence Strategy

Dataset	Method	(1)Avg (σ)	(2)Avg (σ)	(3)Avg (σ)	(4)Avg (σ)
Synthetic 1	GP User-Selected	0.987 \pm (0.028)	0.945 \pm (0.033)	0.958 \pm (0.027)	0.945 \pm (0.044)
	GP Automatically-Selected	0.968 \pm (0.038)	0.948 \pm (0.065)	0.955 \pm (0.046)	0.936 \pm (0.087)
	GP Automatically-Selected+	0.993 \pm (0.010)	0.979 \pm (0.028)	0.983 \pm (0.032)	0.982 \pm (0.024)
Synthetic 2	GP User-Selected	0.967 \pm (0.050)	0.935 \pm (0.067)	0.943 \pm (0.047)	0.951 \pm (0.045)
	GP Automatically-Selected	0.933 \pm (0.054)	0.929 \pm (0.045)	0.923 \pm (0.054)	0.929 \pm (0.049)
	GP Automatically-Selected+	0.978 \pm (0.025)	0.977 \pm (0.013)	0.967 \pm (0.020)	0.975 \pm (0.020)
Synthetic 3	GP User-Selected	0.959 \pm (0.056)	0.928 \pm (0.071)	0.956 \pm (0.045)	0.953 \pm (0.064)
	GP Automatically-Selected	0.948 \pm (0.067)	0.955 \pm (0.064)	0.960 \pm (0.048)	0.956 \pm (0.053)
	GP Automatically-Selected+	0.995 \pm (0.004)	0.996 \pm (0.005)	0.988 \pm (0.008)	0.991 \pm (0.008)

Table 4.8. Training and Test Times for Synthetic Datasets - Automatically-Selected Evidence Strategy

Dataset	Method	Training (min) (σ)	Test (min) (σ)
Synthetic 1	GP User-Selected	495 \pm (2)	3 \pm (0.300)
	GP Automatically-Selected	1070 \pm (2)	2 \pm (0.100)
	GP Automatically-Selected+	1885 \pm (2)	2 \pm (0.700)
Synthetic 2	GP User-Selected	509 \pm (3)	3 \pm (0.300)
	GP Automatically-Selected	120 \pm (2)	2 \pm (0.090)
	GP Automatically-Selected+	2102 \pm (4)	2 \pm (0.800)
Synthetic 3	GP User-Selected	629 \pm (3)	2 \pm (0.0100)
	GP Automatically-Selected	1312 \pm (2)	3 \pm (0.100)
	GP Automatically-Selected+	2665 \pm (5)	4 \pm (0.500)

based approach. Like in the previous experiments in this section, the aforementioned attributes of the synthetic datasets could be freely combined with the following similarity functions: string distance (Levenshtein distance), cosine similarity (SoftTFIDF similarity), bigram, Winkler or Jaro. The GP parameters used in this experiment (see column “Value (2)” in Table 4.3) were chosen after initial tuning experiments - they provided good results and, at the same time, avoided problems with overfitting and extensive training time requirements.

The results of the experiments with the synthetic datasets are presented in Table 4.7. The result of the best individual from the training phase is labeled (1), for both the F1 average (Avg) and standard deviation (σ). The results obtained with the best individual from the training applied to the test datasets are presented with labels (2), (3) and (4) (averages and standard deviations), respectively.

As we can notice, the F1 levels achieved by using the user-selected evidence and the automatically-selected evidence strategies are again similar (averages and standard deviations). These results confirm the previous observed behavior when the real datasets

were used. This demonstrates that our GP-based approach is also able to find suitable deduplication functions when the set of evidence is not previously fixed by the user and, therefore, frees the non-specialized user from the burden of having to choose which similarity function should be paired with each specific attribute.

However, despite the good results achieved with the same parameter setup used in the user selected evidence experiment, we conducted a second run of the automatically-selected evidence experiment with an enhanced GP parameter setup (results presented as “GP Automatically-Selected+” in Table 4.7). Due to the larger search space, we have increased the number of generations and the population size from 30 and 50 to 50 and 80, respectively.

The results show that providing the adequate time (number of generations) and resources (population size) to our GP-based approach, it can produce better solutions than the both previous results (user-selected and automatically-selected evidence strategies) obtained in the experiments for which we kept the same GP parameter setup. The training and test times for all synthetic datasets, shown in Table 4.8, are quite similar, since they present the same characteristics (i.e., number of attributes, number of entries, etc.).

Observing the results of this second set of experiments, we also notice that all suggested functions use less evidence than the total number available. One explanation for this behavior is that, for a given dataset, some attributes can be more discriminative than others when combined with a specific similarity function. This means that our suggested solutions would run faster than solutions provided by other record deduplication approaches that use all available evidence. Moreover, our GP-based approach is able to find deduplication functions even when each evidence component $\langle \textit{attribute}, \textit{similarity function} \rangle$ is not fixed a priori. This is useful for the non-specialized user, because it does not require previously knowledge to setup such parameters.

4.4.4 Experiments with the Replica Identification Boundary

A critical aspect regarding the effectiveness of several record deduplication approaches is the setup of the boundary values that classify a pair of records as replicas or not with respect to the results of the deduplication function.

In this final set of experiments, our objective was to study the ability of our GP-based approach to adapt the deduplication functions to changes in the replica identification boundary, aiming at discovering whether it is possible to use a previously fixed (or suggested) value for this parameter.

To evaluate how much effort is spent by the evolutionary process in order to adapt the deduplication function to different boundary values, we devised the following strategy for the first experiment:

1. Fix the F1 level that should be achieved by the suggested functions.
2. Record the number of generations required by the evolutionary process to suggest a function that reaches this F1 level objective.

We fixed the F1 level objective as 95% of the best results achieved during the training phase in the first set of experiments conducted using the user-selected evidence setup (since it provides a smaller search space) to force the evolutionary process to suggest feasible functions. Thus, the F1 levels used were: 0.95 for the Restaurants dataset and 0.84 for the Cora dataset (see Table 4.5).

To better control the experiment execution time, we chose the same parameter setup used in the experiment with user-selected evidence strategy, with the cosine similarity function being applied to both datasets, since it is faster to execute than the string distance function. Each experiment run used a different identification boundary value, as shown in Table 4.9.

The results in Table 4.9 show the number of generations that were required, in each experiment run, for the deduplication functions to reach the fixed F1 levels. The table also shows that our GP-based approach always generates functions that are able

Table 4.9. Number of generations required to reach F1 above 95%

Dataset	Boundary Value	2	3	5	10	15
Cora	Generations	3	3	4	10	13
Restaurants		2	2	2	9	12

Table 4.10. GP Parameters: (1) Previously Fixed Evidence (2) Previously Fixed Evidence vs Not Previously Fixed Evidence using Synthetic Datasets

Parameter	Value (1)	Value (2)
Max Number of Generations	20	30
Population	100	60
Initial Random Population Method	Full Depth	Full Depth
Reproduction Rate	20%	30%
Selection	Roulette Wheel	Roulette Wheel
CrossOver Rate	80%	70%
CrossOver Method	Random SubTree Exchange	Random SubTree Exchange
Mating (Pairing)	Random	Random
Mutation Rate	2%	2%
Mutation Operation	Random SubTree Insertion	Random SubTree Insertion
Max Random Tree Depth	5	5

to reach the fixed F1 levels, independently of the boundary value used. However, we notice that the bigger the boundary value, the more the number of generations that are necessary to reach the required F1 level.

To better understand the reason for the increase on the number of generations as the boundary value increases, we devised an additional experiment using the synthetic datasets. The strategy for this second experiment was:

1. Fix a specific GP parameter setup for the experiments.
2. Execute the experiments varying only the replica identification boundary in each run in order to observe how the evolutionary process behaves as the boundary value is changed.

The parameters used in the experiment for the real and synthetic datasets are those presented in Table 4.10, column “Value (1)” and “Value (2)”, respectively. These experiments were conducted on both types of dataset, in order to observe how the evolutionary process would deal with datasets presenting different characteristics.

The results are presented in Table 4.11 for the Cora and Restaurants datasets, and in Table 4.12 for the synthetic datasets. The results of the best individual from the

Table 4.11. F1 Results: Identification Boundary - Cora and Restaurant Datasets

Dataset	Boundary	(1)Avg (σ)	(2)Avg (σ)	(3)Avg (σ)	(4)Avg (σ)
Cora	1	0.805 \pm (0.039)	0.826 \pm (0.039)	0.834 \pm (0.054)	0.823 \pm (0.054)
	5	0.788 \pm (0.052)	0.821 \pm (0.055)	0.820 \pm (0.046)	0.801 \pm (0.074)
	10	0.750 \pm (0.130)	0.784 \pm (0.098)	0.795 \pm (0.127)	0.770 \pm (0.119)
	20	0.613 \pm (0.269)	0.636 \pm (0.265)	0.645 \pm (0.271)	0.613 \pm (0.275)
	30	0.632 \pm (0.194)	0.656 \pm (0.195)	0.656 \pm (0.200)	0.648 \pm (0.180)
	50	0.512 \pm (0.268)	0.531 \pm (0.271)	0.554 \pm (0.282)	0.537 \pm (0.279)
	100	0.577 \pm (0.218)	0.586 \pm (0.242)	0.611 \pm (0.233)	0.579 \pm (0.224)
Restaurants	1	1.0 \pm (0.0)	0.823 \pm (0.0)	0.941 \pm (0.0)	0.846 \pm (0.033)
	5	0.957 \pm (0.135)	0.795 \pm (0.128)	0.913 \pm (0.086)	0.784 \pm (0.140)
	10	0.924 \pm (0.209)	0.731 \pm (0.264)	0.836 \pm (0.294)	0.741 \pm (0.268)
	20	0.874 \pm (0.202)	0.776 \pm (0.063)	0.874 \pm (0.115)	0.726 \pm (0.182)
	30	0.685 \pm (0.294)	0.536 \pm (0.245)	0.610 \pm (0.281)	0.425 \pm (0.382)
	50	0.752 \pm (0.311)	0.643 \pm (0.254)	0.724 \pm (0.288)	0.571 \pm (0.367)
	100	0.646 \pm (0.403)	0.584 \pm (0.300)	0.609 \pm (0.383)	0.567 \pm (0.368)

Table 4.12. F1 Results: Identification Boundary using Synthetic Repositories - Synthetic Datasets

Dataset	Boundary	(1)Avg (σ)	(2)Avg (σ)	(3)Avg (σ)	(4)Avg (σ)
Synthetic 1	1	0.997 \pm (0.005)	0.960 \pm (0.024)	0.983 \pm (0.012)	0.970 \pm (0.026)
	5	0.987 \pm (0.028)	0.945 \pm (0.033)	0.958 \pm (0.027)	0.945 \pm (0.044)
	10	0.974 \pm (0.048)	0.937 \pm (0.073)	0.955 \pm (0.045)	0.947 \pm (0.075)
	20	0.910 \pm (0.071)	0.898 \pm (0.086)	0.897 \pm (0.087)	0.881 \pm (0.077)
	30	0.906 \pm (0.037)	0.826 \pm (0.191)	0.911 \pm (0.034)	0.858 \pm (0.079)
	50	0.919 \pm (0.056)	0.868 \pm (0.087)	0.916 \pm (0.051)	0.889 \pm (0.052)
	100	0.896 \pm (0.083)	0.835 \pm (0.100)	0.880 \pm (0.101)	0.872 \pm (0.080)
Synthetic 2	1	0.970 \pm (0.046)	0.940 \pm (0.066)	0.950 \pm (0.039)	0.956 \pm (0.041)
	5	0.967 \pm (0.050)	0.935 \pm (0.067)	0.943 \pm (0.047)	0.951 \pm (0.045)
	10	0.930 \pm (0.081)	0.920 \pm (0.077)	0.919 \pm (0.084)	0.923 \pm (0.079)
	20	0.834 \pm (0.088)	0.813 \pm (0.082)	0.825 \pm (0.110)	0.845 \pm (0.088)
	30	0.857 \pm (0.110)	0.789 \pm (0.118)	0.845 \pm (0.106)	0.844 \pm (0.089)
	50	0.876 \pm (0.087)	0.833 \pm (0.070)	0.871 \pm (0.066)	0.859 \pm (0.084)
	100	0.823 \pm (0.113)	0.809 \pm (0.088)	0.823 \pm (0.108)	0.812 \pm (0.121)
Synthetic 3	1	0.987 \pm (0.012)	0.966 \pm (0.026)	0.971 \pm (0.022)	0.978 \pm (0.022)
	5	0.959 \pm (0.056)	0.928 \pm (0.071)	0.956 \pm (0.045)	0.953 \pm (0.064)
	10	0.881 \pm (0.129)	0.838 \pm (0.146)	0.876 \pm (0.126)	0.883 \pm (0.120)
	20	0.882 \pm (0.114)	0.856 \pm (0.111)	0.882 \pm (0.105)	0.879 \pm (0.125)
	30	0.872 \pm (0.089)	0.828 \pm (0.088)	0.865 \pm (0.076)	0.885 \pm (0.073)
	50	0.864 \pm (0.079)	0.832 \pm (0.069)	0.878 \pm (0.062)	0.877 \pm (0.064)
	100	0.828 \pm (0.069)	0.787 \pm (0.084)	0.861 \pm (0.047)	0.829 \pm (0.078)

training phase is labeled (1) and those obtained in the test phase are labeled (2), (3) and (4) respectively, all of them with average (Avg) and standard deviation (σ) values.

