

10th National Convention on Statistics (NCS)
EDSA Shangri-La Hotel
October 1-2, 2007

A Genetic Algorithm for Constrained Statistical Matching

by

Giovanni A. Flores and Eliezer A. Albacea

For additional information, please contact:

Author's name : Giovanni A. Flores
Designation : Assistant Professor
Affiliation : Institute of Computer Science
Address : University of the Philippines Los Baños, College, Laguna
Tel. no. : (06349) 536-2313 / (06349) 536-2302
E-mail : gaflores@uplb.edu.ph

Co-author's name : Dr. Eliezer A. Albacea
Designation : Professor
Affiliation : Institute of Computer Science
Address : University of the Philippines Los Baños, College, Laguna
Tel. no. : (06349) 536-2313 / (06349) 536-2302
E-mail : eaalbacea@uplb.edu.ph

A Genetic Algorithm for Constrained Statistical Matching by

Giovanni A. Flores and Eliezer A. Albacea¹

ABSTRACT

Constrained statistical matching is a process of merging record instances between two source files such that no record instance would be used in a match more than once and that the overall distance cost of all record-pairs (matches) is minimized. This study presents a Genetic Algorithm (GA) as an alternative to a greedy algorithm of record linkage performed by sequentially finding the cheapest match for every record in one file with any of the remaining unmatched records in the other. The GA, designed initially as a solution to a special case of Linear Transportation Problem of Operations Research, used an adjacency list as a chromosomal representation, a population size of 40, a 90% selection probability, and a 50% crossover and mutation probabilities combined. The resulting computer programs were made to run on varying problem sizes mostly to verify GA's convergence to possibly global minimum-cost solution, and empirical results confirmed its ability in obtaining, under certain conditions, the global minimum, a situation that can not be arrived at via a straight-forward greedy algorithm. In all cases, however, GA – which entails creation and maintenance of a set of solutions (populations) per iteration – showed slower performance compared with greedy algorithm as the number of records to match is increased significantly. Finally, the strengths of both algorithms were combined experimentally, resulting to better performance and quality of solution, and the test results showed that the matching costs found by greedy algorithm were reduced significantly by a GA that use seeding process via a greedy heuristic, i.e. from 411 down to 375, 541 to 496, 680 down to 627, and 15379 to 15156, for record sizes 100x300, 150x450, 200x600, and 19195x39615, respectively. Hence, this investigation provides compelling evidence of GA's ability to find near-optimal solution, which cannot be obtained by traditional greedy search algorithm.

I. Introduction

Situations arise when variables contained in a survey data file are insufficient for a planned statistical analysis. Instead of conducting another expensive data gathering that would include all needed variables, the statistician may choose to get the needed information from an existing data panel to augment the former. That is, a *recipient* data file which contains the sets of variables XY may be merged with a *donor* data file containing the sets of variables XZ, to generate a new data panel now containing the three sets of variables XYZ suitable for the required analysis (X being the set of matching variables). Inevitably, both data files might not share the same set of respondents/units (or the matching variables themselves do not uniquely identify a unit, unlike what Tax Identification Number (TIN) does, for example) making *exact matching* impossible. Hence, *statistical matching* might be performed to approximate a match between two different units yet judged to be 'similar' by

¹ Professor and Assistant Professor, respectively, of the Institute of Computer Science, University of the Philippines at Los Baños.

computing the discrepancies of their matching variables using a distance function. Constrained Statistical Matching (CSM) simply ensures that a donor record get matched with a recipient record exactly once.

Constrained Statistical Matching, therefore, *is* a minimization problem that aims to find the mapping of the recipient records to the donor records which has the lowest overall record-pair discrepancies. One possible approach is to use a greedy approach which pairs a recipient record with an unmatched donor record that would cause the lowest distance cost from it, in a pure sequential fashion, i.e. from the first recipient record down to last record (Albacea et al., 2003). This approach, however, doesn't guarantee a minimum cost mapping.

This study investigates the possibility of using a Genetic Algorithm (GA) as alternative methodology to carry out Constrained Statistical Matching as a minimization problem. Empirical results of GA with respect to the overall mapping cost are analyzed and are compared with the results obtained by a greedy algorithm. Finally, this study explores the possibility of merging the Genetic Algorithm with the greedy approach to improve the efficiency of its search mechanism.

This paper discusses the following subtopics in this order: Section II presents the Constrained Statistical Matching problem, along with the traditional greedy approach to solve it; Section III introduces the mechanism of Genetic Algorithm, and how it is applied to two optimization problems which this study is linking Constrained Statistical Matching to; Section IV takes the models established in Section III to formally design the Genetic Algorithm for Constrained Statistical Matching; Section V gives the results (and discussion) of the experiment; and Section VI summarizes the findings of this study.

II. Constrained Statistical Matching

File merging is a process in which records from two different statistical data files are combined into a single file based on a set of variables (matching variables) common in the two files (Paass, 1986; Soong et al., 2001; Yoshizoe, 1999). It could be classified into two categories: *exact matching* and *statistical matching*. Exact matching links the two files based on unique identifiers, such as address, or social security number present in both files, whereas statistical matching is used when it is impossible to perform exact matching due to absence of unique identifiers, or when the files are sampled from different sources (Cox et al., 1985). To perform statistical matching, one file should be designated as the recipient (or

receiver/base) file, whereas the other file, the donor (or reference/non-base) file, as illustrated in the following example by Barry (1988):

“Consider two data files of measurements taken from the same population [...] whose variables are defined to be of type X, Y, or Z. One file, the [recipient] file, contains information on X and Y variables only. The other file, the donor file, has information on X and Z variables only. That is, both files contain the same X variables whereas the Y and Z variables are each unique to one of the two files. [X as matching variables, Y and Z as non-matching variables]. Statistical matching is to be used to match each case [or record] in the [recipient] file with the one most similar to it in the donor file. Determination of ‘similar’ is by comparison of the common X, or matching, variables in the two files. For example, a donor-person with $X_1 = 65$ years and $X_2 = 6$ children might be considered as the nearest match [most similar] for a [recipient]-person with $X_1 = 64$ years and $X_2 = 7$ children. When a match has been made in this way, the Z measurements for the case in the donor file are imputed to its ‘twin’ in the [recipient] file. This matching and imputing procedure is repeated for all cases in the [recipient] file. The resulting file, containing information on X, Y and imputed Z(Z’) variables will be referred to as the statistically matched file.”

