CHAPTER 3

DUPLICATE RECORD DETECTION USING GA AND PSO

3.1 INTRODUCTION

The present chapter extends the research discussed in chapter 2 by handling the optimization algorithms. Moises G. de Carvalho et al (2011) have proposed a genetic programming approach to record deduplication. This approach automatically proposes duplicate record detection function by combining several pieces of evidence taken from the data. This function makes it possible to identify whether the two records in a repository are the same or not.

Luís Leitão and Pável Calado (2011) have proposed an argument, that structure can have a significant impact on duplicate detection. They proposed a method that automatically restructures database objects to take advantage of the relations between its attributes. The new structure reflects the relative importance of the attributes in the database and avoids the need to perform a manual selection. They applied it to an existing duplicate detection system to test their approach. Experiments performed on several datasets reveal that by using the new learned structure, they consistently outperform both the results obtained with the original database structure and those obtained by letting a knowledgeable user manually choose the attributes for comparison.
Ektefa et al (2011) have proposed a threshold-based method which considers both string and semantic similarity measures for comparing record pairs. This method is experimented on Restaurant dataset and its effectiveness is measured based on several standard evaluation metrics.

Ye Qingwei Wu et al (2010) have proposed a new algorithm using PSO to search the optimized partial contents which is the most similar in two documents. It provides the encoding of the particles. A new related coefficient of strings is defined for strings similarity. Based on the related coefficient function, the new evaluation function of PSO is designed. The most similar partial contents are identified quickly and accurately by the hybrid mutation PSO algorithm. The simulation experiments indicate that the algorithm can effectively search the most similar partial contents in two documents.

Robert Isele and Christian Bizer (2011) addressed an important problem in the discovery of links between entities which identify the same real world object. These links are often generated based on manually written linkage rules which specify the condition which must be fulfilled for the two entities to be interlinked. This approach automatically generates linkage rules from a set of reference links and is based on genetic programming and has been implemented in the Silk Link Discovery Framework. It is capable of generating complex linkage rules which compare multiple properties of the entities and employ data transformations to normalize their values.

Junio de Freitas et al (2010) presents the Active Learning GP (AGP), a semi-supervised GP, and instantiates it for the data deduplication problem. AGP uses an active learning approach. In this process a group of multi-attribute functions votes for classifying record pairs as duplicates or not. When this group majority voting is not enough to predict the class of the data pairs, a user is called to solve the conflict. Experiments prove that AGP guarantees the quality of the deduplication while reducing the number of labeled examples needed.
Elhadi et al (2009) have proposed a method that reports on the experiments performed to investigate the use of a combined part of speech (POS) and an improved Longest Common Subsequence (LCS) in the analysis and calculation of similarity between texts. The text's syntactical structures were used as a representation for the documents. In order to compare and rank the documents according to the similarity of their representative string, an improved LCS algorithm was applied to such a representation. The approach was applied in detecting duplicate documents in a corpus, and also in the filtering of the search engine results.

In reference to the above mentioned approaches, the present research adopts the optimization algorithms such as GA and PSO for duplicate record detection. GA is an adaptive heuristic search algorithm that is based on the evolutionary ideas of natural selection and genetics and is used to solve optimization problems. The reason for using the proposed GA is to find suitable solutions to a given problem without searching the entire search space for solutions when there is more than one objective to be accomplished. This approach takes the concept of GA for deduplication from the existing one (Moises G. de Carvalho et al 2011) and it uses two different similarity metrics Levenstein and Cosine similarity to generate the feature vectors.

The present research also adopts PSO algorithm which is a population-based, stochastic and multi agent parallel global-search technique. Unlike the Genetic Algorithm, the PSO algorithm has no crossover and mutation operators. Since the problem deals with the feature vectors which generate a set of expressions, the proposed approach simply takes the concepts of PSO algorithm for generating the optimal similarity measure to decide whether the data is a duplicate or not. PSO algorithm is used to generate the optimal similarity measure for the training datasets. Once the optimal similarity measure is obtained, the duplicate detection of the
remaining datasets is done with the help of the same measure generated from the PSO algorithm.

