

Team Integration

Do Son Thanh

Abstract—We leverage theoretical advances and the multi-user nature of *argumentation*. The overall contributions of our work are as follows. We model the schema matching network and the reconciliation process, where we relate the experts’ assertions and the constraints of the matching network to an *argumentation framework*. Our representation not only captures the experts’ belief and their explanations, but also enables to reason about these captured inputs. On top of this representation, we develop support techniques for experts to detect conflicts in a set of their assertions. Then we guide the conflict resolution by offering two primitives: *conflict-structure interpretation* and *what-if analysis*. While the former presents meaningful interpretations for the conflicts and various heuristic metrics, the latter can greatly help the experts to understand the consequences of their own decisions as well as those of others. Last but not least, we implement an argumentation-based negotiation support tool for schema matching (ArgSM), which realizes our methods to help the experts in the collaborative task.

I. INTRODUCTION

The task of reconciling a schema matching network was performed by a single expert. As the size of networks in data integration grows, the complex reconciliation tasks should be performed by not only one but several experts, to avoid the overload on a single expert and also to assign each expert the parts of the problem about which he is more familiar. Moreover, typical information systems need to involve a wide range of expertise knowledge, since schemas are often designed by different persons and with different domain purposes. As a result, there is a need for a mechanism that allows not a single expert but an expert team work collaboratively to reconcile the output of automatic matchers.

In this chapter, we develop such a multi-user mechanism to enable collaborative reconciliation process. It is challenging to achieve this goal since we have to face the three following issues. Note that hereby two terms—*experts* and *users*—are used interchangeably to represent the participants in this process.

- 1) *How to encode user inputs?* The inputs of users should be encoded to not only capture fully information given by users but also support reasoning on the information. In other words, from user inputs, we can derive consequences and compute their explanations.
- 2) *How to detect conflicting inputs?* As users might have different opinions about the correctness of correspondences, their inputs inevitably involve conflicts. Detecting conflicts is an important step to eliminate inconsistency.
- 3) *How to guide conflict resolution?* To facilitate conflict resolution, we need to define a mechanism that supports users to exchange knowledge, allow debugging, and contain explanations for the given decisions. Moreover, we provide heuristic metrics to rank possible decisions

TABLE I
TERMINOLOGIES

| Term | Description |
|---------------------------|--|
| Hypergraph | A hypergraph $G = (V, E)$ where V is a set of nodes and $E \subset S(V)$ |
| Correspondence hypergraph | A hypergraph in which vertices are correspondences and hyperedges are violations |
| Violation hypergraph | A hypergraph in which vertices are violations and hyperedges are correspondences |

as well as support “what-if” analysis, which involves computing the foreseeable consequences of the decisions.

To address these issues, we leverage theoretical advances and the multi-user nature of *argumentation*. The overall contributions of our work are as follows. We model the schema matching network and the reconciliation process [12], [6], [17], [5], [18], [11], where we relate the experts’ assertions and the constraints of the matching network to an *argumentation framework*. Our representation not only captures the experts’ belief and their explanations, but also enables to reason about these captured inputs. On top of this representation, we develop support techniques for experts to detect conflicts in a set of their assertions. Then we guide the conflict resolution by offering two primitives: *conflict-structure interpretation* and *what-if analysis*. While the former presents meaningful interpretations for the conflicts and various heuristic metrics, the latter can greatly help the experts to understand the consequences of their own decisions as well as those of others. Last but not least, we implement an argumentation-based negotiation support tool for schema matching (ArgSM), which realizes our methods to help the experts in the collaborative task.

II. PRELIMINARIES

There are many protocols for negotiation process [23], [22]. In these works, the authors considered general settings about resources, users’ preferences, etc.

A. Assumptions

For the sake of simplicity, we make the following assumptions for this problems:

1) Agent assumptions:

- The users are selfless: they have no other goal beside the common goal.
- The users are truthful: they do not tell lies or intentionally harm the system.

2) Environment assumptions:

- Multi-channel: an agent can communicates with many her/his related agents. Therefore, no need to identify which message is intended for which agents.

