On the Stable Marriage of Maximum Weight Royal Couples

Anan Marie, Avigdor Gal
Technion – Israel Institute of Technology
Haifa
32000, Israel
{sananm@cs,avigal@ie}.technion.ac.il

Abstract

In this paper we provide a comparison, both analytic and empirical, of two algorithms that were used in the literature for ensuring a 1 : 1 cardinality constraint in schema matching. We compare an application of a solution to the maximum weighted bipartite graph to schema matching to that of solving a stable marriage problem. Using real-world testbed we show that in practice, both algorithms yield similar results. We then analyze the roots of this similarity, by offering a variation of the stable marriage algorithm for schema matching (*royal couples*) and show the relationships between the royal couples stable marriage problem and the maximum weights bipartite graph problem. Finally, we propose a new heuristic for schema matching, based on the royal couples observation and show its performance with respect to the two algorithms.

Introduction

Schema matching is the task of matching between concepts describing the meaning of data in various heterogeneous, distributed data sources. It is recognized to be one of the basic operations required by the process of data and schema integration (Melnik 2004), and thus has a great impact on its outcome.

A heavily studied area of schema matching involves the enforcement of 1 : 1 mapping cardinality. That is, an element (*e.g.*, a relational database attribute) can be mapped to at most one element of another schema. Two main methods were used to enforce 1 : 1 cardinality constraints. The first represents the schema matching problem as a bipartite graph and then solves a ***M**aximum **W**eighted **B**ipartite **G**raph* (***MWBG***) problem. The second represents the schema matching problem as a set of ranking lists, in which each schema element ranks all elements of the other schema in a decreasing order of similarity. Then, a best mapping is achieved by solving a ***S**table **M**arriage* (***SM***) problem. Both algorithms fall into the category of constraint enforcers in (Lee *et al.* 2007) and both enforce a cardinality constraint of 1:1.

In this paper we provide a comparison, both analytic and empirical, of the two algorithms. Using real-world testbed we show that in practice, both algorithms yield similar results. We then analyze the roots of this similarity, by offering a variation of the stable marriage algorithm for schema matching (dubbed *royal couples*) and show the relationships between the royal couples stable marriage problem and the maximum weights bipartite graph problem. Finally, we propose a new heuristic for schema matching, based on the royal couples observation and show its performance with respect to the algorithms.

The specific contribution of this work are as follows:

- We show the relationships between ***MWBG*** and ***SM*** in the specific context of schema matching.
- We propose a new schema matching heuristic, ***D**ominant **M**atching* (***DM***), that is based on the observations regarding the royal couples stable marriage problem.
- We present a thorough empirical analysis of the performance of ***MWBG***, ***SM***, and ***DM***. Our comparative analysis shows that ***DM*** can increase the Precision over both ***MWBG*** and ***SM*** by about 30%, accompanied by a reduction in Recall of about 6%, making it a reasonable candidate for today's schema matching tasks.

The rest of the paper is organized as follows. We first present a model of schema matching, to be used in this paper, and provide a brief introduction to ***MWBG*** and ***SM***. An analysis of the two schema matchers is given next, both emprically and analytically. Then, we introduce ***DM***, followed by a comparative empirical analysis.

Model

The model we present here is based on (Domshlak, Gal, & Roitman 2007). Let *schema* $S = \{A_1, A_2, ..., A_n\}$ be a finite set of some *attributes*. We set no particular limitations on the notion of schema attributes; attributes can be both simple and compound, compound attributes should not necessarily be disjoint, *etc.* For

any schemata pair S and S', let $\mathcal{S} = S \times S'$ be the set of all possible *attribute mappings* between S and S'. Let $M(S, S')$ be an $n \times n'$ *similarity matrix* over $\mathcal{S}$, where $M_{i,j}$ represents a degree of similarity between the i-th attribute of S and the j-th attribute of S'. The majority of works in the schema matching literature define $M_{i,j}$ to be a real number in $(0, 1)$. $M(S, S')$ is a *binary similarity matrix* if for all $1 \leq i \leq n$ and $1 \leq j \leq n'$, $M_{i,j} \in \{0, 1\}$.