The performance is evaluated under various thresholds on two different datasets Restaurant and Cora. The results of PSO from the evolution of the proposed approach are satisfactory as compared to the results provided by the Genetic Algorithm for the same set of input data.

3.2 GENETIC ALGORITHM

GA is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they denote an intelligent exploitation of a random search applied to solve the optimization problems. GA is by no means random, even though they are randomized. Instead they exploit historical information to direct the search in such a way to enhance performance within the search space. The basic methods of the GAs are designed to simulate processes in natural systems necessary for evolution especially the principles first laid down by Charles Darwin of "survival of the fittest".

3.2.1 Methodology

In this algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals) to an optimization problem, progresses towards better solutions. Usually, solutions are represented in binary as strings of 0’s and 1’s, but other encodings are also possible. The evolution generally starts from a population of randomly generated individuals and occurs in generations. The fitness of every individual in the population is evaluated in each generation. Several individuals are stochastically selected from the current population based on their fitness and modified recombined and possibly randomly mutated to form
a new population. In the iteration of the algorithm that follows, this new population is then used. The algorithm generally terminates when either a maximum number of generations has been produced, or an acceptable fitness level has been reached for the population.

A typical GA requires,

1. a genetic representation of the solution domain.
2. a fitness function to calculate the solution domain.

Normal representation of the solution is an array of bits. The important property that makes these genetic representations suitable is that their parts are easily aligned due to their fixed size, which simplifies crossover operations.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For example, in the knapsack problem an attempt is made to maximize the total value of objects that can be put in a knapsack of some fixed size. A representation of a solution might be an array of bits which takes the value 0 or 1 and each bit represents a different object. This value signifies whether or not the object is in the knapsack. Not all such representation is valid, as the size of objects may exceed the size of the knapsack. The fitness of the solution is calculated as the sum of values of all the objects in the knapsack if the representation is valid or otherwise zero. When it becomes challenging to define the fitness expression, interactive genetic algorithms are used.

Once the genetic representation and the fitness function are defined, GA proceeds to initialize a population of solutions randomly.
Figure 3.1 Process flow of GA
It is then improved through repetitive use of the mutation, crossover and selection operators which are described below.

3.2.1.1 Initialization

Initially several individual solutions are randomly generated to form an initial population. The nature of the problem determines the population size, but typically contains several hundreds or thousands of possible solutions. Usually, the population is generated randomly, allowing the entire choice of possible solutions (the search space). Rarely, the solutions may be scattered in areas where the optimal solutions are likely to be found.

3.2.1.2 Selection

During each successive generation, a proportion of the existing population is selected to breed a new generation. A fitness based process selects individual solutions. Fitter solutions (as measured by a fitness function) are typically more likely to be selected. The best solutions are preferentially selected by certain selection methods which rate the fitness of each solution. Other methods rate only a random sample of the population, as the former process may be very time-consuming. Some of the methods are,

Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes more are the chances of being selected.

Rank selection

Rank selection method initially ranks the population and then every chromosome receives fitness from this ranking. The worst rank will have
fitness 1, second worst 2 and so on. The best will have fitness N, where N is the number of chromosomes in a population.

**Elitism**

Elitism is the name of the method, which first copies the best chromosome to a new population. The rest is done in the classical way. The performance of GA can be rapidly increased by elitism because it prevents losing the best found solution.

### 3.2.1.3 Reproduction

Reproduction is to generate a second generation population of solutions from those selected through genetic operators such as crossover (also called recombination), and/or mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool previously selected. Using the above methods of crossover and mutation, a new solution is created by producing a "child" solution. The solution typically shares many of the characteristics of its "parents". New parents are selected for each new child. This process continues until a new population of solutions of appropriate size is generated.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Normally, the average fitness of the population will be increased by this procedure. This is because only the best organisms from the first generation are selected for breeding, with a small proportion of less fit solutions.
3.2.1.4 Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are,

- A solution which satisfy the minimum criteria
- Fixed number of generations is touched
- Allocated budget either computation time or money is reached
- The highest ranking solution's fitness is reached.
- Obtaining a solution such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

3.2.2 Operators of Genetic Algorithm

There are three common genetic operators namely, selection, crossover and mutation. It is not necessary to employ all of the three operators in a GA because each function is independent of the other functions. The choice or design of operators depends on the problem and the representation scheme employed.