As we can see from these tables, the deduplication functions found when using smaller identification boundaries present higher F1 average and smaller standard deviation values, in both the training and testing phases. The increase on the value of the

Table 4.13. Boundary Variation - Tree Adaptation during Training - Cora Dataset

Boundary Value	F1 Value	Tree Coding
1	0.841	$+ * * c * * ca + cd + de * * ac * ee$
1	0.822	$* * ac + + cd * d * ec$
5	0.819	$* + + ae - 7 - / add * * cd / ad$
5	0.811	$- * + a6c - / ba + b - dc$
10	0.803	$* c * * * + cc + + * ce * 2b + - 4c * b * * b * caac * 3a$
10	0.781	$/ * * + * + * abaca - b * a + dac / - * aca * a + da$
20	0.785	$* - * ac * - da * d - ea - * 8 - 5 * ad * ab$
20	0.690	$+ + - b + c + cc * + cc + 6d * + * * * ae + be +$ $- aa + cac + 6e$
30	0.724	$* / * + + e8 - bc - c - - bdd - aa / - + ae / 1c / 1 / 1c$
30	0.691	$* / - / a8 / b1b1 - / bbc + / e8 + / / - c * 7 * cd -$ $/ bbc8 - db$
50	0.694	$- * + d + ed * * a77 + - ee + e * 47$
50	0.692	$/ + * ae * / ab + * * e7 + bc + + * * e7 + bc +$ $- a1 - / eb * ea + + bcc + * eac - / aac$
100	0.709	$/ - / a + ec - e / + 4 + + 41e - e1 * / + 4c / +$ $4e + ec - - e1 - ec$
100	0.782	$- / * 8 * * ad7 - 1 * c1 + / 8c - c1$

Table 4.14. Boundary Variation - Tree Adaptation during Training - Restaurants Dataset

Boundary Value	F1 Value	Tree Coding
1	1.0	$/ - 8 / 2a - * 2a / 2a$
1	1.0	$* + a - aa1$
5	1.0	$/ + * a * * add * + eed / e + * a * da * * daa$
5	0.909	$+ * + da4 - - + aa + 1e + d1$
10	1.0	$+ * + dd * a * + db * a * a5 - * d5 * ca$
10	0.888	$+ * + d * cd * a5 - * d * + * d5ba * bb$
20	0.981	$+ * + + + gbek + gf * + 9k + fl$
20	0.976	$+ * + + gek + ef * + 9k + fl$
30	1.0	$* / - a * / e - / * 1a * aa / * - ac / ce / - ce + 3da - ab$
30	0.750	$+ a / / e + b + 8a + a + 8 / e + b + 8 * ea + * ea +$ $+ / 7c * e + e6 - + d / - ba - ba * e4 - / - ba - baa$
50	1.0	$/ - + a / d + b / / / ec - 9b * + ed + 1b + cb - + ab + 1b$
50	0.888	$- / * / - dd - * * ccca - * dca - * b * ba / dd * / - cadc$
100	1.0	$/// b / / + cc + cc + // bc / + /// c + bebc1ca +$ $cb / - a11 // bc / b / e / + /// c + beb + cc1 + b / c + cc$
100	0.888	$- / / a + dc - a / / a + dc / a + dc / + cc / a / a -$ $/ aab / a + dc$

Table 4.15. Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 1

Boundary Value	F1 Value	Tree Coding
1	1.0	<i>*g * i + ** fdda</i>
1	1.0	<i>- * + ci * i * + * kgg * al / * cl + + ihc</i>
5	1.0	<i>+ + *ad + hk + + fgh</i>
5	1.0	<i>* + *bf + f + *bfk - +ii - ig</i>
10	0.991	<i>+ + + i/bl/fl + *ie * 8l</i>
10	0.985	<i>* + + *bg + *bgb + +gb8l</i>
20	0.976	<i>/ * - - h/hlh * *a + h + a/al + aa - a/all</i>
20	0.994	<i>+ + - + -dh + j + j + ddl + l + dd * +li + h8</i>
30	0.934	<i>// - * *kk * /kg/kg - * *kkk/kgg - *kk/kg</i>
30	0.988	<i>* * *b * ll - 6b + - * d * *ll - 6bc * g9</i>
50	0.925	<i>* // /d/d - dj - //jcdj - gl0//g8/df</i>
50	0.957	<i>* * *f4 - 6i + * + *l0iii + gi</i>
100	0.934	<i>* + - hgg// - gg8 - kg</i>
100	0.900	<i>- / - i - h / - 1i - h * hl0 - h * hl0 * /gc - *hl0i</i>

Table 4.16. Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 2

Boundary Value	F1 Value	Tree Coding
1	0.988	<i>*l * i + f + *bg - lf</i>
1	0.996	<i>*f * i + f + *b + l + *ll - * * *lg + b - k f f f - lf</i>
5	0.979	<i>* * *gd * fl + +9ak</i>
5	0.998	<i>* + *kl + fi + +fl * jb</i>
10	0.975	<i>* + + *dbi * a * jh + g * k5</i>
10	0.998	<i>* * +jd + f * ib + *l * l5 * * + fk + f * bdk</i>
20	0.908	<i>/ * - + +gk - -k33g - k / - ek - -k / - k3 - kk + +gk - k3</i>
20	0.960	<i>+ / - * - -cdd0d - *ldd//dh * /h - cdd</i>
30	0.864	<i>+ * * + / - cc - cie * ff * +h - cc f / - c + +h * + - +eeie * +f - cc * + / - cc - cie * fff - *f * +h - ccff</i>
30	0.967	<i>//l * -1l * 96 * -1l - 1l</i>
50	0.989	<i>* / + /l - l * ll * + /l - l * ll * *ib * +3 * + /d - l * ll * *ib * +3 * ca * gla * dll - i * li * *kb * dg</i>
50	0.933	<i>+ - // + jd - /jljl/ek - // + gg - /jljl/ek</i>
100	0.961	<i>//h / - *l//hee//heee - *l * ll</i>
100	0.947	<i>+ // // //c3 - f * *bf /lbll - f * lf * - - f * lff * + / - del * -d //ll - f * *blf - 3 + / - de + /lb/cl * -d //fl - f * *blf /lb/cl</i>

Table 4.17. Boundary Variation - Tree Adaptation during Training - Synthetic Dataset 3

Boundary Value	F1 Value	Tree Coding
1	0.987	$/ *** lll + l **** / lkl + l * ll * lg / lk$
1	0.966	$* + *ij * + *** iic * idi * *dbd$
5	0.966	$+ * +dl + hf + * + f * fjll$
5	0.984	$+ * +dl + hl + a * +aaf$
10	0.976	$- * +b * a + gk * 6i/a6$
10	0.976	$- * +b * a + gk * 6i/b6$
20	0.981	$+ * + + +gbek + gf * +9k + fl$
20	0.976	$+ * + + gek + ef * +9k + fl$
30	0.980	$*f/-1-// - //gc-6i+dkc-6i*+ek+ld-1l$
30	0.985	$*f/-g--f-l-//gc-6-4e*+ek+id*+ek+ie-1$
50	0.965	$/**e*--+bak*l*--+g*lf*j*e*--+baa*l*--+*li*lf*jg*b/*lf-*lff*b/*lf-*lffg-*lff$
50	0.962	$/ **f * - + bak * l * - + li * lf - d * ja *b / * lf - * lffg - * lff$
100	0.937	$//b - /bbl - /bbl$
100	0.938	$+1//b - l1 - l1 - l - aa$

boundary led to smaller F1 values and to unstable standard deviation values, as can be observed in the results for the testing phase.

A possible explanation for this behavior can be drawn by the following facts:

1. The replica identification boundary is always a positive value.
2. The values of the evidence instances (the result of applying a string function to an attribute pair) vary from 0.0 to 1.0.
3. In the case of a perfect match for all attributes, the summation of all evidence instance values would be equal to the number of attributes used as evidence and their total multiplication would be equal to 1.
4. Not all attribute pairs must reach a perfect match in order to be considered a replica.

Our GP-based approach tries to combine distinct evidence to maximize the fitness function results, and one major factor that might impact the results is the replica

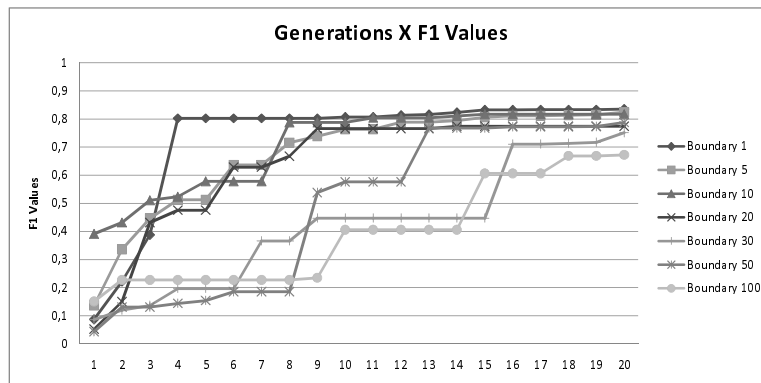


Figure 4.1. Fittest Individual Effort - Generations X F1 Values - Cora Dataset

identification boundary value. Thus, if the chosen boundary value is out of the reach of a possible effective evidence combination, this candidate solution (deduplication function) will fail in the task of identifying replicas.

In addition, this deduplication function must also take into account some errors that may be present in the dataset. The consequence is that not every evidence instance reaches the maximum similarity value (1.0) on a valid match and therefore, their summation or multiplication does not reach the number of attributes available.

Moreover, it is important to notice that, usually, not every evidence is used in the final solutions, as we can see on the results shown in Tables 4.13, 4.14, 4.15, 4.16 and 4.17. In these tables, we show some examples of the best individuals from the training phase (trees are presented in the pre-order output), despite the different boundary values. This was done in order to observe how the deduplication functions that evolved aiming at different boundary values managed to reach similar F1 levels.

Observing Tables 4.13, 4.14, 4.15, 4.16 and 4.17, we can also notice that setting a far distant positive boundary value forces our GP-based approach to create more elaborated functions. In Figures 4.1, 4.2, 4.3, 4.4 and 4.5, we can notice that as the boundary value increases, the evolutionary process requires more time (in number of generations) to raise the F1 values. When the functions evolve using lower boundaries, the F1 levels reach higher values in earlier generations.