To measure the similarity of cases/records between the recipient and donor files, *distance-function* statistical matching may be employed which calculates the discrepancies or *distances* of the values of the matching variables between the records. So to find a match for a recipient record r , one should choose the record from the donor file whose matching variables were the *least* distance from those belonging to r (Albacea et al, 2003; Paass, 1985; Tena, 2002; Barry, 1988). Yoshizoe et al. (1999) presents different distance metrics for statistical matching. An example of a distance function on a set of six matching variables is given below:

Matching Variables:

1. SEX (sex of household head)
2. AGE (age of household head)
3. MS (marital status of household head)
4. HGC (educational attainment of household head)
5. FSIZE (family size)
6. URB (urbanity)

Distance Function:

$$\text{dist}(R[i], D[j]) = (\text{SEX}_{R[i]} - \text{SEX}_{D[j]})^2 + (\text{AGE}_{R[i]} - \text{AGE}_{D[j]})^2 + (\text{MS}_{R[i]} - \text{MS}_{D[j]})^2 + (\text{HGC}_{R[i]} - \text{HGC}_{D[j]})^2 + (\text{FSIZE}_{R[i]} - \text{FSIZE}_{D[j]})^2 + (\text{URB}_{R[i]} - \text{URB}_{D[j]})^2$$

where: $R[i]$ represents the i^{th} record in recipient file R, and $D[j]$ the j^{th} record in donor file D.

If each record in the donor file can be used several times as a match to a recipient record, then it is called *unconstrained statistical matching* (Barry, 1988). Otherwise, it is referred to as constrained statistical matching which ensures that a donor record get matched to recipient record exactly once.

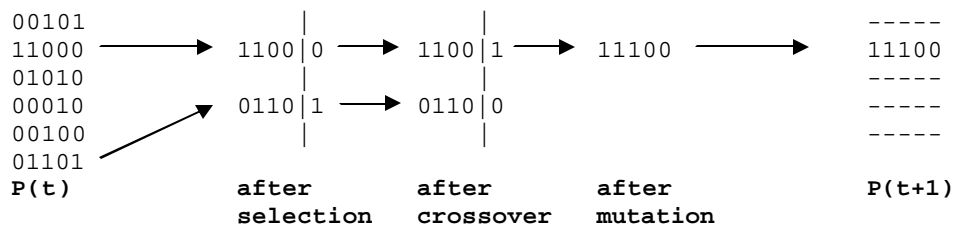
Alegre et al. (2000) outlines empirical steps of a typical statistical matching process:

1. Choosing which file is the recipient, and vice versa. The donor file should represent the reference information that will be added to the recipient file
2. Adjusting files and variable transformations
3. Choosing the matching variables – this is important because the accuracy of the match, and eventually the sort of analysis to be followed up, is extremely dependent on the quality of the matching variables chosen
4. Selection of the matching method – constrained or unconstrained – and the distance function to be applied
5. Validation of the statistically matched file

Constrained Statistical Matching is a minimization problem, whose main objective is to determine the minimum cost mapping of all recipient records to donor records. One straight-forward method to perform constrained statistical matching with the aim of minimizing the overall record-pair discrepancies is by successively matching (from the first down to the last record) a recipient record to an unmatched donor record with the least distance cost from it. This algorithm is referred to as the greedy algorithm, and is outlined in (Albacea et al., 2003). It should be easy to verify that this approach doesn't give a guarantee of finding the minimum cost mapping—for a different ordering of the recipient records would give a different mapping, in turn, a different overall cost. The main advantage of this approach, however, is a fast running time proportional to the product of the sizes (number of records) of the two files.

III. Genetic Algorithm and Related Problems

Genetic Algorithm (GA) is one search technique that has been used extensively on problems whose best algorithms to-date to determine the optimal solution have a running time complexity growing exponentially with the problem size (De Jong et al., 1989; Michalewicz, 1996). Unlike greedy algorithm, which searches for a solution via a point-to-point (single-solution) approach, GA is a population-based metaheuristics which searches for the optimal solution from a collection/population of candidate solutions (Talbi, 1999). This population is subjected to transformations influenced by the principle of natural selection and evolution during which individual solutions strive for survival. To produce (hopefully) better or fit individuals, solutions are recombined using the artificial genetic operators: selection, crossover, and mutation. Below illustrates how GA searches for the optimal solution with the aid of the genetic operators mentioned. The problem aims to maximize the function $f(x) = x^2$, where $0 \leq x \leq 31$, by searching for the largest value of x . This is a trivial problem that doesn't need a GA-like approach, yet its simplicity shows how GA searches for the solution from a population of candidate solutions:



The solutions (chromosomes) to this problem are integers encoded using fixed-length bit-strings. In the selection process, bias is given towards high-ranking chromosomes. (An objective function $f(x) = x^2$ is used to evaluate the fitness value of a chromosome, where x is the decoded value of the chromosome.) So in the example above, chromosomes 11000 (decimal 24) and 01101 (decimal 13) of the current population at time t , $P(t)$, are selected for further transformations. Next, a crossover point common in the two parent chromosomes is established randomly dividing a chromosome into two substrings. This is where the parent chromosomes exchange substrings to perform crossover (i.e. substring 1 of parent 1 pairing with parent 2's substring 2, whereas parent 2's substring 1 with parent 1's substring 2). After the crossover operation, two new offsprings are generated (chromosomes 11001 and 01100, the decimal 24 and 12, respectively). Then a mutation operation is performed on a resulting chromosome by randomly choosing two bit positions to be interchanged (e.g. bit 1 at position

0 exchanges with bit 0 at position 2 of chromosome 11001). Finally, the resulting mutated chromosome 11100 (decimal 28) is inserted into the pool of new candidate solutions to comprise the new population $P(t+1)$.