Let the power-set $\Sigma = 2^{\mathcal{S}}$ be the set of all possible *schema mappings* between this pair of schemata and let $\Gamma : \Sigma \rightarrow \{0, 1\}$ be a boolean function that captures the application-specific constraints on schema mappings, *e.g.*, cardinality constraints and inter-attribute mapping constraints. We say Γ is a null constraint function if for all $\sigma \in \Sigma$, $\Gamma(\sigma) = 1$. Given a constraint specification Γ, the set of all *valid* schema mappings in Σ is given by $\Sigma_\Gamma = \{\sigma \in \Sigma \mid \Gamma(\sigma) = 1\}$.

Formally, the input to the process of schema matching is given by two schemata S and S' and a constraint boolean function $\Gamma : \Sigma \rightarrow \{0, 1\}$.[1] The output of the schema matching process is a schema mapping $\sigma \in \Sigma_\Gamma$.

Schema matchers are instantiations of the schema matching process. Various schema matchers differ mainly in the measures of similarity they employ, yielding different similarity matrices. These measures can be arbitrarily complex, and may use various techniques for name matching, domain matching, structure matching (such as XML hierarchical representation), *etc.*

Let $G = (S, S', E)$ be an undirected bipartite graph, with a node set $V = S \bigcup S'$ representing attributes, where S and S' denote the sides of the graph, and an edge set E. Weights $w : E \rightarrow R^+$ are assigned to edges, representing the degree of similarity between the attributes. A mapping σ in G is a subset of pair-wise disjoint edges of E. An efficient algorithm $\mathcal{O}(n^3)$ for identifying the best mapping, given all pair-wise similarities, is given as a variation of the ***MWBG*** matching algorithm (Galil 1986).[2]

The ***SM*** problem, introduced by Gale and Shapley (Gale & Shapley 1962), was used in (Melnik, Garcia-Molina, & Rahm 2002) to generate schema mappings. In the ***SM*** problem, each of n men and n women lists the members of the opposite sex in strict order of preference. The goal is to find a stable marriage E, a complete matching of men and women with the property that there is no unmatched pair (m, w) such that m prefers w to his partner in E, and w prefers m to her partner in M. Such a pair is called a *blocking pair*.

Gale and Shapley have shown that a stable marriage always exists and can be found in $\mathcal{O}(n^2)$. Their algorithm works through a series of proposals where every time a *free* man proposes to the most preferred woman on his list that did not previously reject him. If a woman is proposed to by a man whom she prefers over her current fiancé or she has not been engaged yet, she *frees* her current fiancé (if exists) and engages with the proposed man, otherwise she *rejects* him. Thus, during the execution of the algorithm, men propose to women and some men and some women become engaged. Women can break engagements if they receive a better offer.

It is worth noting that under the model presented in this section, applying the ***SM*** algorithm to the process of schema matching involves an additional effort that is needed in transforming a matrix of weights to lists of preferences. This effort involves sorting each row and column in the matrix and therefore the complexity of constructing these lists is $\mathcal{O}(n^2 \log n)$, dominating the total complexity of the algorithm.

Royal Couple Matching

We now present a matching algorithm that is based on an interesting observation pertaining to the ***SM*** application to solving schema matching problems, as follows: Recall that the preference lists in the ***SM*** algorithm are generated from a similarity matrix. Therefore, at each iteration of the algorithm there is at least one couple whose elements prefer each other. This is the couple whose entry in the matrix has a maximum value. We term such a pair a *royal couple*.