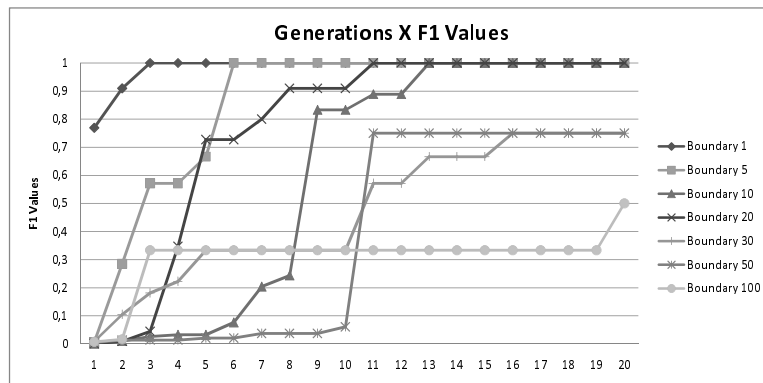


Figure 4.2. Fittest Individual Effort - Generations X F1 Values - Restaurants Dataset

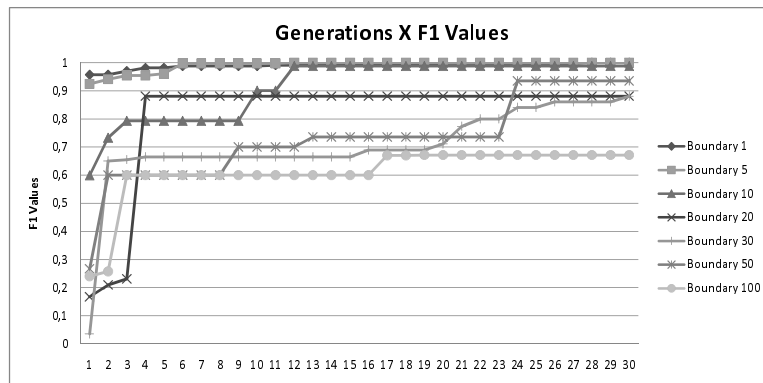


Figure 4.3. Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 1

This behavior occurs because these functions need to adjust themselves to keep the similarity values at the same (or close to the) level of the chosen boundary. In Tables 4.13, 4.14, 4.15, 4.16 and 4.17 it is possible to see that the suggested functions resulted from lower boundary values (1 and 5) are able to keep the function final results in a lower value range (close to 1.0) by using direct multiplications and summations on the available attributes.

However, the suggested functions resulting from the evolutionary process that used higher boundary values (10, 20, 30, 50 and 100) became progressively more elaborated, in some cases using multiplications and sums with integer values and increasing the number of nodes in the trees. At the same time, these functions do not always succeed in

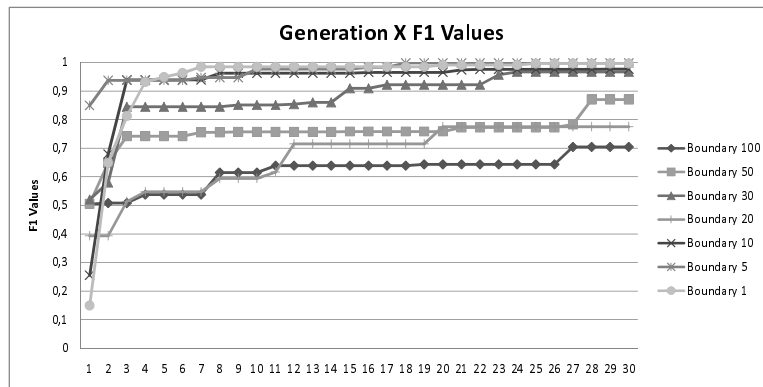


Figure 4.4. Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 2

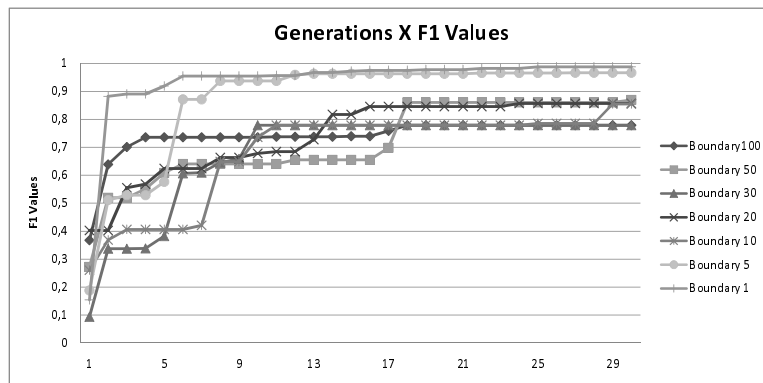


Figure 4.5. Fittest Individual Effort - Generations X F1 Values - Synthetic Dataset 3

reaching suitable F1 results, presenting low F1 averages and higher standard deviations. For this reason, choosing boundary values that minimize this effort saves processing time, since more complex functions require more time to be found.

The results of this final set of experiments show that our GP-based approach generates satisfactory solutions (deduplication functions) if the replica identification boundary is set to a value as close to 1 as possible.

4.5 Remarks on the Results

Based on the results of our experiments, we can make the following remarks:

1. It is not necessary to use every single evidence in order to identify replicas. In specific datasets, just some attributes are sufficient to carry out this task. Moreover, some attributes may be more discriminative than others when combined with a specific similarity function.
2. Specific dataset characteristics can be revealed by GP when the best evidence is discovered. As a consequence, an evidence that is useful for a specific dataset, may not be for another one.
3. We notice that when using our GP-based approach, the user does not have to worry about the problem of choosing which similarity function should be paired with a specific attribute. Our approach is able to automatically find the most suitable combinations, leading to similar results when compared with the previously fixed evidence experiment.
4. We noticed that it may be possible to automatically suggest a replica identification boundary when using our GP-based approach. The use of small values, particularly setting the value of the boundary to 1, was shown to be a suitable choice in our experiments.

Chapter 5

Impact of the GP Parameter Setup on Record Deduplication

To identify and remove replicas present in data repositories is a very expensive task. This happens since this task demands significant computational power in order to execute all requested record comparisons. For this reason, proposed methods for record deduplication must accomplish their goals considering that they should also be as efficient as possible.

In Chapter 4, we presented our GP-based approach to record deduplication. However, despite the good results obtained by it, bad choices of the GP parameters can affect the performance of our approach. For example, some parameter configurations may cause the record deduplication task to demand more time and resources than would be really necessary. On the other hand, using the most suitable values for the GP parameters can lead to faster and more efficient solutions.

In this chapter, we present the results of an experimental study that shows how the selection of GP parameters can impact the performance of the record deduplication task (de Carvalho et al., 2008b). Our experiment results show that different GP setups can cause significant difference over the effort required to obtain suitable solutions. Moreover, the range of the F1 averages for good and bad parameter setups can reach

up to 30%.

With this study, we intend to provide some guidelines for setting the parameters of our GP-based approach to record deduplication. We diminish the user effort on setting the GP parameters for this problem, since we provide detailed explanations on the parameters and what is the impact of each one over the final results.

Here, we present the results of the experiments conducted with our GP-based approach in order to evaluate the impact of the choice of some GP parameters in the results of record deduplication. The results of this evaluation are important because they can be used to suggest to the users the most efficient way to setup the GP parameters in order to identify replicas in data repositories.

This evaluation will comprise experiments with the following GP parameters: population size, number of generations, selection method, pairing method, reproduction and crossover proportion, mutation rate, and the initial random tree creation method.

The standard parameter setup used is presented in Table 5.1. This parameter setup and the evidence used in our experiments were based on a setup that allowed a fair experiment for comparing the evidence selection strategies presented in the previous chapter. Each experiment will vary only one parameter at a time, using this setup as the ground base for further comparisons.

5.1 The Experimental Dataset

In the experiments described in this chapter, we adopted the same synthetic dataset generator, SDG (Christen, 2005), used in the experiments presented in Chapter 4. Thus, for our experiments, we created a synthetic dataset containing 2.000 records. This dataset contains four files of 500 records (300 originals and 200 duplicates), with a maximum of seven duplicates based on one original record (using a Poisson distribution of duplicate records), and with maximum of four modifications in a field and five in the full record. The records in this dataset have the following fields (or attributes):

Parameter	Value
Number of Generations	30
Population Size	50
Initial Random Tree Creation Method	Ramped Half-and-Half
Reproduction Proportion	30%
Selection Method	Roulette Wheel
Mating (Pairing) Method	Random
CrossOver Proportion	70%
CrossOver Method	SubTree Exchange
Mutation Rate	7%
Mutation Operation	SubTree Insertion
Max Initial Random Tree Depth	3

Table 5.1. Standard GP Setup

forename, surname, street number, address1, address2, suburb, postcode, state, date of birth, age, phone number, and social security number.

The use of a synthetic dataset in our experiments made it possible to better evaluate the impacts in quality of the final solutions as a result of the changes in the parameter setup we suggest, since the existing errors and their characteristics are known. All experiments involved the following two steps: (1) the GP framework chooses one or more files for training purposes and (2) the GP framework tests the results of the training step on all remaining files.

For all experiments, we present the mean and standard deviation values after ten runs, using folded cross validation. Likewise the experiments described in the previous chapter, we do not used any blocking technique in order to avoid recall value distortions that might occur when splitting the dataset in smaller blocks and, therefore, affect the evaluation of our GP-based method suggested solutions.

5.2 Population and Generation Size

In this experiment, we handled the two most time impacting parameters of a GP system: the number of individuals that are processed in each generation and the maximum number of generations of the evolutionary process. Since each experiment requires

Parameter	Options	(1)Avrg	(1)StDev	(2)Avrg	(2)StDev	(3)Avrg	(3)StDev	(4)Avrg	(4)StDev
Population Size	10	0.642	0.432	0.649	0.424	0.661	0.425	0.662	0.428
	20	0.798	0.289	0.791	0.289	0.802	0.314	0.796	0.309
	30	0.831	0.108	0.816	0.147	0.840	0.101	0.841	0.121
	50	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	70	0.955	0.054	0.951	0.053	0.963	0.042	0.964	0.037
	100	0.974	0.028	0.981	0.012	0.975	0.017	0.981	0.015
Generation Size	10	0.782	0.172	0.797	0.181	0.813	0.175	0.814	0.169
	20	0.922	0.069	0.928	0.058	0.935	0.047	0.930	0.055
	30	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	40	0.947	0.064	0.949	0.063	0.954	0.054	0.954	0.066
	50	0.926	0.071	0.946	0.057	0.939	0.046	0.940	0.055
	60	0.959	0.036	0.969	0.024	0.962	0.031	0.968	0.023
	70	0.951	0.047	0.954	0.049	0.957	0.038	0.958	0.042

Table 5.2. Population and Generation Size - F1 Averages and Standard Deviations

training on learning datasets, the larger the values of these parameters, the more time will be required for the training phase. This becomes a real concern when applying GP to a problem that is also time demanding such as record deduplication. Moreover, if the experiment lasts longer than necessary, the solutions may end up over-fitted (over-specialized in the learning dataset). However, if lower values are used, the GP process will not be able to find good solutions. For this reason, it is important to choose the most suitable values for these parameters in order to achieve the desired goal.

The results of the experiments are presented in Table 5.2. The result of the best individual of the training phase is labeled as (1)Avrg and (1)StDev, for the average and standard deviation respectively. The results obtained in the test phase are labeled as (2), (3) and (4), corresponding to each test file described before. The results of the other experiments are presented in tables with a similar structure.

The results in Table 5.2 show that increasing the number of both individuals and generations leads to gains in the F1 averages, and also to more stable results (lower standard deviations). However, it can be noticed that beyond a certain number of generations and population individuals, 40 and 50 respectively in our synthetic dataset, there is an substantial increase on the effort necessary to obtain better results. For example, we need an increase of about 40% in the population size (50 to 70) to obtain gains of about 5% in the quality of the deduplication process. We can also observe that,

at the indicated values (40 for generations and 50 for population), our approach has already reached a stable result level, as can be seen by the low values of the standard deviation in the training and test datasets. Using much higher values than these would take substantive additional time and demand significantly more resources to obtain small F1 gains. If, however, the user has the time and resources for increasing the size of the population, this may lead to additional gains. On the other hand, increasing the number of generations may increase the chance of overfitted solutions.

5.3 Crossover–Reproduction Proportion and Pairing Methods

In this experiment, we deal with the parameters that usually impact the diversity of the solutions during the course of the evolutionary process: (1) the proportion between individuals that are reproduced and the ones that suffer genetic modification, and (2) the methods used to pair the individuals (during the crossover operation). Our objective is to suggest a “genetic pool” size by finding good reproduction and crossover rates for record deduplication, and also the most suitable way of pairing the individuals. For this, we used three well known strategies (described in Chapter 3): *random pairing*, *ranking pairing* and the *mirror pairing*.