Below is the pseudocode of a standard Genetic Algorithm that uses fixed-length bit-string to encode the solution adapted by the problem above. When the loop eventually terminates, the final population is expected to contain (hopefully) the target solution sought for.

```
procedure GA
begin
    t <- 0
    initialize P(t);
    evaluate P(t);
    while (not terminating-condition) do
    begin
        t <- t + 1
        select P(t) from P(t-1)
        alter P(t)
        evaluate P(t)
    end
end
```

3.1. Traveling Salesman Problem (TSP)

Two combinatorial minimization problems that have been addressed using genetic optimization (and as we shall see later are very much related to Constrained Statistical Matching problem) are the Traveling Salesman Problem (TSP) and the Balanced Linear Transportation Problem (BLTP). In TSP, the goal is for a traveler to visit a number of cities, with the constraint that every city must be visited exactly once, and that the total distance cost is minimized. To date, enumeration of all possible routes is the only known algorithm that guarantees the optimal solution, with running time complexity that is exponential in nature (Garey et al, 1979). So for a problem with n cities, the number of possible tours is $n!/2n$ (the starting city and the direction of travel are irrelevant), which makes this technique of enumeration impractical for large values of n . Several algorithms with polynomial running time complexities have been suggested to approximate *near*-best solutions (Garey et al, 1979; Goldberg, 1989; Falkenauer, 1997; Michalewicz, 1996). Using a GA approach, the

chromosome may be represented using the path notation, e.g. 5-2-1-3-4, for a 5-city TSP. This solution encoding is accompanied by permutation-specific genetic operators like Partially-Matched Crossover (PMX), Cycle Crossover, Position Crossover, Scramble mutations, etc. (Goldberg, 1989; Whitley et al., 1995; Michalewicz, 1996).

3.2. Balanced Linear Transportation Problem (BLTP)

In BLTP, on the other hand, the objective is to ship the supply of homogenous units of commodities stored in M source points to meet the demands in N destination points with minimum total shipping cost, mathematically expressed as (Dean et al., 1992):

$$\begin{aligned} & \text{minimize } \sum_{i=1}^m \sum_{j=1}^n \text{cost}(i, j) \cdot (x_{ij}) \\ & \text{subject to:} \\ & \sum_{j=1}^n x_{ij} = \text{sour}(i), \text{ for } i = 1, 2, \dots, m, \\ & \sum_{i=1}^m x_{ij} = \text{dest}(j), \text{ for } j = 1, 2, \dots, n, \\ & x_{ij} = 0, \text{ for } i = 1, 2, \dots, m \text{ and } j = 1, 2, \dots, n. \end{aligned}$$

Figure 1. The Linear Transportation Minimization Problem

Note: $\text{cost}(i, j)$ represents the cost of transporting some (or all) of the supplies in the i^{th} source point to satisfy the some (or all) of the demands in the j^{th} destination point. x_{ij} is the amount of supplies shipped to satisfy a demand. The first of the two constraints ensures that the total amount shipped from a particular source point doesn't exceed the total amount of supplies available for that particular source point, whereas for the second constraint, the demand for each destination point should be satisfied with an exact number of shipped commodities. Finally, it is a balanced transportation problem with the total supply equals the total demand. A genetic algorithm designed to solve this problem was detailed in (Michalewicz et al., 1991). The chromosomal representation of the solution was an $M \times N$ matrix, accompanied by matrix-specific crossover and mutation operators inspired by the stepping stone algorithm of linear programming.

IV. Methodology

Any algorithm that attempts to find the optimal solution via brute-force enumeration of all solution points in the search space will eventually experience computing performance degradation as the problem size increases significantly (running time increases exponentially with the problem size). The use of an alternative methodology, though doesn't guarantee an optimal solution but a near-optimal one obtainable within a reasonably fast polynomial running time, would be an acceptable approach indeed. While greedy approach does outperform exhaustive enumerative search in terms of computing time, one inquiry remains to be addressed: could we improve on greedy's approach? Is there a way by which we could refine the search further to make the solution as close to the real optima as possible, even at the expense of extra (polynomial) computing time?

4.1. Modeling approach

If we are to enumerate all possible mappings of recipient to donor records, then we are assured of getting the global minimum solution. Assuming N_{RF} and N_{DF} represent the number of records contained in the recipient and donor files, respectively (where $N_{RF} = N_{DF}$), then using the exhaustive search algorithm the solution is surely one of the different mapping permutations of $N_{DF}! / (N_{DF} - N_{RF})!$. Hence, this study using a GA approach aims at finding this global minimum cost matching (or the near optimal one) without the explicit enumeration of the search space. Specifically, it aims to verify if GA's results would be comparable to those found by greedy algorithm for this type of problem.

To model the Genetic Algorithm for Constrained Statistical Matching, we'll use the two problems previously mentioned (TSP and BLTP) as guide in the design process. Not only do they provide hints on choosing GA as the approximation tool, but also serve as the very foundation of our design. Simply put, we'll show how to transform Constrained Statistical Matching into both problems (or near to them) and to adapt whatever these template problems have to offer to address the needs specific to Constrained Statistical Matching.