3.2.2.1 Encoding of a chromosome

A chromosome should in some way contain information about the solution it represents. The most general way of encoding is a binary string. A binary string (16 bits) represents each chromosome. Each bit in the string can represent some characteristics of the solution. Another possibility is to use a
vector of integers or real numbers, with each integer or real number representing a single parameter.

### 3.2.2.2 Selection

The aim of the selection procedure is to reproduce more copies of the individual whose fitness values are higher. The selection procedure plays a vital role in significantly driving the search forward, producing good solutions in a short time.

### 3.2.2.3 Crossover

This operation makes the GA different from other algorithms, such as dynamic programming. It is used to create two new individuals (offspring) from two existing individuals (parents) that have been chosen from the current population by the selection operation. There are several methods of doing this. Two common crossover operations are one-point crossover and two-point crossover.

One-point crossover is the simplest crossover operation. From the pool of individuals, two individuals are randomly selected as parents formed by the selection procedure and cut at a randomly chosen point. The tails, the portions after the cutting point are swapped and the two new individuals are produced.

In two-point crossover, two random cutting points are chosen and the bits between the cutting points are exchanged across the two selected parents to form two new offspring.
3.2.2.4 Mutation

In mutation, all individuals in the population are checked bit by bit and the bit values are randomly reversed according to a specific rate. Unlike crossover, this is a monadic operation. That is, a child string is produced from a single parent. The mutation operator makes the algorithm to search for new areas. Eventually, it helps the GA in avoiding the premature convergence and also helps in finding the global optimal solution.

3.2.3 Simple Genetic Algorithm Procedure

1. Choose the initial population of individuals

2. Evaluate the fitness of each individual in that population

3. Repeat on this generation until termination (time limit, sufficient fitness achieved)
   (i) Select the best-fit individuals for reproduction
   (ii) Breed new individuals through crossover and mutation operations to give birth to offspring
   (iii) Evaluate the individual fitness of new individuals
   (iv) Replace least-fit population with new individuals

The general process of GA is shown in the Figure 3.1.

3.2.4 Advantages of Genetic Algorithm

GA has received considerable attention due to its potential as a novel optimization technique. There are three major advantages in applying Genetic Algorithms to optimization problems.
1. GA does not have much mathematical requirements. Due to their evolutionary nature, GA will search for solutions without looking to the specific inner working of the problem. It can handle any kind of objective functions and constraints (i.e. linear or non-linear) defined on discrete, continuous or mixed search spaces.

2. The evolution operators make GAs very effective at performing the global search.

3. GAs provides a great flexibility to hybridize with domain-dependent heuristics to make an efficient implementation for a specific problem.

### 3.2.5 Issues of Genetic Algorithm

GAs does not use much knowledge about the problem to be optimized and do not deal directly with the parameters of the problem. They work with the codes, which represent the parameters. Thus the first issue in a GA application is the way in which the problem parameters are to be denoted and the second issue is creation of the initial population of possible solutions. The third issue is selection of suitable genetic operators. Finally, as with the other search algorithms, GA has to know the quality of already found solutions to improve them further.

### 3.3 PARTICLE SWARM OPTIMIZATION

Russell Eberhart and James Kennedy invented PSO by the inspiration of the flocking and schooling patterns of birds and fish in 1995. Initially, they developed computer software simulations of birds flocking around food sources and then later functioned on optimization problems. PSO is an optimization technique which provides an evolutionary based search.
PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO optimizes a problem by taking a population of candidate solutions, and by moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position. It is also guided towards the best known positions in the search-space, which are updated as better positions. This is likely to move the swarm towards the best solutions.