Regarding the result of the reproduction and crossover proportion experiments, Table 5.3 shows that the 90-10 ratio leads to the highest F1 averages and also to the most stable results (e.g., it presents the lowest standard deviations). Close results can be obtained also using the 80-20 and 70-30 ratios, however, they are less stable. This behavior may be explained because the 90-10 ratio creates a genetic pool size more suitable for spawning good individuals.

Regarding the changes in the pairing methods, the mirror pairing presents the highest F1 averages in the training and test datasets. However, the results of the random pairing are statistically similar to the mirror pairing in terms of both average

Parameter	Options	(1)Avrg	(1)StDev	(2)Avrg	(2)StDev	(3)Avrg	(3)StDev	(4)Avrg	(4)StDev
Crossover-Reproduction Proportion	90-10	0.952	0.039	0.952	0.039	0.955	0.038	0.954	0.039
	80-20	0.919	0.102	0.926	0.086	0.915	0.108	0.925	0.088
	70-30	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	50-50	0.797	0.180	0.798	0.207	0.816	0.179	0.841	0.151
Pairing Method	Random	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	Ranking	0.898	0.124	0.910	0.133	0.904	0.141	0.907	0.147
	Mirror	0.927	0.077	0.937	0.081	0.943	0.063	0.948	0.060

Table 5.3. Crossover-Reproduction Proportion and Pairing Method - F1 Averages and Standard Deviations

Parameter	Options	(1)Avrg	(1)StDev	(2)Avrg	(2)StDev	(3)Avrg	(3)StDev	(4)Avrg	(4)StDev
Selection Method	Wheel	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	Tournament	0.933	0.077	0.940	0.076	0.941	0.065	0.939	0.081
	Random	0.953	0.023	0.947	0.032	0.958	0.026	0.955	0.024
	Greedy	0.917	0.086	0.915	0.098	0.918	0.081	0.925	0.084
	Ranking	0.977	0.014	0.975	0.024	0.975	0.023	0.974	0.022
Mutation Rate	10%	0.944	0.052	0.949	0.046	0.952	0.040	0.950	0.040
	5%	0.889	0.122	0.905	0.117	0.907	0.099	0.906	0.110
	3.3%	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	2.5%	0.853	0.218	0.856	0.231	0.882	0.196	0.869	0.206
	2%	0.924	0.051	0.934	0.044	0.929	0.046	0.933	0.048

Table 5.4. Selection Method and Mutation Rate - F1 Averages and Standard Deviations

and standard deviations. We can also point out that the random method is slightly more stable in some cases. One additional advantage of the random pairing is that it is easier to implement, saving sorting operations that are required by the other two methods.

5.4 Selection Method and Mutation Rates

In this experiment, we have changed the values of the parameters that are responsible for the GP ability to adapt the individuals to specific goals during the evolutionary process: the methods used for selecting individuals for future populations and how often the mutation occurs during the evolutionary process. The selection methods used in our experiments were: *roulette wheel*, *tournament*, *random*, *greedy* and *ranking*. Regarding the mutation rates, our main aim was to find the most adequate mutation rate for record deduplication.

As we can see in Table 5.4, the ranking selection method presented the best F1 averages and the lowest standard deviations. However, this result can be considered

Parameter	Options	(1)Avrg	(1)StDev	(2)Avrg	(2)StDev	(3)Avrg	(3)StDev	(4)Avrg	(4)StDev
Initial Tree Creation Method	Half-and-Half	0.910	0.071	0.920	0.060	0.929	0.066	0.921	0.077
	Full Depth	0.968	0.017	0.977	0.015	0.974	0.011	0.980	0.013
	Grown	0.818	0.217	0.832	0.225	0.827	0.228	0.829	0.226

Table 5.5. Initial Tree Creation Method - F1 Averages and Standard Deviations

statistically similar to those of the random and tournament selection methods. One explanation for this outcome may be particular characteristics of the record deduplication problem. Since the ranking selection method uses only the fittest individuals at each generation, it might have accelerated the convergence process (in this particular problem) towards the most suitable solutions, whereas the other selection methods might have slowed the convergence speed when mixing the genetic content of good and bad individuals.

Regarding the changes in the mutation rates, both, the 2% and 10% rates, presented statistically equivalent solutions. This behavior can be explained by the crossover disruptive effect (Banzhaf et al., 1998) over the relative small population, used in the standard experiment setup, that might have nullified the most impacting changes created by the mutated individuals. In other words, the crossover operation might be enough in this environment to find the most suitable solutions.

5.5 Initial Random Tree Creation Method

In this final experiment, we studied the impact that the initial random trees have on the GP solutions at the end of the process. We used the most commonly methods found in GP systems: *ramped half-and-half*, *full depth* and *grown*.

As we can see from Table 5.5, the full depth method leads to the highest F1 averages, and also to the most stable results (lowest standard deviations). The ramped half-and-half method presents results that are statistically similar to the full depth, but with slightly higher standard deviation values. The grown method presents the lowest F1 averages and a very unstable behavior.

These results can be explained by the fact that the initial (and usually larger) trees

created by the full depth and ramped half-and-half methods provide an initial solution that spreads better over the search space, when compared with the smaller trees that usually result from the grown method.

5.6 Remarks on the Results

Based on the results of our experiments, we can make the following remarks:

1. Beyond a certain level, it is necessary a substantial increase in the number of generations and in the population size in order to obtain small gains in F1 values. In our experiments, this level is reached when the values of the generation and population sizes are, respectively, 40 and 50. However, bad choices (small values) for these parameters can deteriorate the F1 results in more than 30%.
2. The most suitable crossover-reproduction proportion found in our experiments was 90-10. Good results can also be obtained using 80-20 and 70-30, however, the most stable results were achieved by the 90-10 proportion.
3. Albeit the better averages of the mirror pairing method, there is no statistically advantage over the other pairing methods used in our experiments. Thus, our suggestion is to use the most straightforward method, random pairing, in order to save computational time.
4. Ranking selection presents the best results among the selection methods experimented. Compared with the method that presented the worst result, it provides an improvement of up to 4% in F1 average values.
5. Higher mutation rates provide statistically similar results when compared with lower ones. This might mean that the crossover operation, in this problem, already creates a good diversity among the population. This fact leads to our suggestion of keeping the mutation rate at the lower value of 2%, in order to save computational time.

-
6. In GP, a good strategy to reach suitable final results is to improve the way the evolutionary process starts, in our case, the way the initial population is created. Our experiments with initial random tree creation methods suggest the use of the full depth method. By only changing this parameter, the difference between the worst and the best F1 averages varied over 10% in some cases.

Chapter 6

An Evolutionary Approach to Schema Matching

The schema matching problem can be defined as the task of finding semantic relationships between schema elements (e.g., relation attributes or XML tags) from two distinct data repositories (e.g., relational tables or XML files). Solving this problem is key in many data integration environments that use large and complex data sources, such as heterogeneous databases, data warehouses, and web data repositories. This problem usually occurs due to the number of different possibilities of expressing the same real world concepts when modeling a data repository. It can also arise when different data modeling paradigms (e.g., the ER and object oriented models) are used to model distinct applications in a same domain.

Despite the existence of elaborated tools that usually provide advanced graphic resources for helping with this task, the matching between elements from the candidate schemas is usually “hand crafted”, i.e., the user has to specify, manually, the actual matches. Thus, this is a labor intensive, very expensive, and error prone process.

As discussed in Chapter 2, to overcome this situation, several alternative tools based on automatic or semi-automatic approaches have been proposed (Dhamankar et al., 2004; Doan et al., 2001; Hernández et al., 2001; Madhavan et al., 2001; Milo

and Zohar, 1998). These approaches aim at combining different pieces of evidence extracted from the repository structures, the stored data, and even external resources (i.e., semantically related datasets, web catalogs, ontologies, etc.) to provide information about the semantic relationships.

However, in some real world scenarios it may not be possible to use evidence such as these. There are specific situations in which no information about the schema structure is available (for instance, in cases where the schema is not known) or the information is limited, such as in the case of semistructured data (Rahm and Bernstein, 2001). In this information-restricted scenario, only the data instances may be available for use. Indeed, schema matching approaches that rely only on data instances are the unique option for this task in unfavorable scenarios, such as fraud detection, crime investigation, counter-terrorism, and homeland security. Most often, data repositories manipulated in such scenarios are deliberately deprived from any form of identification or structural clues to make it difficult to be processed and analyzed.

Another complex issue we need to deal with is that most real-world situations involve *complex matches*. Differently from the task of finding 1-1 matches (that represent a relationship between single elements from distinct schemas), identifying complex matches is a more challenging problem. This happens because complex matches require finding combinations of existing elements in one schema that are related to combinations of elements into another, as illustrated in Figure 6.1. Thus, in such situations, the search space of possible solutions is unbounded (Dhamankar et al., 2004).

In this chapter, we propose an evolutionary approach¹ that aims at automatically finding complex matches between schema elements of two semantically related data repositories. Since we only exploit the data stored in the repositories for this task, we rely on matching strategies that are based on record deduplication and information retrieval techniques to find complex schema matches.

The reason for adopting an evolutionary approach is the size of the search space

¹Our approach adopts concepts and ideas that are strongly inspired by genetic programming, but does not use the “classic” data structures usually adopted for representing the solutions.

Figure 6.1. Semantic Relationships between Repository Schemas: Complex Matches

Table A

given name	surname	street number	address 1	address 2	suburb
rosie	leslie	26	coranderrk street	rowethorpe	hill end
katherine	hand	18	derrington crescent	homewood	kingsthorpe
M.	white	23	prescott street		bonbeach

Table B

full name	Age	address	area
rosie leslie	43	coranderrk 26, rowethorpe	hi end
katherine hand	33	derrington crescent 18 , homewood	kingsthorpe
M. white	39	prescott str	bonbeach

resulting from the large number of possible semantic relationships existing between element combinations from two schemas. Evolutionary techniques are known for their capability to efficiently find suitable answers to a given problem, without having to explore the entire space of solutions and when there is more than one objective to be accomplished (Banzhaf et al., 1998) (in our case, finding several different valid matches at the same time).

The goal of our approach is to find semantic relationships between schema elements, in a restricted scenario in which it might not be possible to use all information due to the absence of structural data or to the high cost of obtaining it (e.g., for security reasons), or even when useful schema information is limited, as in the case of semistructured data. Alternatively, our approach can improve the results of other approaches (e.g., (Dhamankar et al., 2004)) that combine different pieces of information, such as schema structure, attribute labels, schema domain, and data instances.

Our evolutionary approach is able to identify complex matches with high accuracy, despite the restriction on the use of structural information. Other approaches that only explore stored data instances, such as (Doan et al., 2001) and (Li and Clifton, 2000), are not able to find complex matches even when using external information (e.g., training examples).

As pointed out in Chapter 2, a more recent work (Warren and Tompa, 2006) describes a specific multi-column string matching approach that actually performs data mappings between schema elements, i.e., it identifies the operations that are required to transform a set of schema elements into another one. However, the authors point out that their approach requires the existence of some common data between the repositories in order to find a solution. Additionally, that approach also needs some external information, such as an initial set of attributes that were previously matched, in order to solve complex mappings.

To demonstrate the effectiveness of our approach, we conducted an experimental evaluation using real and synthetic datasets. The results of this evaluation show that our approach was able to find complex matches, in some cases all of them, in three different scenarios: when the repositories do not share any common data, when the repositories partially share some data, and when the same data can be found in both repositories but in different formats or differently structured. More precisely, our approach achieved 100% accuracy in the synthetic datasets and up to 60% in the real ones, in the task of finding complex matches (more details on these experiments in Section 6.2). Our results are competitive when compared to those achieved by a more elaborated (hybrid) approach (Dhamankar et al., 2004) (using the same real datasets) that relies on both schema and actual data to find complex matches.

6.1 Proposed Approach

In this section, we describe our evolutionary approach to schema matching. Thus, first we present the schema matching solution representation we adopt for supporting the evolutionary process and briefly describe how genetic operations are applied during this process. Then, we explain two matching strategies we adopt in our approach to combine and compare the available data with the goal of finding and evaluating matches in different situations. The first is a record-oriented strategy that is based on a record

deduplication method that we have proposed (de Carvalho et al., 2008a), whereas the second is an attribute-oriented strategy that employs information retrieval techniques. The choice of which matching strategy to use depends on the data integration scenario considered, as we shall explain in the next section when describing our experiments.