4.2. CSM and TSP

Assume a 3-record recipient file to be statistically matched to a 6-record donor file. A solution to this problem is a mapping of the three recipient records/indices (1, 2, 3) to any

three donor records/indices (1, 2, 3, 4, 5, 6, 7, 8, 9); for example, the recipient indices 1, 2, 3 matching with the donor indices 5, 1, 4, respectively. Notice that the latter set of indices represents one of the possible unique orderings (or permutations) of donor records, specifically one of the possible permutations of 3 donor records selected from a set of 9 records, i.e. $P(6, 3)$. In this regard, CSM could be viewed as a traveling salesman problem where the cities to be visited correspond with the donor records to be matched with, and as such any known TSP algorithm could be adapted with ease to address the CSM problem only after taking the following considerations:

- a) A TSP is always a “complete” tour/ordering of the cities; whereas a CSM, with $N_{RF} < N_{DF}$, is a *partial* ordering of the donor records.
- b) In TSP a tour is evaluated by adding up the distances between all adjacent cities in the ordering; whereas in CSM, what is being summed up are the discrepancies of the donor records in the partial ordering from the list of recipient records. So for example, if we have an ordering of 5-1-4-..., in TSP it is evaluated as distance(5, 1) + distance(1, 4) + distance (4, ...; whereas in CSM, it is distance(1, 5) + distance (2, 1) + distance(3, 4) + (4,

4.3. CSM and BLTP

Constrained Statistical Matching could also be transformed into the Balanced Linear Transportation Problem (BLTP). To illustrate, we'll use the same scenario above of matching a 3-record recipient file with a 6-record donor file, where M source points correspond with N_{DF} , and N destination points with N_{RF} . For each source point, there would be exactly 1 unit of supply, whereas for each destination point $6/3 = 2$ units of demand. (Note: total supply is equivalent to the total demand, hence a balanced transportation problem). One possible representation of the solution to this problem is depicted below:

		Demand values:			2	2	2	
			1	2	3	N destination pts		
Supply values:	1	1	0	1	0			
	1	2	0	0	1			
	1	3	1	0	0			
	1	4	0	0	1			
	1	5	1	0	0			
	1	6	0	1	0			
		M						
		Source pts						

Figure 2. Matrix data structure for LTP

The 6x3 matrix contains the amount of supplies shipped from M=6 source points to satisfy the demand in N=3 destination points. Now, to evaluate this matrix from BLTP's standpoint, we'll use the formula in Figure 1, with the knowledge that $\text{cost}(i, j) = \text{dist}(R[i], D[j])$, giving us the following:

$$\text{Evaluation cost} = \text{cost}(1, 3) + \text{cost}(1, 5) + \text{cost}(2, 1) + \text{cost}(2, 6) + \text{cost}(3, 2) + \text{cost}(3, 4)$$

Equation 1. The BLTP cost evaluation

Yet, in the context of CSM, the overall cost is expressed in Equation 2, which ensures that a recipient record is matched to a single donor record only:

$$\begin{aligned} \text{CSM cost} = & \text{minimum between cost}(1, 3) \text{ and cost}(1, 5) + \\ & \text{minimum between cost}(2, 1) \text{ and cost}(2, 6) + \\ & \text{minimum between cost}(3, 2) \text{ and cost}(3, 4) \end{aligned}$$

Equation 2. The CSM cost evaluation

4.4. Final GA Model

Though problem transformation (from CSM to either of TSP or BLTP) does provide us with the immediate solution to our problem (by adapting the known solutions for the later problem), there are at least two reasons that hinder us from directly adapting these two problems to model the constrained statistical matching problem:

1. The TSP model, as emphasized previously, represents a *complete* permutation of the donor records, when in reality the number of recipient records seldom matches with the number of donor records. Hence, for CSM problem, the solution is merely a *partial* permutation of the donor records. Using the existing GA implementation for TSP to solve the CSM problem may not prove effective. (A case study in line with this has been done and after applying the different operators specific to the path representation of the chromosome, no performance improvement was observed, so it was conjectured that the chromosomal representation, after all, may not be appropriate for the current problem.)

2. While the BLTP model captures the semantics behind the CSM problem better than the TSP model, the current implementation of GA for BLTP (Michalewicz et al., 1991) when adapted for CSM incurs unnecessary memory space overheads, as depicted in Figure 2. (There are significantly more zero entries than 1's).

To address these issues, we start with a modification of the BLTP model by changing its matrix representation to a more space-saving data structure. Notice that in Figure 2, the matrix stores only two type of values, either 0 or 1. This representation neatly resembles the adjacency matrix of graph problems. This adjacency matrix was then transformed into an adjacency list representation as shown in Figure 3. It is interesting to note that this modification in our BLTP model results to an alternative way of expressing the other model of concern, the TSP model: this time, instead of ordering the cities (donor indices) in a linear fashion, we have a permutation of objects across a two-dimensional data structure. Moreover, this new representation expresses Equation 2 in a more intuitive way to compute for the matching cost of recipient-to-donor record mapping. Finally in our adjacency list representation, choosing for each destination point one source point with the least distance cost from it (as dictated by Equation 2), results to a subset of M source points, specifically a *partial* permutation of M source points which is precisely the representation were looking for from TSP that aims to model the Constrained Statistical Matching problem with $N_{RF} < N_{DF}$.

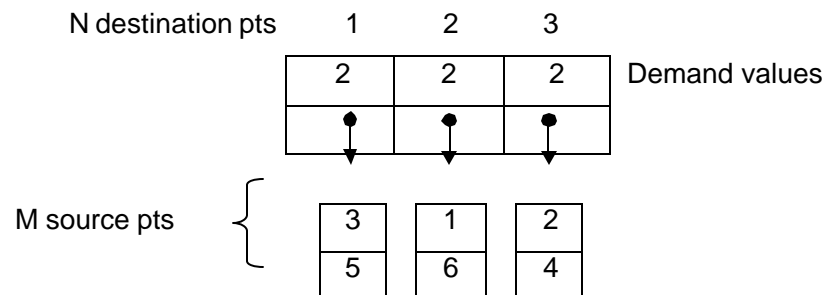


Figure 3. Adjacency list representation

Here's now the resulting design of the Genetic Algorithm:

1. chromosomal representation – adjacency list
2. evaluation function – CSM's distance function formula
3. genetic operators (selection, crossover, mutation)
 - a) Selection: The sampling mechanism is a mixture of dynamic, preservative, generational, and elitist models (Michalewicz, 1996), and is given below:
 - i. Select r parents from $P(t)$. Each selected chromosome is marked as applicable to exactly one fixed genetic optimization.