PSO is metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. Unlike in the classic optimization methods such as Gradient Descent and Quasi-newton methods, the PSO does not require the optimization problem to be differentiable. This is because the PSO does not use the gradient of the problem being optimized. PSO can therefore be used on optimization problems that are partially irregular, noisy or change over time.

The PSO algorithm is a population-based, stochastic and multi agent parallel global-search technique. Unlike GA, the PSO algorithm has no crossover and mutation operators. The PSO algorithm is based on the mathematical modeling of various collective behaviors of the living creatures that display complex social behaviors.

In this algorithm, while a pattern (i.e., particle) is developing a new situation, both the cognitive component of the relative particle and the social component generated by the swarm are used. This situation enables the algorithm to effectively develop the local solutions into global optimum solutions. However, the algorithm is significantly affected by the initial values of the parameters used in the weighting of the cognitive and social components and the weighting strategy of the velocity vector.
3.3.1 Outline of Particle Swarm Optimization

The outline of PSO is stated as follows,

1. Create a ‘population’ of agents (called particles) uniformly distributed over X.

2. Evaluate each particle’s position according to the objective function.

3. If a particle’s current position is better than its previous best position, update it.

4. Determine the best particle (according to the particle’s previous best positions).

5. Update particles’ velocities according to,

\[ \text{velocity}_{ex} = \text{velocity}^0 + \phi (p_{best} - \text{pos}^0) + \phi (g_{best} - \text{pos}^0) \]

Where,

\[ \text{velocity}^0 \] = current velocity

\[ p_{best} \] = current best position

\[ \text{pos}^0 \] = current position

\[ g_{best} \] = global best position

\[ \phi, \phi \] = random values in range [0, 1]

6. Move particles to their new positions according to,

\[ \text{pos} = \text{pos}^0 + \text{velocity}_{ex} \]

(3.2)

7. Go to step 2 until stopping criteria are satisfied.
The algorithm keeps track of three global variables,

- Target value or condition
- Global best (Gbest) value specifying which particle is currently closest to the Target
- Stopping value indicating when the algorithm should halt if the target is not found

Each particle consists of,

- Data specifying a possible solution.
- A velocity value indicating how much the data can be changed.
- A current best (pBest) value indicating the closest the particle's data has ever come to the target.

![Figure 3.2 Population Topologies](image-url)
PSO algorithm uses population topologies, or neighborhoods. These neighborhoods could involve two or more particles which are predetermined to act together. Few common topologies are shown in the Figure 3.2.

(A) Single-sighted - Individuals compare themselves to the next best.

(B) Ring topology – In this, each individual compares only to those to the left and right.

(C) Fully connected topology - Everyone in this type is compared together.

(D) Isolated - Here individuals only compare to those within specified groups.

3.3.2 Advantages of Particle Swarm Optimization

1. Since PSO is based on the intelligence, it could be applied to both scientific research and engineering use.

2. There is no crossover and mutation calculation and the calculation is very simple in PSO.

3. PSO adopts the real number code, and it is decided directly by the solution.

4. PSO is efficient in global search.

5. PSO is the only algorithm that does not implement the survival of the fittest.
3.4 AN APPROACH TO DUPLICATE RECORD DETECTION USING GA

3.4.1 Similarity Computation for All Pair of Records

In this section, the similarity computation is carried out by finding the similarity functions on each record field. A comparison is made by each function between the similarity of each field with the other records and a similarity value for each field is assigned. Levenshtein distance, Cosine similarity are the two similarity measures used in this approach. The two measures are computed for all the attributes of record pairs as different similarity operations have varying significance in different domains.

3.4.2 Feature Vectors

Feature vectors represent the set of elements that are required for the detection of duplicate elements from the data repository. The vectors can be obtained from the processing of the two similarity measure values. In order to adapt similarity measures for each field of the database with respect to the particular data domain for attaining accurate similarity computation, the present research uses Levenshtein and Cosine similarity measures to generate the feature vector.