Algorithm 1 `Royal Couples`

1: `Input:` an $n \times n$ similarity matrix M
2: **repeat**
3: Find the maximum value in M
4: Insert the maximum value $M(i, j)$ to the output mapping
5: Delete the i-th row and the j-th column from M
6: **until** M is not empty

We use this observation in constructing the royal couple algorithm (Algorithm 1). At each iteration, the algorithm selects a pair of attributes (i, j) with the maximum value $M_{i,j}$ in the similarity matrix M, adds the pair (i, j) to the output mapping E and deletes the i-th row and the j-th column of M, to satisfy the 1:1 cardinality constraint. The time complexity of Algorithm 1 is $\mathcal{O}(n^2 \log n)$, which is again the cost of sorting all rows and all columns.

Theorem 1 *The matching computed by the* Royal Couples *algorithm is a stable marriage.*

Proof. Let E be the mapping computed by the *Royal Couples* matching algorithm. We need to show that there is no blocking pair (m, w). Suppose, by way of contradiction, that there is a blocking pair $(m, w) \notin E$. Assume $(m, w') \in E$ and $(m', w) \in E$. Assume also,

[1] For ease of exposition, we constraint our presentation to a matching process of two schemata. Some approaches to schema matching, such as holistic schema matching (He & Chang 2005; Su, Wang, & Lochovsky 2006) operate on any number of schemata, and although their presentation in the form of a high-dimensional matrix is possible, we refrain from presenting it herein.

[2] More sophisticated (parallel) algorithms solve this problem in $\mathcal{O}(n^{2.5})$.

without loss of generality, that (m, w') was selected before (m', w). Consider the iteration i in which the algorithm selects (m, w'). There are two possible cases:

- At iteration i, (m, w) was still in the matrix: Since the algorithm selected (m, w') at iteration i, (m, w') is the current maximum value. In particular, $M_{m,w'} > M_{m,w}$, *i.e.*, m prefers w' to w and therefore (m, w) is not a blocking pair, a contradiction.
- At iteration i, (m, w) was already removed from the matrix: this case is impossible as we shall now show. At iteration i, (m, w') is still in the matrix and also (m', w), since (m, w') is removed before (m', w). However, since (m, w) was deleted before this iteration, it means that either row m or column w were deleted (line 5 of Algorithm 1) $\Rightarrow$ (m, w') or (m', w) is not in the matrix anymore, a contradiction.

According to Theorem 1, Algorithm 1 can serve in solving the stable marriage problem. It is worth noting that Algorithm 1 does not improve on Gale and Shapely algorithm in any way. In particular, its worst case time complexity is the same. Still, its presentation serves in demonstrating the inter-relationships between ***SM*** and ***MWBG*** and serves as a basis for the new heuristic presented in this paper.

Empirical Comparison of *MWBG* and *SM*

We now present an empirical evaluation of ***MWBG*** and ***SM***. We report in details on our experimental setup, the data that was used, and the evaluation methodology. We then present the experiment results and provide an empirical analysis of these results.

Experiment setup

For generating the similarity matrices, we have used the Combined OntoBuilder heuristic (Gal *et al.* 2005), combining four matchers as detailed below:

Term: A term is a combination of a label and a name. Term matching compares labels and names to identify syntactically similar terms. To achieve better performance, terms are preprocessed using several techniques originating in IR research. Term matching is based on either complete word or string comparison.

Value: Value matching utilizes domain constraints (*e.g.*, drop lists, check boxes, and radio buttons) to compute similarity measure among terms. The availability of constrained value-sets becomes valuable when comparing two terms that do not exactly match through their labels.

Composition: A composite term is composed of other terms (either atomic or composite). Composition can be translated into a hierarchy. This schema matcher assigns similarity to terms, based on the similarity of their neighbors.