- ii. Select $\text{pop_size} - r$ distinct chromosomes from $P(t)$ and copy them to $P(t+1)$.
 - iii. Let r parent chromosomes breed to produce exactly r offspring. Insert these r new offspring into population $P(t+1)$.
- b) Crossover (modified version of crossover for BLTP): For simplicity of presentation, the steps below pertain to the original matrix representation:
- Input: 2 parent chromosomes P1 & P2 (see Figure 4)
- Output: 2 unaltered parent chromosomes P1 & P2, with 2 resulting children chromosomes C1 & C2
- i. Create DIV matrix which would contain overlap entries of bit 1 between parents P1 and P2. (Figure 5)
 - ii. Create REM matrix which would contain non-overlapping entries of bit 1 between parents P1 and P2. (Figure 5)
 - iii. Partition REM matrix into two resulting matrices: REM1 and REM2. (Figure 6)
 - iv. Combine DIV with REM1 to create child C1; combine DIV with REM2 to create child C2. (Figure 7)

Parent P1		RFindices					Parent P2		RFindices				
		0	1	2	3	4			0	1	2	3	4
Dfindices	0	0	0	0	0	1	DFindices	0	0	0	1	0	
	1	0	0	0	1	0		1	0	0	0	0	0
	2	0	1	0	0	0		2	0	0	1	0	0
	3	1	0	0	0	0		3	1	0	0	0	0
	4	1	0	0	0	0		4	0	0	0	1	0
	5	0	0	1	0	0		5	0	0	0	1	0
	6	1	0	0	0	0		6	0	0	0	0	1
	7	0	0	1	0	0		7	0	0	1	0	0
	8	0	0	0	1	0		8	0	0	1	0	0
	9	0	0	1	0	0		9	0	1	0	0	0
	10	0	0	0	0	1		10	0	1	0	0	0
	11	0	1	0	0	0		11	1	0	0	0	0
	12	0	1	0	0	0		12	0	1	0	0	0
	13	0	0	0	1	0		13	0	0	0	0	1
	14	0	0	0	0	1		14	0	0	0	0	1