6.1.1 Schema Match Solution Representation

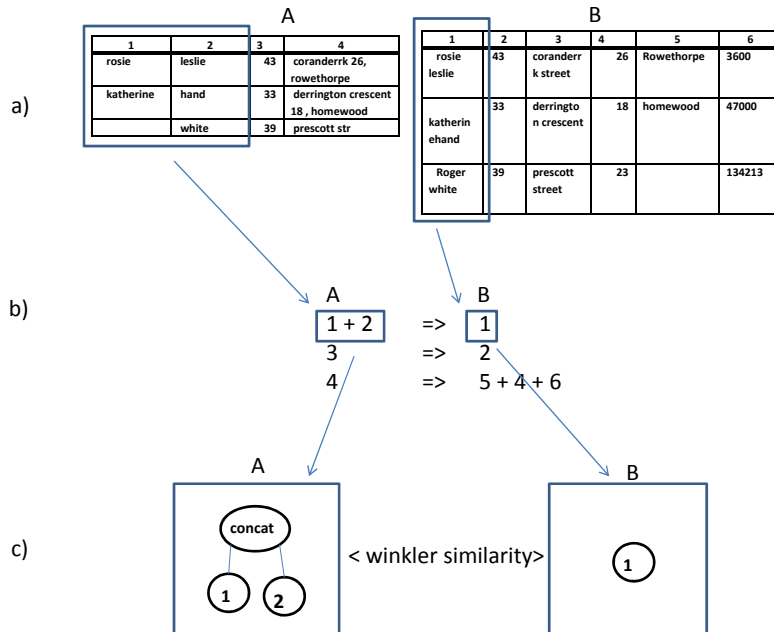
The basic blocks that we use to build the solutions (schema matches) for our problem are the *schema elements*. A *schema element* can be seen as a property inherent to an entity, or associated with that entity, for data representation purposes (e.g., the name of a person and the price of an item). Hence, for example, in a relational table, a schema element corresponds to an attribute (column) of this table.

Schema elements can be grouped into one or more sets. A *set of schema elements* comprises one or more schema elements. However, for our purposes, such a set can only hold schema elements that belong to the same *element data class*. An *element data class* aggregates elements that are manipulated in the same way, by methods or operations that make sense only for that data class². For example, we can list the following element classes and some of their possible operations:

- Strings: concatenation, subtraction, substitution, insertion, replacement;
- Dates: summation, subtraction, comparison (e.g., equal to, after than, before than);
- Numbers: summation, division, subtraction, multiplication, exponentiation;
- Sound streams: concatenation, insertion, substitution, normalization;
- Images: color substitution, transposition;

²We notice, for instance, that two schema elements of type *string* belong to the same data class, no matter their defined length.

Figure 6.2. A Match Representation Example



The relationships between schema elements, i.e., a *match*, is represented by two *combination trees* and a *similarity measure*. A combination tree describes how a set of schema elements from a repository may be combined to match another schema element or a specific set of schema elements from another one. For example, a pair of combination trees can be used to find a match between schema elements that represent personal names in two distinct repositories. The similarity measure is used to evaluate the quality of the match. We further elaborate on this issue later.

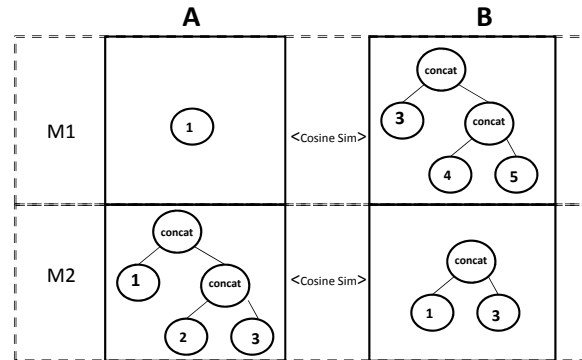
All combination trees must be formed by schema elements that belong to the same data classes, which are represented by tree leaves. The internal nodes of the combination trees are functions and operators that are able to manipulate schema element instances from the same data class of the leaves. Thus, we can say that a combination tree is from data class X if their schema elements (their leaves) are all from data class X . In our modeling, both combination trees in a match must comprise elements from the same data class. By using a list of relationships between combination trees from different repositories, we produce a *schema matching solution*, which in our evolutionary approach is modeled as a unique *individual* in a population.

To illustrate this representation, consider two data repositories A and B as shown in Figure 6.2. Repository A comprises four schema elements (*forename*, *surname*, *age*, and *fulladdress*) and, repository B, six (*fullname*, *age*, *streetname*, *streetname*, *city*, and *postalcode*). Since we do not use any structural information (such as attribute labels), the schema elements are labeled in our approach by id numbers (Figure 6.2(a)), 1 to 4 and 1 to 6 in repositories A and B, respectively. For the sake of simplicity, in our example we consider all schema elements as being of the same class. We also show how our combination trees (Figure 6.2(c)) may represent one of the several schema matches (Figure 6.2(b)) found between repositories A and B. Notice that id numbers identify the leaf nodes (schema elements) in the combination trees and the Winkler similarity measure is used to evaluate the matches.

In our next example, shown in Figure 6.3, the individual comprises two matches (in our experiments, the number of matches that correspond to an individual is usually bigger) that are instantiated as randomly generated trees (Banzhaf et al., 1998). Notice that match M1 does not make any sense because it matches personal names with addresses whereas match M2 also presents some problems as it tries to match a combination of *forename*, *surname* and *age* (attributes 1, 2, and 3) in repository A with a combination of *fullname* and *streetname* (attributes 1 and 3, respectively) in repository B, being again a bad match.

This schema matching solution representation (with several matches represented by lists of combination trees within an individual) aims at enabling our evolutionary process to search for several valid matches at the same time (e.g., M1 and M2 in our example). Since there are several matches within an individual, the search for good matches occurs more or less in parallel, since a good individual must have several good matches. This avoids the undesired side effect of having GP looking for a unique single “best match” using as many attributes as possible. We in fact observed this behavior in an early version of our approach when we modeled each individual as a single match.

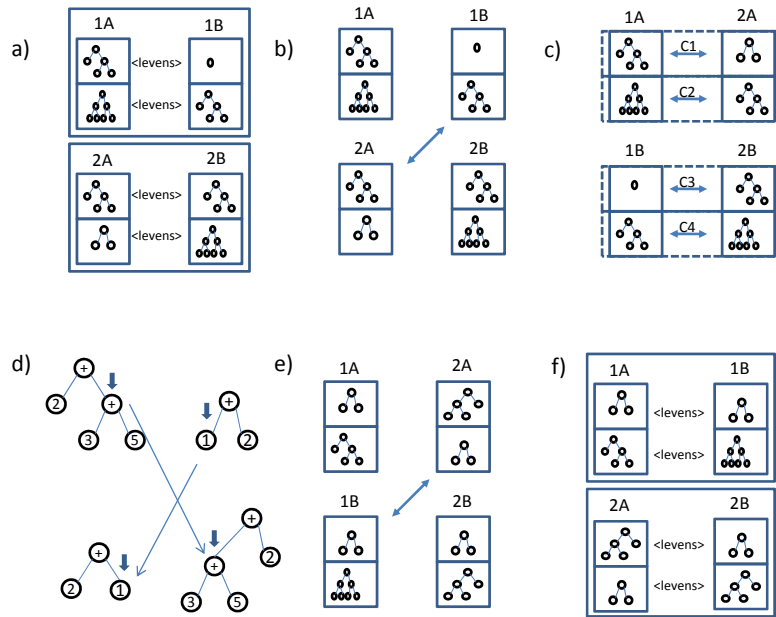
Moreover, we adopt this structure (list of combination trees) for representing

Figure 6.3. Random Generated Individual with Two Schema Matches

relationships between schema elements because it is suitable for genetic operations (Banzhaf et al., 1998; Koza, 1992). As we use a generational algorithm (Banzhaf et al., 1998), within each cycle the population of individuals is handled by genetic operations (such as *selection*, *mutation*, and *crossover*) to create new individuals with new and different characteristics from its predecessors. Figure 6.4 illustrates how the crossover operation (Step 6 of the generational algorithm described in Chapter 3) creates new individuals in our approach. As we can see, first two individuals are selected from the population (Figure 6.4(a)). Then, matches related to the same repository are paired (1A with 2A and 1B with 2B) (Figure 6.4(b)). Next, a list of crossover operations (C1, C2, C3 and C4) is scheduled between combination trees from different individuals (Figure 6.4(c)). Figure 6.4(d) zooms in the C1 crossover operation between the combination trees related to repository A. Finally, matches are paired (Figure 6.4(e)) to form the new individuals (1A with 1B and 2A with 2B) which are then combined to be used in the next round of the evolutionary process.

The similarity measure resulting from each match evaluation is used to classify the combination trees as a valid match or not. This similarity measure is calculated by means of a similarity function that should be chosen according to the data class of the combination trees. For instance, if the combination trees combine strings, the similarity function (e.g., a string distance function or Jaro similarity function) should be able to measure how similar two strings are. The higher the value returned by

Figure 6.4. Crossover Operation



this function, the higher the probability that the combination trees represent a valid semantic relationship between the two repositories. However, there are several of such matches possible, thus leading to a huge search space for the evolutionary approach to find the best solutions.

In order to control the search for possible solutions, we need to narrow the evolutionary process. This is done by restricting the application of the *crossover* and *mutation* operations only to matches that involve combination trees comprising schema elements from the same data class. This avoids schema elements from different data classes to appear in the same combination tree and, therefore, prevents the evolutionary process to evaluate individuals that lead to invalid solutions. Also, during the creation of the initial population, we do not allow the same schema element to appear in more than one combination tree of the same individual. This constraint is somewhat maintained by the evolutionary process itself, since its violation would imply in matchings of lower quality within an individual. This is another indirect benefit of our solution representation.

The final solution is computed by taking the results of the evaluations over all

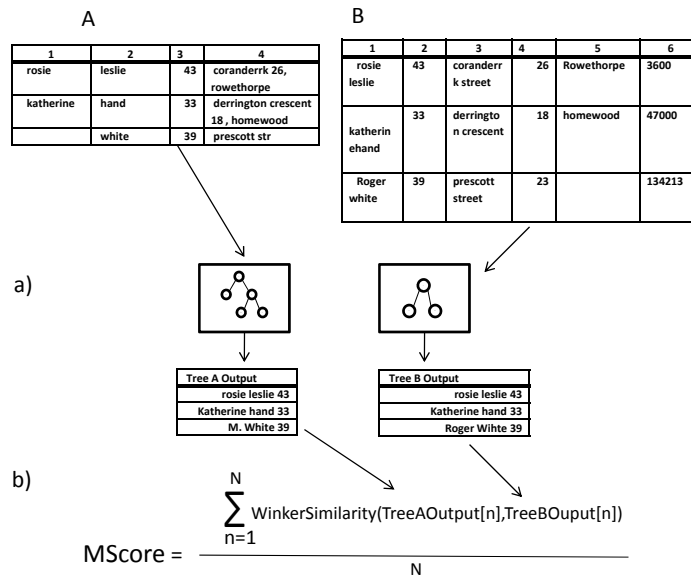
matches of each single individual. For this, we use as our fitness function the average of the summation of the individual similarity values obtained by all matches in the candidate solution. The higher the similarity of all matches, the more accurate is the final solution.

However, there are several ways of computing these similarity values for the fitness function. In our approach, in order to deal with different data scenarios, we devise two distinct matching strategies to find and compute values between the matches: a record-oriented strategy that is based on our record deduplication approach described in Chapter 4, and an attribute-oriented strategy that exploits information retrieval techniques. We detail these strategies in the following sections.

6.1.2 Record-oriented Strategy

As discussed in Chapter 4, a record deduplication process aims at identifying in a data repository records that refer to the same real world entity or object in spite of misspelling words, typos, different writing styles or even different schema representations or data types. Thus, when repositories share some data about the same real-world entities, we can combine the data associated with several schema elements to identify whether two or more entries in a repository are replicas or not. We accomplish this by using functions that combine schema elements using our schema matching representation. If we are able to find schema element combinations that lead to a positive record replica identification, this means we have successfully identified valid schema matches.