- Unlimited bandwidth: the agent can communicate with other agents without worrying sharing the communication bandwidth with others
- Reliable: there is no lost or tempered message during communication.

III. PROBLEM DEFINITION

Team integration problem $P(k, W, G)$ is the problem of harnessing the work of k workers on the workload W efficiently to reach the common goal G . In the schema matching domain, given a set of correspondences C and the constraints Γ , we can detect a set of violations $V = \{v_1, v_2, \dots, v_n \mid v_i = \{c_j, \dots, c_k\} \subset C\}$ that need to be resolved by k workers. Let $C^* = fb(C)$ be the set of correspondences after resolution, that C^* satisfies Γ and maximizes an objective function is the goal of the team integration process.

Formally, the team integration problem in schema matching is defined as follows:

SCENARIO. Team integration in schema matching Given a set of constraints Γ , a set of correspondence C and a set of derived violation $V = Vio_{\Gamma}(C)$ of C with respect to Γ , the team integration problem $P(k, W, G)$ of k workers can be modeled as follows:

- The workload W of this problem is V .
- The goal G of this problem is that the feedbacked correspondences C^* satisfies Γ and the objective value $obj(C^*)$ is maximized, where $obj(\cdot)$ is a pre-defined objective function.

In our context, we assume that the worker ability are equal and the role of each worker are the same. In order to solve the above problem, we need to solve two inherent sub-problems: design a fair task assignment and a conflict resolution protocol.

QUESTION 1. Task assignment Given a set of violations V and k workers, a task assignment is k subsets $T = \{V_1, V_2, \dots, V_k\}$ divided from V , where $V_i \subset V$, such that $V_i \cap V_j = \emptyset$ and $\bigcup_i V_i = V$.

QUESTION 2. Conflict resolution Design a communication paradigm between users such that the conflicts in the feedback process are resolved.

IV. TASK ASSIGNMENT

In this section, we formulate the first question in the team integration problem by fair task assignment.

Definition 1. Fair task assignment A task assignment $T = \{V_1, V_2, \dots, V_k\}$ is fair if it satisfies 2 following conditions:

- **Workload Balance:** $\frac{1}{b} \times \frac{\sum_{i=1}^k W_i}{k} \leq W_i \leq b \times \frac{\sum_{i=1}^k W_i}{k}$, where $W_i = \sum_{v \in V_i} |v|$ is the workload of V_i and $b \geq 1$ is a load-imbalance tolerance factor.
- **Communication Minimizing:** The amount of communication between workers is minimum. The amount of communication is measured as the total of overlapping correspondences: $msg(T) = \sum |C_i \cap C_j|$ where C_i is the correspondences of V_i .

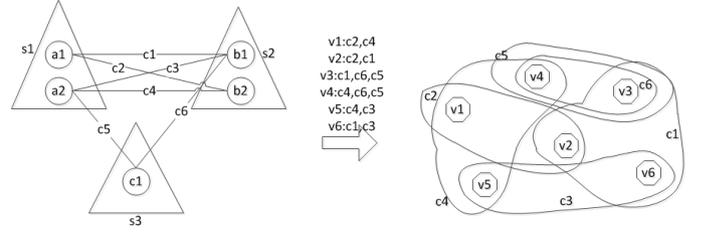


Fig. 1. A violation hypergraph is created from a network of schema

In order to solve this problem, we reformulate it into a k -way hypergraph partitioning problem. The solution for this problem is discussed next. Finally, we describe the information each worker possesses after assigned the task.

A. Hypergraph model for Task assignment

A hypergraph H is a pair $H = (N, E)$ where a hyperedge $e \in E$ can connect more than two nodes in N . Therefore, a node and a hyperedge can naturally represent a violation and a correspondence respectively. That is, in H , $N \equiv V$ and $E \equiv C$. Moreover, since each violation contains different number of correspondences, each violation-node v is associated with a weight equals the number of correspondences in it: $w(v) = |v|$.