Figure 4. Parent chromosomes before crossover

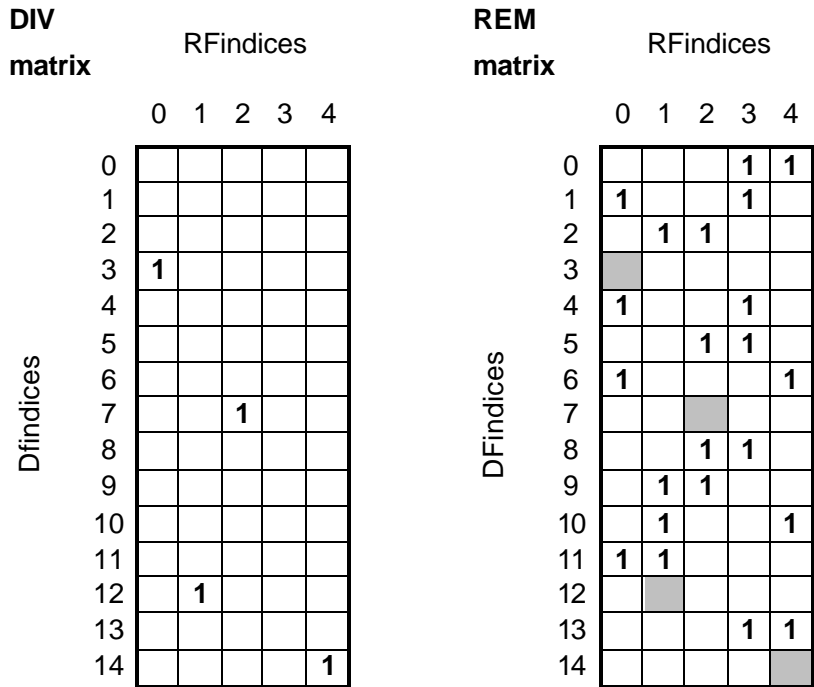


Figure 5. DIV and REM matrices

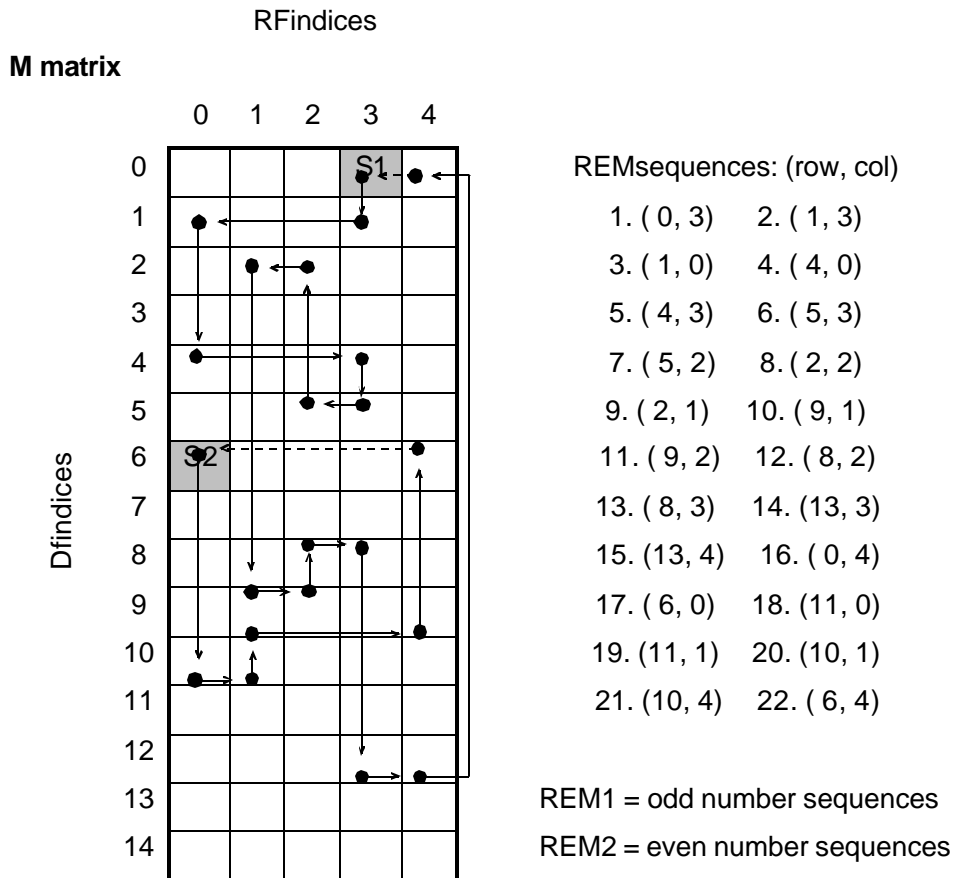


Figure 6. Partitioning of REM matrix

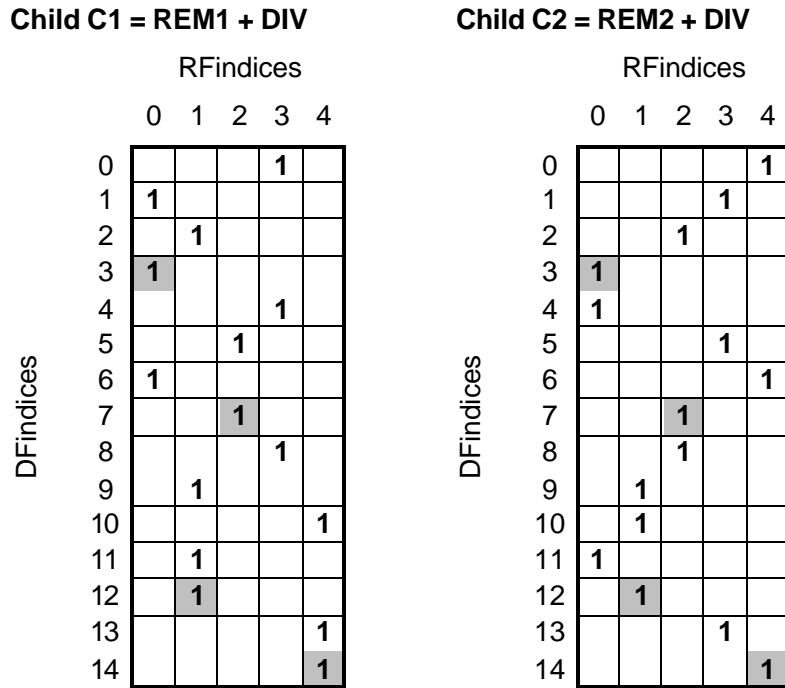


Figure 7. Resulting offspring after crossover

- c) Mutation: The steps below apply to the adjacency list representation:
- Input: parent chromosome P
- Output: unaltered parent chromosome P, with child chromosome C
- i. Create a copy of P, name it C.
 - ii. Randomly choose r indices subset of DIndices, and c indices subset of RIndices, where r and $c \geq 2$.
 - iii. Re-shuffle those entries in the intersection.

$$r = \{0, 9, 14, 11\}, \quad |r| = 4$$

$$c = \{3, 1, 4\}, \quad |c| = 3$$

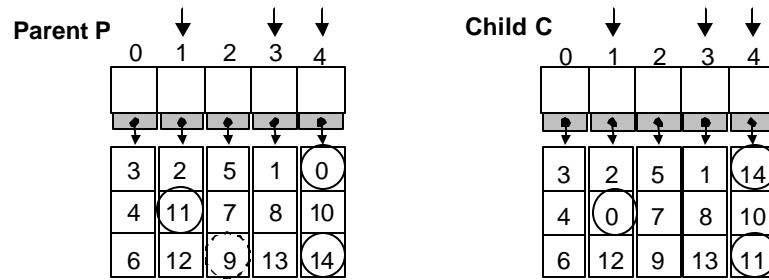


Figure 8. Mutation operation

V. Results and Discussion

Figure 9 compares the quality of solution obtained by GA and greedy algorithm over five problem instances of varying sizes. Exhaustive enumerative-search algorithm was used primarily to get the true minimum cost for each instance. The graph shows how greedy algorithm finds a solution quickly (not necessarily optimum), whereas GA spends additional time improving the solution it initially had, later moving past the solution obtained by greedy, and eventually landing on the global minimum cost solution. The empirical results show how greedy algorithm and GA differ in the way a solution is obtained. The latter, unlike the former, has the feature of improving the solution it initially obtained further until it leads to (hopefully) the true optimal solution.

Figures 10, 11, and 12 compare the performance of the two algorithms for larger problem instances. Though the global minimum cost was unknown for each instance, it could be observed that GA moves on to converge to a value past the one obtained by greedy algorithm.

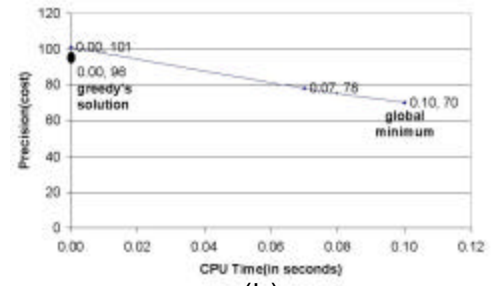
While, on the one hand, GA had a *good* chance of arriving at the minimum cost solution if only given enough time to run compared with greedy algorithm, GA's speed in improving the solution, on the other hand, slowly deteriorated as the size of the problem is increased. Two enhancements on GA itself were made, both cutting down significantly the converging time of GA.