Precedence: The precedence relationship is unique to OntoBuilder and therefore worth of a lengthier discussion. In any interactive process, the order in which data are provided may be important. In particular, data given at an earlier stage may restrict the availability of options for a later entry. For example, a car rental site may determine which car groups are available for a given session, using the information given regarding the pick-up location and time. Therefore, once those entries are filled in, the information is sent back to the server and the next form is brought up. Such precedence relationships can usually be identified by the activation of a script, such as (but not limited to) the one associated with a SUBMIT button. Precedence can be translated into a precedence graph. The matching algorithm is based on a technique we dub *graph pivoting*, as follows. When matching two terms, we consider each of them to be a pivot within its own ontology, thus partitioning the graph into semantically related subgraphs. The semantics of pivoting is taken from the ontological analysis, and in the case of precedence the graph it partitioned into a subgraph of all preceding terms and all succeeding terms. By comparing preceding subgraphs and succeeding subgraphs, we determine the confidence strength of the pivot terms.

In addition to Combined, we have also experimented with the matrices of Composition and Precedence individually. We have applied ***MWBG*** and ***SM*** to each matrix (see dataset description below). All algorithms were implemented using Java 2 JDK version 1.5.0_09 environment, using an API to access OntoBuilder's matchers and get the output matrices. The experiments were run on a laptop with Intel Centrino Pentium m, 1.50GHz CPU, 760MB of RAM Windows XP Home edition OS.

Data

For our experiments, we have selected 230 Web forms from different domains, such as dating and matchmaking, job hunting, Web mail, hotel reservation, news, and cosmetics. We extracted each Web form ontology using OntoBuilder. We have matched the Web forms in pairs (115 pairs), where pairs were taken from the same domain, and generated the exact mapping (a mapping generated by a human observer) for each pair.[3] The ontologies vary in size and the proportion of number of attribute pairs in the exact mapping relative to the target ontology. Another dimension is the size difference between matched ontologies.

Evaluation methodology

For evaluation we use two main metrics, namely Precision and Recall. Precision is computed as the ratio of correct attribute mappings, with respect to some exact

[3]All ontologies and exact mappings are available for download from the OntoBuilder Web site, http://ie.technion.ac.il/OntoBuilder).

mapping, out of the total number of attribute mappings suggested by a heuristic. Recall is computed as the ratio of correct attribute mappings, out of the total number of attribute mappings in the exact mapping. Both Recall and Precision are measured on a $[0, 1]$ scale. An optimal schema matching results in both Precision and Recall equal to 1. Lower precision means more false positives, while lower recall suggests more false negatives. To extend Precision and Recall to the case of non $1:1$ mappings, we have adopted a correctness criteria according to which any attribute pair that belongs to the exact mapping is considered to be correct, even if the complex mapping is not fully captured. This method aims at compensating the matchers for the $1:1$ cardinality enforcement. We have also used F-measure, a harmonic average of Precision and Recall.

Results

	%	Average Precision Improv.	Average Recall Improv.	Average F-measure Improv.
MWBG>SM	12.50	40.71	39.62	40.26
SM>MWBG	14.29	22.09	22.09	22.09
SM∼MWBG	46.43			
SM=MWBG	26.79			

Table 1: Comparison of ***MWBG*** and ***SM***: Combined

Table 1 summarizes the results of our empirical comparison with the Combined algorithm. ***MWBG>SM*** means that ***MWBG*** outperformed ***SM*** in terms of both Precision and Recall and vice versa for ***SM>MWBG***. ***SM∼MWBG*** means that their Precision and Recall values were exactly the same and ***SM=MWBG*** means that not only they share the same Precision and Recall value, but also their best mappings were identical. The second column shows the percentage of the pairs for which the relationship holds. For those cases where the performance differ, we also record the improvement in each of the measures the superior algorithm had over the inferior one.

The main observations from these experiments are that in about 73% of the pairs, ***MWBG*** perform exactly the same as ***SM***, and in 37% out of these experiments, they also propose the same mapping. For the remaining 32%, no algorithm is shown to be dominant. Nevertheless, for those cases where ***MWBG*** outperformed ***SM*** it achieves on average double as much improvement than when ***SM*** outperformed ***MWBG***.