3.4.3 Feature Extraction

A set of expression is supplied as input to the GA. It finds better expressions among the supplied inputs.
Choose the initial population of individuals represented in the Table 3.1

**Table 3.1 Initial population**

<table>
<thead>
<tr>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a+b)+(c+d)</td>
</tr>
<tr>
<td>(a+b)*(c+d)</td>
</tr>
<tr>
<td>(a-b)+(c+d)</td>
</tr>
<tr>
<td>(a+b)*(c-d)</td>
</tr>
</tbody>
</table>

The variables ‘a’, ‘b’, ‘c’ and ‘d’ represent the similarity values obtained from the similarity calculations. According to the proposed approach, there will be four similarity values generated as per the Levenshtein distance function and the Cosine similarity functions. Every similarity measure generates two similarity values after dividing the records into two parts. With the help of these four similarity values ‘a’, ‘b’, ‘c’ and ‘d’, a set of expression has been developed for finding the duplicate or non-duplicate pairs. The expressions generated from the feature vector are subjected for fitness evaluation.

Evaluate the fitness of each individual in that population.

Every optimization program is bounded with some fitness functions, according to the nature and behavior of the inputs and outputs, the fitness functions also change. The value generated from the fitness function is called fitness value. The proposed approach uses accuracy as fitness function.

\[
\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{false positives} + \text{true negatives} + \text{false negatives}}
\]  

(3.3)
Fitness values are calculated for all expressions that are initially selected. This process is continued till the iteration specified by the user. The expression with the highest fitness value is selected as the best fit expression.

![Diagram of record similarity computation]

**Figure 3.3 Computation of record similarity**

3 New population generation

Choose the best-fit expression for reproduction. Crossover and mutation operations are applied to best fit expressions to generate new population. Individual fitness is calculated for the new population. Replace least fit expression with the new one.
Example.

Crossover

\[(a+b)*(c-d)\] , \[(a-b)*(c+d)\] \(\Rightarrow\) \[(a+b)*(c+d)\]

Mutation

\[(a+b)*(c+d)\] \(\Rightarrow\) \[(a+b)/(c-d)\]

4 Continue the process for n iteration and select the best expression which identifies the record duplication.

3.5 AN APPROACH TO DUPLICATE RECORD DETECTION USING PSO

Similarity Computation for all the pairs of records and the feature vector generation are carried out as discussed in the sections 3.4.1 and 3.4.2 respectively.

3.5.1 Feature Extraction

A set of expression is supplied as input to the PSO to find better expression among the supplied inputs. PSO consists of different phases of execution which are explained below.

- Population

The PSO algorithm starts with initializing the population which is a set of expression used for the validation of duplicates. The population is a swarm of expression specified by the user. The input can be represented as in the Table 3.1.
• **Fitness**

The proposed approach uses accuracy as fitness function. Fitness values are calculated for all the expressions that are initially selected. This process is continued till the iteration specified by the user. The expression with the highest fitness value is selected as the global best.

• **New population generation**

The new populations are generated for finding the best fit expressions among the other expressions in the population. In the proposed method, the velocity and the position of each value are updated according to the equation specified in the following example.

**Example** Evaluation function = \( \frac{(a - b)}{(c + d)} \)

Pbest = \((a + b) \times (c - d)\)

Gbest = \((a - b) \times (c + d)\)

Update the velocity for the above equation.

While considering the pbest and the evaluation function, \( \varphi \) gives the random position for an operator.

\((a + b) \times (c - d)\), \((a - b)/(c + d)\) \(\Rightarrow ^{+3}\) the best fit operator

Considering the gbest and the evaluation function,

\((a - b) \times (c + d)\), \((a - b)/(c + d)\) \(\Rightarrow ^{+1}\) the best fit operator.
Thus, the position can be updated, by placing the best fit operator in the evaluation function instead of the operator, which it is processing currently.

\[(a + b)/(c + d) \rightarrow *' \rightarrow \text{the best fit operator.}\]

New solution \[(a + b)*(c + d)\]

Fitness of the new solution is calculated and if the solution has better fitness value than the pbest, it is then considered as the pbest and if it is better than the gbest, the solution is then considered as gbest.