The first enhancement made GA start with a population containing relatively *good* solutions to improve on. This seeding process was performed by a greedy-like algorithm designed specifically for the type of chromosome used, though any fast approximation algorithm might be experimented with. Figure 13 shows a remarkable improvement in the

running time of GA (compare this with figures 10, 11 and 12).

The second possibility improved the running time of GA by fine-tuning the parameters of GA itself; that is, running GA with a different set of population size, number of crossover and mutation per generation, etc (Deb et al., 1999). For recipient and donor files with 86 and 192 records, respectively, one set of parameters lowered the cost from 439 down to 420 after 7.55 minutes had elapsed, while another set was observed to improve the same cost down to 419 after only 58.25 seconds.

RFsizDFsiz		EXHAUSTIVE		GREEDY		GA	
e	e	cost	time	cost	time	cost (from- to)	time
3	9	106	0.00s	136	0.00s	106-106	0.00s
4	12	72	0.01s	102	0.00s	72-72	0.00s
5	15	60	0.41s	86	0.00s	87-60	0.41s
6	18	70	17.44s	96	0.00s	101-70	0.10s
7	21	104	874.38s	104	0.00s	170-104	0.06s



(b)

Figure 9. GA's convergence to a global optimum; (b) on 6RF x 18DF

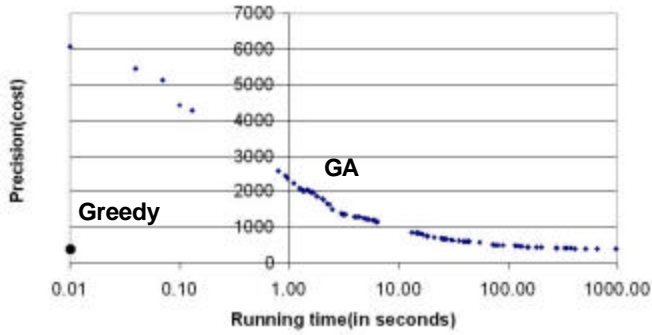


Figure 10. GA versus greedy algorithm on 100RF x 300DF

GA's performance on 100RF x 300DF

Time	Cost
372.50s	408
403.42s	403
507.57s	402
510.83s	400
652.30s	393
959.26s	392

GA's performance on 150RF x 450DF

Time	Cost
1099.88s	534
1329.64s	518
1418.06s	516
1561.98s	514
1718.17s	513
1814.57s	509
1915.32s	507
2782.16s	500
3350.92s	497
3569.07s	496
3633.64s	494
4879.06s	492

vs Greedy: 541 in 0.01s

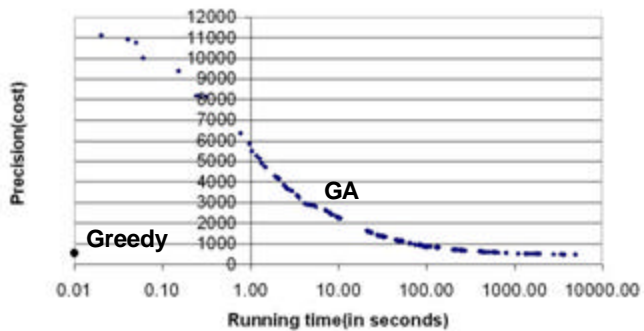
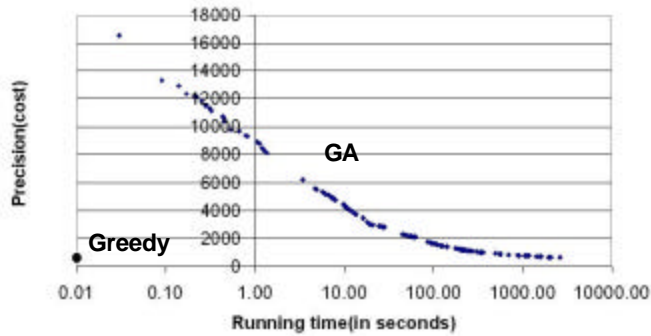