Tables 2 and 3 show results for the Composition and Precedence algorithms, respectively. Again, for the majority of the pairs, ***MWBG*** and ***SM*** perform the same. Here, however, the average improvement of one algorithm over the other is about the same.

MWBG and *SM* interrelationship

Our empirical results served as a motivation to explore a sufficient condition that ensures equal performance

	%	Average Precision Improv.	Average Recall Improv.	Average F-measure Improv.
MWBG>SM	14.04	18.74	18.74	18.74
SM>MWBG	13.16	18.74	18.74	18.74
SM∼MWBG	48.25			
SM=MWBG	24.56			

Table 2: Comparison of ***MWBG*** and ***SM***: Composition

	%	Average Precision Improv.	Average Recall Improv.	Average F-measure Improv.
MWBG>SM	26.79	27.00	27.00	27.00
SM>MWBG	14.29	29.57	29.57	29.57
SM∼MWBG	41.96			
SM=MWBG	16.96			

Table 3: Comparison of ***MWBG*** and ***SM***: Precedence

of the two algorithms in schema matching applications. In this section we aim at analyzing the interrelationship between the ***MWBG*** algorithm and the ***SM*** algorithm. First, to assist us in our analysis, we shall introduce the notion of a *dominant pair*.

Definition 2 *Let M be a similarity matrix over a pair of schemata S and S'. A pair (i, j) is* dominant *if 1) $M_{i,j} \geq M_{i,j'}$ for $1 \leq j' \leq n$ and 2) $M_{i,j} \geq M_{i',j}$ for $1 \leq i' \leq n$.*

In the following lemmas, we assume, for simplicity sake, that the similarity matrix does not contain two identical values. Also, we assume that S and S' are of identical arity, *i.e.*, there are n attributes in each schema. These assumptions do not affect the correctness of our lemmas, they just make it simple to prove them. We first show the importance of dominant pairs to the output of ***SM***.

Lemma 3 *Let S and S' be two sets of attributes, and M be a similarity matrix over S and S'. Each* dominant *pair in M must be included in the stable marriage computed by Gale and Shapley algorithm.*

Proof. Let (i, j) be a *dominant* pair, and let σ_s be a stable marriage computed by Gale and Shapley algorithm. Suppose $(i, j) \notin \sigma_s$. Assume also, $(i, j') \in \sigma_s$ and $(i', j) \in \sigma_s$. Since (i, j) is a dominant pair, j appears before j' in i's preference list, and i appears before i' in j's preference list. In other words, i prefers j to its partner in σ_s (j') and j prefers i to its partner in σ_s (i'). Thus, the pair (i, j) is a blocking pair according to σ_s. A contradiction.

Lemma 4 provides an upper bound on the weight of a mapping, generated by ***MWBG***.

Lemma 4 *Let S and S' be two sets of attributes, and M be a similarity matrix over S and S'. The weight of the mapping (the sum of the weights of the pairs in the mapping) computed*

by the **MWBG** *is at most* $\min(\sum_{i=1}^{n} r_i, \sum_{i=1}^{n} c_i)$, *where* $r_i = \max(M_{ij} : j = 1, \ldots, n)$ *and* $c_j = \max(M_{ij} : i = 1, \ldots, n)$

Proof. Suppose, without loss of generality, that $\sum_{i=1}^{n} r_i \leq \sum_{i=1}^{n} c_i$. Let σ denotes the maximum weighted matching, and W_σ denotes its weight. $W_\sigma = \sum_{i=1}^{n} M_{i,\sigma_i}$. Since $\forall i(M_{i,\sigma_i} \leq r_i)$, then $W_\sigma \leq \sum_{i=1}^{n} r_i$

Theorem 5 *Let S and S' be two sets of attributes, and M be a similarity matrix over S and S'. If there are n* dominant *pairs and no two* dominant *pairs reside in the same row or the same column, then the* **MWBG** *algorithm and the* **SM** *algorithm are equivalent,* i.e.*, result in the same matching.*