- **Optimization**

In this phase, each population generated is iterated for the fitness evaluation. According to the PSO algorithm, a fitness parameter is set for filtering particles with the best fitness value. After the full execution of the process, the algorithm comes up with a number of solutions. The solutions are filtered out and the best expression with high fitness value is selected as the best solution for determining the duplicates.

- **Termination**

Termination criteria for the proposed approach are set by the user itself. The termination criteria will be the number of iterations needed for getting the best result from the population. Every time a program is executed, number of results will be generated. As the termination criterion is reached, as per the algorithm, it will be providing the best solution.
3.5.2 Duplicate Detection using Optimized Expression

Once the optimal expression is generated from the PSO, the same expression is used to find the duplicate or non-duplicate records. Here, the threshold $T$ is fixed to find the margin between the duplicate and non-duplicate pairs.

3.6 RESULTS AND DISCUSSION

The proposed approach deals with the duplicate record detection based on optimization based algorithms such as GA, PSO. The performance of the proposed approach is evaluated in the following section under different evaluation criteria on two different datasets Restaurant and Cora.

3.6.1 Accuracy Based Analysis on Restaurant Dataset

The graphs plotted below show the accuracy percentage of the proposed algorithm and the GA on the basis of the number of iterations under various thresholds $T_1$, $T_2$ and $T_3$ on Restaurant Dataset.

**Experiment 3.1:** The Table 3.2 and the Figure 3.4 shows that the proposed algorithm outperformed GA on Threshold $T_1$. The accuracy of PSO on different iterations is 6% more than the GA.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70.9</td>
<td>76.8</td>
</tr>
<tr>
<td>10</td>
<td>70.9</td>
<td>79.6</td>
</tr>
<tr>
<td>100</td>
<td>76.8</td>
<td>80.6</td>
</tr>
</tbody>
</table>
Figure 3.4 Accuracy based on threshold T1 for Experiment 3.1

**Experiment 3.2:** The Table 3.3 and the Figure 3.5 shows that the proposed algorithm outperformed GA on Threshold T2. The accuracy of PSO on different iterations is 5% more than the GA.

Table 3.3 Accuracy based on Threshold T2 for Experiment 3.2

<table>
<thead>
<tr>
<th>Iterations</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>71.9</td>
<td>76.9</td>
</tr>
<tr>
<td>10</td>
<td>73.9</td>
<td>80.6</td>
</tr>
<tr>
<td>100</td>
<td>77.8</td>
<td>81.6</td>
</tr>
</tbody>
</table>

Figure 3.5 Accuracy based on Threshold T2 for Experiment 3.2
Experiment 3.3: The Table 3.4 and the Figure 3.6 shows that the proposed algorithm outperformed GA on Threshold T3. The accuracy of PSO on different iterations is 7% more than the GA.

Table 3.4 Accuracy based on Threshold T3 for Experiment 3.3

<table>
<thead>
<tr>
<th>Iterations</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60.9</td>
<td>69.6</td>
</tr>
<tr>
<td>10</td>
<td>65.9</td>
<td>72.8</td>
</tr>
<tr>
<td>100</td>
<td>71.8</td>
<td>76.6</td>
</tr>
</tbody>
</table>

Figure 3.6 Accuracy based on Threshold T3 for Experiment 3.3

3.6.2 Accuracy Based Analysis on Cora Dataset

The graphs plotted below are the accuracy percentage of the proposed algorithm and the GA on the basis of the number of iterations under various thresholds T1, T2 and T3 on Cora dataset.