Thus, in this strategy, each schema element E is combined in a combination tree with other schema elements from its same data class. For example, a very simple linear combination of schema elements may be expressed by the concatenation of several string schema elements, such as $E_1 + E_2 + E_3 + E_4$. The tree input is a set of schema elements, all belonging to the same data class and whose instances are extracted from the data repository, and its output is a combined schema element whose data class is the same of the leaves. In other words, when using the record-oriented strategy, the

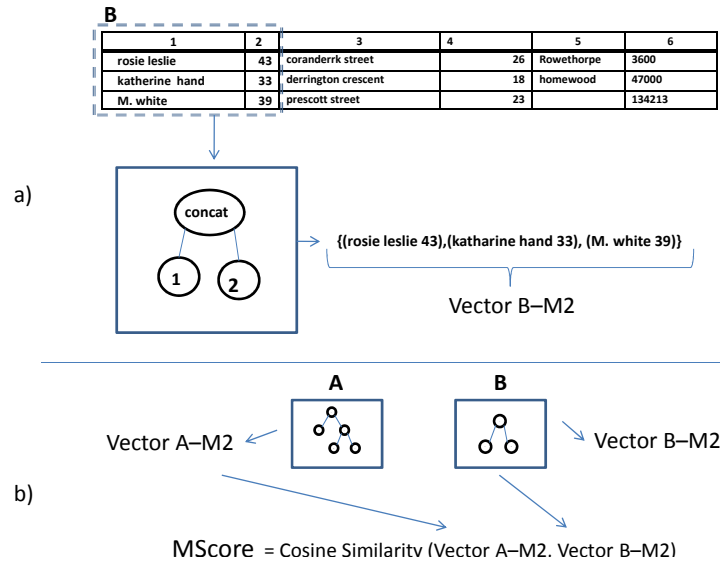
Figure 6.5. Match M2 Fitness Evaluation using a String Similarity Measure

trees encode how the schema element instances are combined, one repository entry at time, as illustrated in Figure 6.5.

As we can see, the output of the combination tree from one repository is compared with the output of the combination tree from the other repository, using a similarity function compatible with them: if the resulting similarity value is above a chosen boundary, the matching is considered valid; otherwise, the matching is considered invalid (de Carvalho et al., 2008a). In the example shown in Figure 6.5, the results obtained from each tree are used to build “output” tables. Then, the entries of these tables are pairwise compared using a previously defined similarity function (for example, based on the Winkler similarity measure), as illustrated in Figure 6.5(a). Finally, the average of the similarity values found comparing the table entries (MScore in Figure 6.5(b)) is used as a score to determine whether the schema match is valid or not. The fitness of the individual is the average of the scores of all matches that it beholds.

6.1.3 Attribute-oriented Strategy

Another possibility for determining similarities between two schema elements (or combinations of them) is to consider the whole content of an schema element (or attribute)

Figure 6.6. Match M2 Fitness Evaluation using the Cosine Similarity Measure

as a single unified content for comparison purposes. As we shall see later, this is useful when the schemas being matched are semantically related but share no data in common (i.e., data about a same entity). In this case, we rely on the classic vector space model (Baeza-Yates and Ribeiro-Neto, 1999) widely used in information retrieval as the basis for the similarity comparisons. The choice of this model is twofold: (1) it is easy to implement and usually provides an effective solution for text similarity problems and (2) its cosine function allows ranking the matches found between schema elements (or their combinations) from both repositories based on their degree of similarity, which is important to establish which are the most likely valid matches.

For example, consider two schema elements (or combinations) A_n and B_m from repositories A and B, respectively. Each one of these schema elements is represented by a set of terms extracted from their respective instances in A and B. The similarity between A_n and B_m is determined by the degree of similarity between their respective sets of terms A_n and B_m and is given by

$$\text{sim}(A_n, B_m) = \frac{\sum_{i=1}^t w_{i,n} \times w_{i,m}}{\sqrt{\sum_{i=1}^t w_{i,n}^2} \times \sqrt{\sum_{i=1}^t w_{i,m}^2}} \quad (6.1)$$

where, for a term k_i in the set B_m , $w_{i,n}$ is the weight associated with the pair (k_i, A_n) ,

$w_{i,m}$ is the weight associated with the pair (k_i, B_m) , and t is the number of terms in the sets A_n and B_m . If $sim(A_n, B_m)$ is greater than a similarity threshold previously specified, the schema elements are considered a valid match.

The weight $w_{i,n}$ is given by

$$w_{i,n} = f_{i,n} \times idf_i \quad (6.2)$$

where $f_{i,n}$ is the normalized frequency of the term k_i found in A_n and idf_i is the Inverse Document Frequency (IDF) of the term k_i that indicates its importance of the term k_i inside A_n .

The normalized frequency $f_{i,n}$ is given by

$$f_{i,n} = \frac{freq_{i,n}}{\max_l(freq_{l,n})} \quad (6.3)$$

where $freq_{i,n}$ is the raw frequency of the term k_i found in the schema element A_n (i.e., the number of times that the term k_i is found in A_n) and $\max_l(freq_{l,n})$ is the biggest frequency among the frequencies of the terms k_i , that are found in A_n .

The IDF value idf_i is given by

$$idf_i = \log \frac{N}{n_i} \quad (6.4)$$

where N is the total number of schema elements considered and n_i is the number of schema elements in which the term k_i appears.

Finally, the weight $w_{i,m}$ is given by

$$w_{i,m} = \left(0.5 + \frac{0.5 \times freq_{i,m}}{\max_l(freq_{l,m})}\right) \times idf_i \quad (6.5)$$

where $freq_{i,m}$ is the raw frequency of the term k_i found in B_m and $\max_l(freq_{l,m})$ is the biggest frequency among the frequencies of the terms k_i found in B_m . The

two constant values 0.5 were suggested by Salton and Buckley (ibdi (Baeza-Yates and Ribeiro-Neto, 1999)) to balance the relevance of terms in a document collection. Since then, these values have been successfully used in the implementation of virtually all practical information retrieval systems.

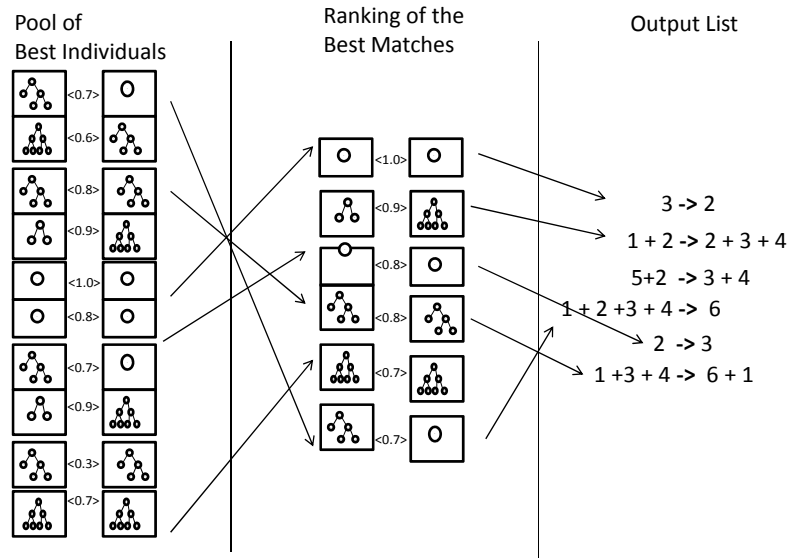
The main difference between the record- and the attribute-oriented strategies lies on the way each match similarity is calculated for an individual. Thus, in the attribute-oriented strategy, each tree encodes the instructions on how to handle the data (extracted from actual instances) in order to fill a space vector. The vector now “represents” the tree output, as illustrated in Figure 6.6(a). The vectors (from both trees in each match) are then compared using the cosine similarity function. This calculated similarity is now the match score, MScore, shown in Figure 6.6(b). If this score is above a chosen boundary, the match is considered valid; otherwise, the match is considered invalid. Likewise the record-oriented strategy, the fitness of the individual is the average of the scores of all matches that it beholds.

6.1.4 Selecting the Best Matches

During the evolutionary process, the combination trees are manipulated by means of genetic operations. At each generation, each individual is inspected to assess how it solves the matching problem. This is accomplished by calculating its fitness. Thus, each schema match (for instance, M1 and M2 in our example) found for an individual is evaluated.

This continuous process creates several possible ways of building matches with different combinations of schema elements. This happens because of the several possible distinct trees that might be created, some spawning very good matches, others very bad ones. However, only the best individuals are selected for further generations, since they include the best schema matches.

At the end of the process, the best individuals are chosen to compose a ranking of the best schema matches, as illustrated in Figure 6.7. The best matches found in

Figure 6.7. Selecting the Best Matches for the User Output List

each individual are then extracted and a single list is created and processed to remove eventual repetitions, since the same match solution may appear in distinct individuals. Finally, this list is translated to a human-readable format and returned as the result of the schema matching process.

6.2 Experiments

In this section, we discuss the results of the experiments conducted to evaluate our evolutionary approach to schema matching. There were three sets of experiments:

1. Experiments with repositories that partially share the same data (partially overlapped data scenario): The objective is to evaluate how our approach performs when there is some overlapping data in the repositories, despite different representations and possible errors, such as typos and phonetic errors. This is usually a common situation in real-world schema matching cases.
2. Experiments with repositories that present the same data (fully overlapped data scenario): The objective is to evaluate our approach when the repository data

instances are fully mapped, i.e., for each entity in the first repository, there is at least one corresponding entry in the second one.

3. Experiments with repositories that do not share the same data (disjoint data scenario): The objective is to evaluate how our evolutionary approach, using our proposed attribute-oriented strategy, performs when the repositories do not share any common data. This is the most difficult case for any instance-level schema matching approach, since there is not any corresponding entry between the repositories.

Our experimental evaluation adopts the same accuracy metric used in (Dhamankar et al., 2004), that is, the fraction of all target schema elements whose candidate match is correct, as given by:

$$Accuracy = \frac{NumberOfCorrectlyIdentifiedMatches}{NumberOfMatches}$$

For conducting these experiments, we have implemented a prototype schema matching system that consists of two modules: a 1-1 match searcher and a complex match searcher.

6.2.1 Datasets

In our experiments, we used two real datasets widely employed for evaluating schema matching approaches (Dhamankar et al., 2004; Lee et al., 2007; Madhavan et al., 2005). Both datasets (Real Estate and Inventory) were obtained from the Illinois Semantic Integration Archive³. These real datasets were used only in the disjoint data scenario experiment, since they do not present any overlapping data. Despite relatively small, these datasets are “de facto standards” used by several schema matching approaches and their use allows a direct comparison with them. In addition, we generated other datasets using the Synthetic Dataset Generator (SDG) (Christen, 2005) available in the Febrl package.

³<http://pages.cs.wisc.edu/~anhai/wisc-si-archive/>

Since real data is not easily available for experimentation, due to privacy and confidentiality constraints, we used the synthetic datasets to conduct a larger set of experiments to evaluate our approach. SDG generates datasets containing personal names (based on frequency tables for given forenames and surnames), addresses (based on frequency tables for localities, postcodes, street numbers, etc.), phone numbers, and personal number ids (like the social security number). Moreover, SDG is able to simulate real errors such as attribute swapping, typos, and phonetic errors. Thus, the generated synthetic datasets present anomalies that occur in real-world repositories.

For our experiments, we generated three synthetic datasets with the following attributes: *forename*, *surname*, *street number*, *address1*, *address2*, *suburb*, *postal-code*, *state*, *date of birth*, *age*, *phone number*, and *social security number*. These synthetic datasets are much larger than the Real Estate and Inventory datasets and present the following characteristics:

- The synthetic dataset 4 (Synthetic 4) was generated for experiments that deal with the partially overlapped data scenario. It consists of two files, each one with 2000 entries, with 10% of the entries being shared between them.
- The synthetic dataset 5 (Synthetic 5) was generated for experiments that deal with the full overlapped data scenario. It consists of two files, each one with 2000 entries, with all entries being shared between them.
- The synthetic dataset 6 (Synthetic 6) was generated for experiments that deal with the disjoint data scenario. It consists of two files, each one with 2000 entries, with no common entries between them.