The example below illustrates the process of constructing a hypergraph from a set of violations:

Example 1. In Figure 1, we have a network of schemas created from 3 schemas: s_1, s_2, s_3 . From this network of schemas, six violations $v_1\{c_2, c_4\}, v_2\{c_1, c_2\}, v_3\{c_1, c_5, c_6\}, v_4\{c_4, c_5, c_6\}, v_5\{c_3, c_4\}, v_6\{c_1, c_3\}$ are detected together with their correspondences. Based on this information, we can construct a violation hypergraph with 6 nodes and 6 hyperedges.

The violations and its correspondences can be modelled as a violation hypergraph as follows:

Definition 2. A violation hypergraph H_V is a pair $H_V = (N, E)$ where

- N is a set of nodes created from a set of violations $N = \{v_i \mid v_i \in V\}$ and each node v_i is associated with a weight $|v_i|$.
- E is a set of hyperedges created from a set of correspondences $E = \{e_i \subset N \mid \exists c_j \in C, \forall v_k \in e_i, c_j \in v_k\}$

With this modelling, the task balancing problem in definition 1 can be resolved as a k -way hypergraph partitioning problem as follows:

PROBLEM 1. Given a hypergraph $H_V(N, E)$ and an overall load-imbalance tolerance factor b such that $b \geq 1$, the goal is to partition the set N into k disjoint subsets, N_1, N_2, \dots, N_k such that

- The total weight of nodes in each subset N_i is bounded by $\frac{1}{b} \times \frac{\sum_{i=1}^k |N_i|}{k} \leq |N_i| \leq b \times \frac{\sum_{i=1}^k |N_i|}{k}$ where $|N_i|$ is the total weight of nodes in N_i .
- The sum of external degrees $\sum |E(N_i)|$ of a partitioning is minimized, where the external degree $|E(N_i)|$ of a

partition N_i is defined as the number of hyperedges that are incident but not fully inside this partition.

B. Hypergraph partitioning

There are various approaches for computing a k-way partition, which can be categorized into following schemes:

- **Recursive bisection paradigm:** reduces k-way partitioning problem into performing a sequence of bisections, but at least NP-hard [7]. Many heuristics algorithms have been developed, as surveyed in [2].
- **Multi-level paradigm:** In this class of hypergraph bisections algorithm [9], [15], [1], [25], an iterative refinement process is employed by constructing a sequence of successively smaller (coarser) hypergraphs. The substantially successful tool for this scheme is hMetis [1].
- **Direct-computing paradigm:** In [24], the authors showed that a method that compute k-way partitions directly produced much better than recursive method (computing k-way partitions successively via recursive bisections) in terms of optimizing objectives such as sum of external degrees, scaled cost and absorption [2]. There are many research works on this direction, most recent by K-PM/LR algorithm [3].

There is also a hybrid algorithm developed in [16], which combines direct-computing paradigm and multi-level paradigm. As claimed by the authors, this algorithm outperforms either k-way FM or K-PM/LR algorithms.

// TODO: Hypertree Decomposition

C. Collaborative resolution network

After dividing the hypergraph into many partitions, we can build a *collaborative resolution network CRN* that represents the relationship between partitions (workers) and the shared correspondences between them. A collaborative resolution network *CRN* is a graph where nodes are partitions and there is an edge between two nodes if there is at least one common correspondence between them. Each edge in a *CRN* is labeled with the common correspondences between the partitions.

Each worker u_i is responsible for resolving violations in partition N_i . Moreover, workers who share the same edge must communicate to resolve conflicts on the common correspondences. The following example illustrates a *CRN*:

Example 2. Back to example 1, if we have 3 workers and we need to find a fair task assignment for them, we can divide the hypergraph in Figure 1 into three partitions by solving the 3-way hypergraph partitioning problem: $N_1\{v_1, v_5\}$, $N_2\{v_3, v_4\}$, $N_3\{v_2, v_6\}$. The collaborative resolution network is illustrated in Figure 2.

Worker u_1 is responsible for resolving violations v_1, v_5 and she must communicate with worker u_2 to resolve any conflict on correspondence c_4 and worker u_3 on correspondences c_2, c_3 .