Experiment 3.4: The Table 3.5 and the Figure 3.7 shows that the proposed algorithm outperformed GA on Threshold T1. The accuracy of PSO on different iterations is 2% more than the GA.
Table 3.5 Accuracy based on Threshold T1 for Experiment 3.4

<table>
<thead>
<tr>
<th>Iteration</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80</td>
<td>79</td>
</tr>
<tr>
<td>10</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>100</td>
<td>85</td>
<td>92</td>
</tr>
</tbody>
</table>

Figure 3.7 Accuracy based on Threshold T1 for Experiment 3.4

Experiment 3.5: The Table 3.6 and the Figure 3.8 shows that the accuracy of PSO on different iterations is 2% less than GA.

Table 3.6 Accuracy based on Threshold T2 for Experiment 3.5

<table>
<thead>
<tr>
<th>Iteration</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>10</td>
<td>85</td>
<td>81</td>
</tr>
<tr>
<td>100</td>
<td>89</td>
<td>90</td>
</tr>
</tbody>
</table>
Experiment 3.6: The Table 3.7 and the Figure 3.9 shows that the accuracy of PSO on different iterations is 2% more than GA.

Table 3.7 Accuracy based on Threshold T3 for Experiment 3.6

<table>
<thead>
<tr>
<th>Iteration</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>10</td>
<td>81</td>
<td>80</td>
</tr>
<tr>
<td>100</td>
<td>88</td>
<td>94</td>
</tr>
</tbody>
</table>

Figure 3.8 Accuracy based on threshold T2 for Experiment 3.5

Figure 3.9 Accuracy based on Threshold T3 for Experiment 3.6
3.6.3 Experimental Results

In the Table 3.8, the solution or expressions are arranged based on the best fitness value of a particular expression for a threshold T1. Expressions are listed for different number of iterations of the two different algorithms. From the Table 3.9, it is clear that, the proposed PSO algorithm has the upper hand over the existing GA.

**Table 3.8 Best fit solutions for GA and PSO**

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Algorithm</th>
<th>Fitness values</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PSO</td>
<td>0.76</td>
<td>'(a-c)*(b+d)'</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.70</td>
<td>'(a-b)+(c+d)'</td>
</tr>
<tr>
<td>10</td>
<td>PSO</td>
<td>0.79</td>
<td>'(a*c)+(b+d)'</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.70</td>
<td>'(a+b)-(c+d)'</td>
</tr>
<tr>
<td>100</td>
<td>PSO</td>
<td>0.80</td>
<td>'(a-c)+(b+d)'</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>0.76</td>
<td>'(a+b)+(c+d)'</td>
</tr>
</tbody>
</table>

**Table 3.9 Top obtained solutions sorted based on the fitness value for GA and PSO**

<table>
<thead>
<tr>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>'(a+b)+(c+d)'</td>
<td>'(a-c)+(b+d)'</td>
</tr>
<tr>
<td>'(a+b)-(c+d)'</td>
<td>'(a*c)+(b+d)'</td>
</tr>
<tr>
<td>'(a-b)+(c+d)'</td>
<td>'(a-c)*(b+d)'</td>
</tr>
<tr>
<td>'(a+b)+(c-d)'</td>
<td>'(a-c)+(b*d)'</td>
</tr>
<tr>
<td>'(a+b)-(c-d)'</td>
<td>'(a-c)-(b*d)'</td>
</tr>
</tbody>
</table>
3.7 SUMMARY

The deduplication has been one of the most emerging techniques for data redundancy and duplication. The duplication creates lots of problems in the information retrieval system. The approach adopts GA for duplicate record detection. The similarity computation is carried out for all pairs of records using Levenshtein distance and Cosine similarity measures. GA is applied to extract the features by initializing the population with a set of expressions. The accuracy is calculated for each expression. Mutation and crossover operations are carried out to find the fittest expression among the expressions in the population. Finally, the best expression with a high fitness value is chosen as the best solution for determining the duplicates.

The present research also adopts PSO algorithm. Initially swarm of particles (set of expressions) is specified by the user. The fitness of each such particle is evaluated. Velocity and the position of the particle are updated using the particle best and global best parameters to obtain new solutions. Finally, high fitness value solution is chosen as the best for determining the duplicates. The experimentation showed that PSO provides better performance and accuracy than GA.