Proof. Assume, without loss of generality, that $\sum_{i=1}^{n} r_i \leq \sum_{i=1}^{n} c_i$. Let σ be the set of all the dominant pairs. σ is a complete $1:1$ mapping. According to Lemma 3, σ is a stable matching. Let d_i denote the dominant pair in the i-th row and $W(d_i)$ its value in the matrix. $W(d_i) = r_i \Rightarrow \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} r_i$. Since $W(\sigma) = \sum_{i=1}^{n} d_i$ then $W(\sigma) = \sum_{i=1}^{n} r_i$. And according to Lemma 4 σ is a maximum matching.

Theorem 5 provides a sufficient condition for the equivalence of the ***MWBG*** algorithm and the ***SM*** algorithm, in terms of dominant pairs. We conclude with the following observation. The ***MWBG*** algorithm maximizes the overall benefit of the output mapping while the ***SM*** algorithm maximizes the local benefits of one of the genders (may it be men or women). In schema matching problem instances, the stable marriage instance is symmetric. Thus, when maximizing the local benefits of the men, it also maximizes the local benefits of the women. Thus, we obtain a powerful stable marriage that is both male-optimal and female-optimal.

Dominants

To illustrate a possible benefit of the analysis above, we now propose a new heuristic for schema matching. Recall that in Algorithm 1 we constraint the best mapping to be of cardinality $1:1$, following ***SM***. The basic assumption behind the *Royal Couples* algorithm is that the pair (i, j) with the maximum confidence measure in the matrix has the highest chance of being in the exact mapping. Then, to maintain the $1:1$ cardinality constraint, we delete the row i and the column j from the matrix. These deletions may affect the next maximum value in the matrix. In other words, deletion of a row or a column that contains relatively high confidence values, may generates new maximum values that were relatively low. Consequently our basic assumption becomes weaker.

Theorem 5 provides a sufficient condition to the identical behavior of ***MWBG*** and ***SM***. What happens when this condition is not satisfied? One such scenario may occur whenever there are less than n dominant values. In such a scenario, both ***MWBG*** and ***SM*** are bound to err and add false positive attribute mappings. Those false positive mappings may be different between the two algorithms. Another scenario involves non-$1:1$ mappings, with more than n dominant values. In this case, there are several dominant values in a single row or column (note that those dominant values must have identical values). ***MWBG*** and ***SM*** will fail to identify some of the dominant values, and will be forced to make some arbitrary choices, and again the two algorithms may make different choices. To be able to identify such scenarios, we make a do with the $1:1$ cardinality constraint, and offer a heuristic that finds the pairs with the best chance of being in the exact mapping, in a way that selecting one pair would not affect others.

Algorithm 2 Dominants Matching

```
1: Input: an n × n similarity matrix M
2: for all row i do
3:     calculate the maximum value in i, insert this
       value to array R.
4: end for
5: for all column j do
6:     calculate the maximum value in j, insert this
       value to array C
7: end for
8: for all column j do
9:     If M(i, j) = R(i) = C(j), insert (i, j) to the out-
       put mapping.
10: end for
```

The ***Dominants Matching*** (***DM***) heuristic is given in Algorithm 2. The main assumption guiding this heuristic is that the dominant pairs are the most probable to be in the exact mapping since the two attributes involve in a dominant pair prefer each other most. Note that with this heuristic not all the target attributes are mapped and that an attribute in one schema may be mapped to more than one attribute in another schema, whenever attribute pairs share the same similarity level (effectively, introducing no cardinality constraint). The time complexity of Algorithm 2 is $O(n^2)$.

Figure 1 compares between the average performance of ***MWBG***, ***SM***, and ***DM*** for the Combined algorithm. For each of the three heuristics we present their average Precision, Recall, and F-measure. We first note that an average Precision of 45% is common on large and challenging real-world schemata. We refer the interested reader to the new benchmark of OAEI[4] for more information.