In order to introduce complex matches in the synthetic datasets, we have combined some schema elements (*forename + surname* and *address1 + address2 + street number*) in one of the files of each generated dataset, similarly as we find in the Real Estate and Inventory datasets. We notice, however, that none of the datasets include

Table 6.1. Dataset Characteristics

Characteristic	Synthetic 4, 5, 6	Real Estate	Inventory
Total of Elements in File A	12	32	44
Total of Elements in File B	7	19	38
Total of 1-1 Matches	7	7	27
String Matches	3	6	11
Numerical Matches	4	1	16
Total of Complex Matches	2	12	11
String Matches	2	5	4
Numerical Matches	0	7	7

Table 6.2. GP Parameters: (1) Partially Overlapped (2) Fully Overlapped (3) Disjoint

Parameter	Value (1)	Value (2)	Value (3)
Max Number of Generations	20	20	20
Population	100	100	200
Reproduction Rate	20%	20%	20%
Selection	Ranking	Ranking	Ranking
CrossOver Rate	80%	80%	80%
CrossOver Method	Random Exchange	Random Exchange	Random Exchange
Mating (Pairing)	Random	Random	Random
Mutation Rate	2%	2%	2%
Max Random Tree Depth	3	3	3
Similarity Function	Levenshtein Distance	Sort Winkler	Cosine Similarity
Similarity Boundary	0.90	0.85	0.95
Number of Matches in a Individual	6	6	6

complex matches on numerical schema elements. Table 6.1 summarizes the matching characteristics⁴ of all datasets.

For each data scenario, different datasets and parameter setups (evolutionary parameters and similarity functions options) were used. We present this information in Table 6.2.

We ran our experiments in a workstation with the following hardware and software configuration: Pentium Core 2 Duo Quad (2 Ghz) processor, with a 4 GB RAM DDR2 memory and a 320 GB SATA hard drive, and running a 64-Bits FreeBSD 7.1 Unix-based operational system.

⁴In Table 6.1, by numerical matches we mean matches involving numerical values in general, as well as dates and other numeric types that require some conversion.

6.2.2 Basic Steps

In order to allow a fair comparison, each experiment performs the following steps:

1. An initial list of schema elements from both files is provided to our matching system. In the current version of our system, we consider all schema elements as strings in order to evaluate how our approach performs in this situation. Dealing with other types of elements is left for future work.
2. Our 1-1 match searcher processes the initial list of schema elements. Here, all 1-1 combinations of schema elements are tested in order to find the simplest or most obvious matches. The valid 1-1 matches are identified and removed from the list. This step helps improving the results of the complex match searcher, since it prunes the list of schema elements to be considered.
3. The complex match searcher receives the list of schema elements from the 1-1 match searcher and starts the evolutionary process. However, differently from what is done by the 1-1 match searcher in the previous step, the complex match searcher compares combinations of schema elements (combination trees). This enables the complex match searcher to find both 1-1 (eventually not found before) and complex matches.
4. A ranking of the best matches is returned as a list that comprises the schema element sets and their respective similarity values.

6.2.3 Partially Overlapped Data Scenario

This first set of experiments evaluates the capability of our evolutionary approach to find complex matches in partially overlapped data repositories. In this case, our schema matching system has to deal with some “noise” during the search process, since there is no external evidence or structural information available to be used. This is the most common real-world scenario (Rahm and Bernstein, 2001). We used only the Synthetic

Table 6.3. Results: Partially Overlapped Data Scenario

Matches	Accuracy
All 1-1 Matches	57%
String 1-1 Matches	100%
Numeric 1-1 Matches	24%
All Complex Matches	75%
String Complex Matches	75%

4 dataset for these experiments, since there is no partially overlapped data in the Real Estate and Inventory datasets.

In this scenario, our searchers employ the record-oriented strategy to find the most similar schema element combinations by comparing the corresponding instances stored in both files. The most similar combination is considered a possible replica, thus it is used for matching purposes.

Table 6.3 shows the results for the partially overlapped data scenario using Levenshtein, a well known edit distance function, as the similarity function. This function was the most effective among many others (e.g., Winlker, Jaro and Soundex) we used in our set of experiments. The similarity boundary value for this experiment was empirically chosen after initial tuning tests and its value was set to 0.9, since it maximized the effectiveness of the Levenshtein function.

Our 1-1 match searcher was able to find 57% of all matches. As we can see, most of the unidentified matches are numeric ones, which is mainly due to the fact that we have treated all schema elements as strings. As a consequence, the unidentified 1-1 matches increased the search space of the complex match searcher. On the other hand, the complex matching searcher found 75% of the complex matches. This result includes mostly complex matches involving addresses and one partial match⁵ involving personal names.

We notice that finding semantic relationships among numeric schema elements is a problem by itself. Even in elaborated hybrid systems, such as iMAP (Dhamankar et al., 2004), this is considered a separate task. For instance-level approaches, like

⁵A partial match is a complex match that misses one or more schema elements.

ours, specific strategies for identifying existing data patterns, other than those based on finding common terms and word frequencies, must be deployed.

6.2.4 Fully Overlapped Data Scenario

This set of experiments evaluates the capability of our evolutionary approach to find complex matches in fully overlapped data repositories. This means that the comparisons always take place between entries that correspond to the same real-world entities. This scenario is a special or particular case of the partially overlapped data scenario, in which the disjoint data is removed (or ignored) in order to improve the search effectiveness.

For the same reason discussed above, we used in these experiments only the Synthetic 5 dataset. We have performed several runs with this dataset and noticed that it is not required to use all its entries in order to reach good results. For instance, using only 500 entries we were able to obtain similar results when using all entries available in the files. This can be explained by the fact that both match searchers only compare entries corresponding to the same real-world entities, which considerably reduces the matching effort.

Table 6.4 shows the results for the fully overlapped data scenario. Likewise, these results were also obtained after a set of experiments using several different similarity functions. In this scenario the Winkler similarity function was the most effective. This function returns an approximate string matching measure using the Winkler string comparator on the word-sorted input strings (if there are more than one word in the input strings), which improves the results for swapped words. This also improves the effectiveness of our searchers, because it reduces the space of possible solutions, since we are looking for matches between schema elements, and not for the actual mappings between them (which can be done with tools like the one described in (Warren and Tompa, 2006)). In the previous set of experiments, the most effective similarity function was a specific edit distance (Levenshtein) function because the partially over-

Table 6.4. Results: Fully Overlapped Data Scenario

Matches	Accuracy
All 1-1 Matches	100%
Strings 1-1 Matches	100%
Numeric 1-1 Matches	100%
All Complex Matches	100%
String Complex Matches	100%

lapped data scenario requires a fine-grain replica identification process. The similarity boundary value for this experiment was also empirically chosen after initial tuning tests and its value was set to 0.85, since it maximized the effectiveness of the Sort Winkler similarity function.

As shown in Table 6.4, our 1-1 match searcher was able to identify all 1-1 matches. Even matches involving numeric schema elements (treated as strings) were identified. This happened because the comparisons were performed between entries corresponding to the same real-world entities and the numerical values for each entry were mostly the same ones in both files, allowing the correct matching. The complex match searcher was also able to identify all existing matches. Moreover, since we have used the Sort Winkler similarity function, some of the correct answers found in the output solution presented the same schema elements, but with different concatenation sequences (e.g., *forename* after *surname*). For matching purposes, this is not a problem because the final solution includes the correct schema elements.

These results show that the effectiveness of our match searchers was improved by the specific characteristics of the fully overlapped data scenario. In this scenario, the match searchers only have to identify subsets of schema elements that are able to improve the overall similarity results. In the previous experiments, with partially overlapped data, our searchers also need to deal with different entities during the comparisons. Comparisons involving different entities may mislead the replica identification, being, therefore, considered as “noise” for the searchers.

6.2.5 Disjoint Data Scenario

This last set of experiments evaluates the capability of our evolutionary approach to find schema matches in disjoint data repositories. This means that they do not share entries that correspond to the same real-world entities and, therefore, cannot be used as “rules” for identifying semantic relationships. For this reason, it is the most difficult scenario for any schema matching system based on the instance-level approach (Rahm and Bernstein, 2001).

Since the repositories do not share entries that correspond to the same real-world entities, we cannot use our record-oriented strategy in this scenario. Hence, we use instead our attribute-oriented strategy, which considers a similarity function derived from the classic vector space model.

The rationale of this strategy is to find matches between schema elements that present similar term frequencies and share common terms (e.g., personal names, addresses and product descriptions). For this, we compare text vectors that are assembled using all data instances stored in the repository that are associated with a combination of schema elements. The higher the similarity value between the text vectors, the higher the likelihood of the schema element combinations being related to each other. The similarity boundary for this experiment was empirically chosen after initial tuning tests and its value was set to 0.95, since it maximized the effectiveness of the cosine similarity function.

We used the Synthetic 6, Inventory and Real Estate datasets in these experiments. However, we notice that the two real datasets are too small (only 100 entries), since they were created using real data, aiming at experiments with hybrid schema matching approaches. Considering that real datasets are usually much larger, we overcome this situation by using a synthetic dataset whose data better resembles that found in real applications.

Table 6.5 shows the results of our experiments in the disjoint data scenario. We use the acronyms RS, INV and ST3 when referring to the Real Estate, Inventory

Table 6.5. Results: Disjoint Data Scenario

Matches	Accuracy
RS All 1-1 Matches	85%
RS String 1-1 Matches	100%
RS Numeric 1-1 Matches	0%
RS All Complex Matches	25%
RS String Complex Matches	60%
RS Numeric Complex Matches	0%
INV All 1-1 Matches	40%
INV String 1-1 Matches	100%
INV Numeric 1-1 Matches	0%
INV All Complex Matches	20%
INV String Complex Matches	56%
INV Numeric Complex Matches	0%
ST6 All 1-1 Matches	42%
ST6 String 1-1 Matches	100%
ST6 Numeric 1-1 Matchings	0%
ST6 All Complex Matches	100%
ST6 String Complex Matches	100%

and Synthetic 6 datasets, respectively. Our 1-1 match searcher missed all matches involving numeric and date schema elements (*age, birth dates, prices and discounts*) in all datasets. However, it was able to identify all 1-1 matches involving string schema elements in all datasets.

The complex match searcher found 60%, 56% and 100% of the complex matches, involving string schema elements, in the Inventory, Real Estate and Synthetic 6 datasets, respectively. Similarly to the results in the partially overlapped data scenario, most of the complex matches found involve addresses and the partial matches involve schema elements associated with personal names. Some examples of matches found are:

- Inventory dataset: $ship\text{-}address = (ship\text{-}address + ship\text{-}postal\text{-}code) + (ship\text{-}city + ship\text{-}country)$
- Real Estate dataset: $house\text{-}address = (house\text{-}street + house\text{-}city) + house\text{-}zip\text{-}code$
- Synthetic 6 dataset: $full\text{-}name = forename + surname$

Thus, as we can see, despite the fact that the repositories do not share any common data, our evolutionary approach was capable of successfully handling the information

extracted from the data instances in order to identify semantic relationship among the string schema elements existing in the repositories.

However, it should be noted that, when using our attributed-oriented strategy, it may be difficult to find a match in situations such as:

- *Type mismatch.* It is difficult to match semantic related schema elements that adopt different data types (e.g., boolean (true, false) and string (yes, no));
- *Numeric schema elements.* When numeric element schemas have different ranges (e.g., between 0 and 1 in one repository and between 10 and 1.000 in the other one) or any calculation is required for finding the matches (like in the case of numeric complex matches (Dhamankar et al., 2004)), our strategy is unable to identify any relationship between the schema elements;
- *Data classes with distinct data representations.* It is difficult to find matches between semantic related schema elements whose data classes adopt distinct data representations (e.g., months represented by their names in one repository and by numbers in the other one).

6.3 Remarks on the Results

Based on the results of our experiments, we can make the following remarks:

1. Our record-oriented strategy is effective in the partially and fully overlapped scenarios, being able to identify numerical matches in several cases (mostly in the fully overlapped scenario).
2. Both our record- and attribute-oriented strategies managed to find complex matches. Moreover, our attribute-oriented also found them in the disjoint data scenario. Thus, our approach is the first instance-based approach that achieves such results.