Figure 11. GA versus greedy algorithm on 150RF x 450DF



GA's performance on 200RF x 600DF

Time	Cost
2580.79s	668

vs Greedy: 680 in 0.02s

Figure 12. GA versus greedy algorithm on 200RF x 600DF

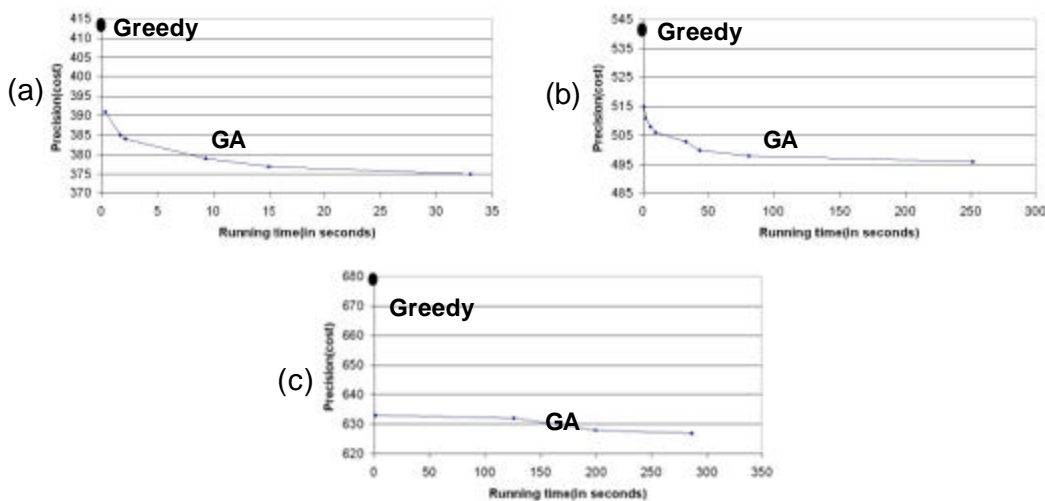


Figure 13. GA on (a) 100RF x 300DF, (b) 150RF x 450DF, and (c) 200RF x 600DF

VI. Summary and Conclusion

This study presented a Genetic Algorithm (GA) for Constrained Statistical Matching after the latter was transformed into two closely related combinatorial optimization problems of Operations Research: the Traveling Salesman Problem and the Balanced Linear Transportation Problem. The merging of the two models provided a hint on the type of chromosome (and accompanying genetic operators) that the genetic algorithm must have. The implementation of the resulting algorithm was made to run on different problem instances done mainly to verify GA's ability to converge to a (possibly) global minimum cost matching. Empirical results confirmed GA's claim of finding, under certain conditions, the best solution which cannot be arrived at by a pure greedy algorithm approach, by subjecting the population of candidate solutions to genetic transformations. This feature, however, was compensated by extra computing time making GA to run slower than greedy algorithm, made more apparent as the problem instances increased in size. Hence, allowing GA to

search initially from a set of relatively *good* solutions rather than from a population of randomly generated ones offered a good opportunity for the two algorithms to work together. This was accomplished by letting greedy algorithm create the initial population which GA iteratively fine tunes next using its set of genetic operators. This hybridization process manifested a 2-fold advantage than what a pure greedy or genetic algorithm could offer:

1. a cheaper cost matching than what could be obtained by a pure greedy algorithm;
and,
2. an execution time relatively faster than what a pure-blind GA with initially random poor-quality solutions could have

Bibliography

- Albacea, EA and AN Gironella. 2003. Building panel data for monitoring poverty in the Philippines. Proceedings of the ADB TA 3656 PHI: Improving Poverty Monitoring Surveys Seminar.
- Alegre, J, J Arcarons, S Calonge and A Manresa. 2000. Statistical matching between different data sets: an application to the Spanish household survey(EPF09) and the income tax file(IRPF90). Proceedings of the 2000 Workshop on Fighting Poverty and Inequality Through Tax Benefit Reform: Empirical Approaches.Barcelona, Spain.
- Barry, JT. 1988. An investigation of statistical matching. Journal of Applied Statistics, Vol. 15, pp. 275-283.
- Cox, LH and Rf Boruch. 1985. Emerging policy issues in record linkage and privacy. Journal of Official Statistics, Vol. 4, No. 1, pp. 3 -16.
- Deb, K and S Agrawal. 1999. Understanding interactions among genetic algorithm parameters. In Banzhaf, W. and C. Reeves (eds.). Foundations of Genetic Algorithms. San Francisco, California: Morgan Kaufmann Publishers, Inc.
- De Jong, Ka and Wm Spears. 1989. Using genetic algorithms to solve NP-complete problems. Proceedings of the Third International Conference on Genetic Algorithms. <http://www.aic.nrl.navy.mil/~spears/papers/icga89.ps.gz>.
- Dean, T and L Greenwald. 1992. A formal description of the transportation problem. <ftp://ftp.cs.brown.edu/pub/techreports/92/cs92-14.ps.Z>.
- Falkenauer, E. 1997. The lavish ordering genetic algorithm. Proceedings of the Second International Conference on Metaheuristics. <http://www.ulb.ac.be/sma/publications/mic97.ps>.
- Flores, GA and EA Albacea. 2004. Constrained statistical matching via genetic algorithm. [GS Graduate Thesis] College, Laguna, Philippines: University of the Philippines.
- Garey, MR and DS Johnson. 1979. Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco.
- Goldberg, DE. 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley.
- Michalewicz, Z. 1996. Genetic algorithms + data structures = evolution programs. 3rd, Revised, & Extended ed. Springer, New York.
- Michalewicz, Z and GA Vignaux. 1991. A genetic algorithm for the linear transportation problem. IEEE Transactions on Systems, Man, and Cybernetics, Vol 21, No. 2, March/April 1991.
- Paass, G. 1985. Statistical record linkage methodology. Proceedings of the 100th Session of the International Statistical Institute, Amsterdam, International Statistical Institute.
- Soong, R And M De Montigny. 2001. The anatomy of data fusion. Proceedings of the 2001 Worldwide Readership Research Symposium. Venice.
- Talbi, EG. 1999. A taxonomy of hybrid metaheuristics. Journal of Combinatorial Optimizations. <http://www.lifl.fr/~talbi/hybrid98.ps.gz>.
- Tena, JKS. 2002. Constrained statistical matching of 1997 FIES and 1998 APIS of region 4 data. [INSTAT Undergraduate Special Problem] College, Laguna, Philippines: University of the Philippines.

Whitley, D and N Yoo. 1995. Modeling simple genetic algorithms for permutation problems.
<http://www.cs.colostate.edu/~genitor/1995/permutations.ps.gz>.

Yoshizoe, Y and M Araki. 1999. Statistical matching of household survey files. Proceedings of the
52nd Session of the International Statistical Institute, Helsinki, Finland.

http://www.epcc.ed.ac.uk/overview/publications/training_material/tech_watch/97_tw/techwatch-ga/ga-

[tw-2.html](http://www.epcc.ed.ac.uk/overview/publications/training_material/tech_watch/97_tw/techwatch-ga/ga-tw-2.html). Order-based problems and the TSP problem.

<http://www.stat.washington.edu/tantrum/project.ps.gz>_ Unpublished document.

<http://www.zonalatina.com>. Zona Latina: Latin American Media & Marketing Articles.