DM is the clear winner in terms of Precision (increase of about 29%) while sacrificing in terms of Recall (decrease of about 6%). For a predefined weighing of Precision and Recall, using F-measure, ***DM*** is again the best heuristic improving over both ***MWBG*** and ***SM*** by about 7%). Similar results were observed for the Precedence and Graph algorithms, and we refrain from showing the results here.

[4] http://oaei.ontologymatching.org/2006/results/directory/

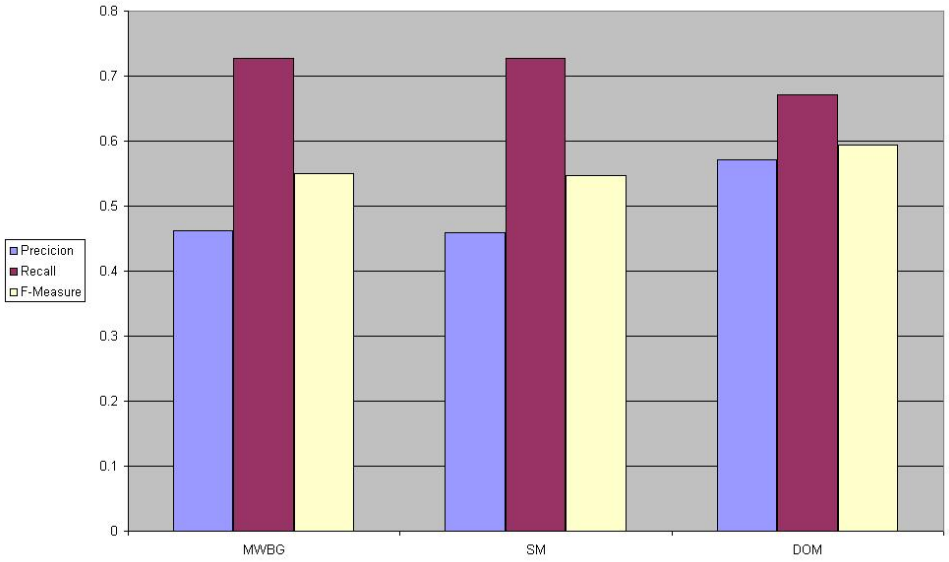

Figure 1: Comparative Performance Analysis for the Combined Algorithm

Conclusions

In this paper we focused on the inter-relationships between two well known algorithms for constrained, $1:1$ schema matching. We provided an empirical analysis, showing their comparable performance and then provided a theoretical analysis, showing a sufficient condition for both algorithms to yield identical outcome.

This analysis can serve in the design of new heuristics, as was already shown in this paper. Other directions for the design of new heuristics involve the use of both algorithms together to improve one measure (*e.g.*, Precision) at the cost of another (*e.g.*, Recall).

References

Domshlak, C.; Gal, A.; and Roitman, H. 2007. Rank aggregation for automatic schema matching. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 19(4):538–553.

Gal, A.; Modica, G.; Jamil, H.; and Eyal, A. 2005. Automatic ontology matching using application semantics. *AI Magazine* 26(1).

Gale, D., and Shapley, L. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15.

Galil, Z. 1986. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys* 18(1):23–38.

He, B., and Chang, K.-C. 2005. Making holistic schema matching robust: an ensemble approach. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, 429–438.

Lee, Y.; Sayyadian, M.; Doan, A.; and Rosenthal, A. 2007. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal* 16(1):97–122.

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 IEEE CS International Conference on Data Engineering*, 117–140.

Melnik, S. 2004. *Generic Model Management: Concepts and Algorithms*. Springer-Verlag.

Su, W.; Wang, J.; and Lochovsky, F. 2006. Aholistic schema matching for web query interfaces. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, 77–94.