3. Although our attribute-oriented strategy can also be used in the partially and fully overlapped data scenarios, the record-oriented strategy is more effective on them, as we identified in preliminary experiments. This is explained by the fact that our record-oriented strategy does not face the difficulties the attribute-oriented strategy does when identifying numerical matches.
4. The results achieved by our schema matching strategy are similar to the ones reported by iMAP (Dhamankar et al., 2004). For the Real Estate dataset, iMAP and our record-oriented strategy found 58% and 20% (60% of the string matches) of all matches, respectively. In the Inventory dataset, both iMAP and our record-oriented strategy found 20% of all matches. These results are noticeable, since our strategy does not use any structural information other than the class of the schema elements, which can be easily determined by a simple data inspection.

Chapter 7

Conclusions

Data integration is one of the oldest topics in database research, however, there is not yet a “complete and single” solution for some problems related to it (e.g., schema matching and record deduplication). What makes these problems hard to solve are situations caused by semantic heterogeneity, such as erroneous relationships between schema elements and the existence of replicated data in repositories created by the aggregation of different sources. These situations usually require more storage devices and consume more processing power and energy for handling and managing information.

For these reasons, there has been a large investment from companies and government institutions on the development of effective solutions to these problems (Bell and Dravis, 2006; Wheatley, 2004). However, despite the fact that this investment is producing some results (as reported in a recent data quality survey (Bell and Dravis, 2006)), most of the existing solutions are based on approaches that are highly dependent of human supervision (e.g., tuning parameter values for commercial record deduplication systems) or are even manually crafted (e.g., a schema matching manually carried out by a database designer). As a consequence, these are labor intensive, very expensive, and error prone solutions.

In this thesis, we aimed at providing solutions to data integration related problems by means of evolutionary techniques. Our GP-based approach to record deduplication

and our evolutionary approach to schema matching are able to automatically provide efficient and high accuracy solutions by using the available resources more efficiently, besides being able to handling most of the problems related to parameter setup as demonstrated by our experiments.

7.1 Summary of Results

As the first contribution of this thesis, we proposed a GP-based approach to record deduplication. Our approach is able to automatically suggest deduplication functions based on evidence present in the data repositories. The suggested functions properly combine the best evidence available in order to identify whether two distinct record entries represent the same real-world entity.

As shown by our experiments, our approach achieves competitive results when compared with Marlin, a state-of-the-art SVM based system used as baseline. Moreover, our suggested functions use fewer evidence which means that they are computationally less demanding. Additionally, regarding our better results, unlikely most deduplication approaches described in the literature that use exactly the same similarity function for all available attributes, ours is capable of combining distinct similarity functions according to the attribute types. Thus, our approach is able to automatically choose the best function for each specific case, which is, certainly, one of the reasons of our better results.

Other significant result from our experiments is that our GP-based approach is able to automatically find suitable deduplication functions, even when the best evidence (similarity function and record attribute) is not previously known. This is useful for the non-specialized user, who does not have to set up the best evidence for the replica identification task. This is one of the main contributions of our approach, that distinguishes it from all previous proposals in the literature which require user-selected evidence for their initial setup.

We also presented the results of experiments on the replica identification boundary value, using real and synthetic datasets. Our experiments show that our GP-based approach is able to adapt the deduplication functions generated during the evolutionary process to different replica identification boundary values used to classify a pair as a match or not. Moreover, the results suggest that the use of a fixed and small lower boundary value eases the evolutionary effort and also leads to better solutions.

As our second contribution, we proposed a novel evolutionary approach to schema matching. Our approach is able to automatically find relationships between schema elements from two semantic related repositories. It adopts matching strategies that exploit record deduplication and information retrieval techniques to find complex schema matches using only the actual data stored in the repositories. To the best of our knowledge, our proposed approach is the first instance-level approach that deals with complex matches.

Our experimental results show that our evolutionary approach to schema matching is able to identify complex matches with high accuracy. In our experiments, it was able to find complex matches in three different data repository scenarios, when the repositories partially share some data, when the same data can be found in both repositories and when the repositories do not share any common data, achieving for string complex matches 100% of accuracy in synthetic datasets and up to 60% in real datasets.

7.2 Future Work

There are several possible research directions from the results presented in this thesis. Since our contributions address problems related to both the semantic and instance integration phases of the data integration process, we divided our future work suggestions, as follows:

- Extend the range of use of our GP approach for record deduplication by:

-
- Conducting additional experiments using different repositories (e.g., synthetic and real data) to find which situations (or scenarios) our proposed GP-approach would not be the most adequate alternative to the user. Since record deduplication is a very expensive and computationally demanding task, it is important to know what are the cases for which our approach would not be the most suitable alternative.
 - Improving the effectiveness of the solutions by employing clustering techniques to the final solution provided by the record deduplication process. This additional step might be able to find missing replicas that were not identified during the deduplication itself.
 - Improving the efficiency of the GP training phase by selecting the best training examples (Gonçalves et al., 2009). Selecting the most representative examples can minimize the training effort required by our GP-based approach without affecting the final solution quality.
 - Extend the range of use of our evolutionary approach to schema matching by:
 - Experimenting with datasets from other domains in order to evaluate how our proposed approach deals with additional repositories with different number of attributes and actual data characteristics (e.g., different data types and languages). Other repositories may require different tuning for the parameters in our attribute- and record-oriented strategies. Moreover, additional experiments may discover if these parameters can be automatically suggested, thus, freeing the user from the burden of tuning them.
 - Investigating other techniques, such as user-feedback, to improve schema matching results when dealing with numeric and date data classes, type mismatch situations, and also disjoint data repositories.
 - Investigating how our schema matching strategies can benefit from using additional information provided by existing metadata such as data frequencies,

average size of specific attribute instances, etc.

- Developing a schema level or hybrid evolutionary approach to schema matching based on the ideas proposed in our evolutionary instance-based approach. Instance-based approaches face difficulties on some scenarios, for instance, when handling numeric data, that can be overcome by using structural information. It is possible to modify the representation we proposed for the schema matching problem to combine evidence from both the actual data and the repository structure.

Bibliography

- Angeline, P. J. and Kinnear, Jr., K. E., editors (1996). *Advances in Genetic Programming*, volume 2. MIT Press.
- Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.
- Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming - An Introduction on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers.
- Batini, C., Lenzerini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computation Survey*, 18(4):323–364.
- Bell, R. and Dravis, F. (2006). Is you data dirty? and does that matter? Accenture Whiter Paper - <http://www.accenture.com>.
- Bergström, A., Jaksetic, P., and Nordin, P. (2000). Enhancing information retrieval by automatic acquisition of textual relations using genetic programming. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pages 29–32.
- Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.
- Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 39–48.

- Carvalho, J. C. P. and da Silva, A. S. (2003). Finding similar identities among objects from multiple web sources. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, pages 90–93.
- Castano, S., Antonellis, V. D., and di Vimercati, S. D. C. (2001). Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297.
- Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 313–324.
- Christen, P. (2005). *Probabilistic Data Generation for Deduplication and Data Linkage*. Lecture Notes in Computer Science. Springer.
- Cohen, W. W. (2000). Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321.
- Cohen, W. W. and Richman, J. (2002). Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 475–480.
- Cummins, R. and O’Riordan, C. (2006). Evolving local and global weighting schemes in information retrieval. *Information Retrieval*, 9(3):311–330.
- de Carvalho, M. G., Gonçalves, M. A., Laender, A. H. F., and da Silva, A. S. (2006). Learning to deduplicate. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 41–50.
- de Carvalho, M. G., Laender, A. H. F., Gonçalves, M. A., and da Silva, A. S. (2008a). Replica identification using genetic programming. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing*, pages 1801–1806.

- de Carvalho, M. G., Laender, A. H. F., Gonçalves, M. A., and Porto, T. C. (2008b). The impact of parameter setup on a genetic programming approach to record deduplication. In *Anais do XXIII Simpósio Brasileiro de Bancos de Dados*, pages 91–105.
- Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 383–394.
- Doan, A., Domingos, P., and Halevy, A. Y. (2001). Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 509–520.
- Doan, A., Domingos, P., and Levy, A. Y. (2000). Learning source description for data integration. In *Proceedings of the 2000 International Workshop on the Web and Databases*, pages 81–86.
- Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16.
- Fan, W., Fox, E. A., Pathak, P., and Wu, H. (2004). The effects of fitness functions on genetic programming-based ranking discovery for web search: Research articles. *Journal of the American Society for Information Science and Technology*, 55(7):628–636.
- Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of American Statistical Association*, 66(1):1183–1210.
- Gonçalves, G., de Carvalho, M. G., Laender, A. H. F., and Gonçalves, M. A. (2009). Seleção automática de exemplos de treino para um método de deduplicação de registros baseado em programação genética. In *Anais do XXIV Simpósio Brasileiro de Bancos de Dados*.

- Gordon, M. (1988). Probabilistic and genetic algorithms in document retrieval. *Communications of ACM*, 31(10):1208–1218.
- Guha, S., Koudas, N., Marathe, A., and Srivastava, D. (2004). Merging the results of approximate match operations. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 636–647.
- Hernández, M. A., Miller, R. J., and Haas, L. M. (2001). Clio: a semi-automatic tool for schema mapping. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, page 607.
- Hirsch, L., Hirsch, R., and Saeedi, M. (2007). Evolving Lucene search queries for text classification. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1604–1611.
- Kinnear, Jr., K. E. (1993). Generality and difficulty in genetic programming: Evolving a sort. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 287–294.
- Koudas, N., Sarawagi, S., and Srivastava, D. (2006). Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 802–803.
- Koza, J. R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press.
- Lacerda, A., Cristo, M., Gonçalves, M. A., Fan, W., Ziviani, N., and Ribeiro-Neto, B. (2006). Learning to advertise. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–556.
- Lawrence, S., Giles, C. L., and Bollacker, K. D. (1999a). Autonomous citation matching. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 392–393.

- Lawrence, S., Giles, L., and Bollacker, K. (1999b). Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71.
- Lee, Y., Sayyadian, M., Doan, A., and Rosenthal, A. S. (2007). eTuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal*, 16(1):97–122.
- Li, W.-S. and Clifton, C. (1994). Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 1–12.
- Li, W.-S. and Clifton, C. (2000). Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowledge and Engineering*, 33(1):49–84.
- Madhavan, J., Bernstein, P. A., Doan, A., and Halevy, A. (2005). Corpus-based schema matching. In *Proceedings of the 21st International Conference on Data Engineering*, pages 57–68.
- Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49–58.
- Masand, B. (1994). Optimizing confidence of text classification by evolution of symbolic expressions. pages 445–458.
- Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th IEEE International Conference on Data Engineering*, page 117.
- Miller, R. J., Haas, L. M., and Hernández, M. A. (2000). Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 77–88.

- Milo, T. and Zohar, S. (1998). Using schema matching to simplify heterogeneous data translation. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 122–133.
- Mitra, P., Wiederhold, G., and Jannink, J. (1999). Semi-automatic Integration of Knowledge Sources. *Proceedings of the Second International Conference on Information Fusion*.
- Mitra, P., Wiederhold, G., and Kersten, M. L. (2000). A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of the 7th International Conference on Extending Database Technology*, pages 86–100.
- Newcombe, H. B., Kennedy, J. M., Axford, S., and James, A. (1959). Automatic linkage of vital records. *Science*, 130(3381):954–959.
- Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.
- Sarawagi, S. and Bhamidipaty, A. (2002). Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–278.
- Tejada, S., Knoblock, C. A., and Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, 26(8):607–633.
- Tejada, S., Knoblock, C. A., and Minton, S. (2002). Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 350–359.
- Torres, R. d. S., Falcão, A. X., Gonçalves, M. A., Papa, J. P., Zhang, B., Fan, W., and Fox, E. A. (2009). A genetic programming framework for content-based image retrieval. *Pattern Recognition*, 42(2):283–292.

- Warren, R. H. and Tompa, F. W. (2006). Multi-column substring matching for database schema translation. In *Proceedings of the 32nd International Conference on Very large Data Bases*, pages 331–342.
- Wheatley, M. (2004). Operation clean data. *CIO Asia Magazine*, August 2004 - <http://www.cio-asia.com>.
- Yan, L. L., Miller, R. J., Haas, L. M., and Fagin, R. (2001). Data-driven understanding and refinement of schema mappings. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 485–496.
- Ziegler, P. and Dittrich, K. R. (2004). User-specific semantic integration of heterogeneous data: The Sirup approach. In *First International IFIP Conference on Semantics of a Networked World*, volume 3226 of *Lecture Notes in Computer Science*, pages 44–66. Springer.