V. CONFLICT RESOLUTION

After partitioning the *violation hypergraph* H_V into many partitions, we have balanced the assigned tasks among users.

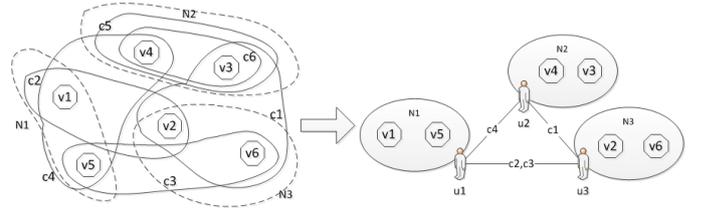


Fig. 2. A collaborative resolution network of 3 workers

In the next step, users will give feedbacks on correspondences in their partitions to correct violations. A feedback given by user is a set of possible target attributes of one source attribute.

A source attribute a might be shared by multiple users. In order to reach agreement on target attribute for a , these users must participate in a negotiate process divided into rounds of message exchange. In each round, every user propose a set of possible target attributes of a . At the end of each round, every user has received all the proposals and try to reduce the number of possible target attributes of a .

When users' preferences are common knowledge, negotiation process can reach a fix-point of agreement using distributed protocols like *monotonic concession protocol* (MCP) [22]. In the following sections, we encode this negotiation process by designing a negotiation language and a MCP specified for schema matching problem. In our simple version, we assume that users are negotiating over a single source attribute only. The extension to multiple source attributes is straightforward and natural.

A. Model of negotiation process

A negotiation process can be modeled as follows. The basic elements of a negotiation process are *users* and *proposals*:

- **Users:** are denoted by $U = \{u_1, u_2, \dots, u_n\}$. Users communicate to each other by exchanging proposals in order to agree on a common variable x (no shared memory).
- **Channels (multi-way channels):** are the channels which is used to broadcast the proposals of $u_i \in U$ to **all other users** of U . The channels are assumed to be reliable (no message loss, no message corrupted, no message duplicated).
- **Proposals:** denoted by φ which is proposed by user u on a certain variable. For clarity, we will later provide a formal definition of proposal in section V-B.
- **User actions:** each user has a set of actions that they can provoke to exchange the proposals to each other:
 - $\text{send}(\varphi)$: sending of proposal φ
 - $\text{receive}(\varphi)$: reception of proposal φ

Schema matching is the problem of finding an attribute of target schema for a given source attribute. Thus, we present a source attribute a by a variable $x(a, s)$, where s is target schema. A value of x is a certain target attribute of s . The domain of x is the set of all attributes of s .

We model the negotiation process h as a finite sequence of rounds $h = \{r_0, r_1, \dots, r_k\}$. At each round, every user puts forward a proposal. A round starts when a user proposes a new proposal and ends when all users in U have given

their proposals. A round r_k consists of a tuple of proposals $\langle \varphi_1^k, \varphi_2^k \dots \varphi_n^k \rangle$ where φ_i^k is the proposal made by user u_i at the k -th round.

B. Negotiation Proposal

A negotiation proposal φ for a variable x can be presented as a set of possible values of x as follows. //TODO

The example below illustrates a proposal:

Example 3. Let x be a shared variable and its domain $D(x)$ is $D(x) = \{ \text{"LastName"}, \text{"FirstName"}, \text{"Name"}, \text{"MiddleName"} \}$. Some proposals for x that a user can propose are:

- $\varphi_1 \triangleq x = \text{"LastName"} \vee x = \text{"BillingAddress"}$
- $\varphi_2 \triangleq x = \text{"LastName"} \vee x = \text{"BillingAddress"} \vee x = \text{"Name"} \vee x = \text{"MiddleName"}$
- $\varphi_3 \triangleq x = \text{"Name"} \vee x = \text{"MiddleName"}$

Based on the proposals proposed by users in the last round, we can find out if an agreement has been made.

Definition 3. Given a negotiation process h , an agreement has been made if the following condition holds

$$\varphi_1^{|h|} \Leftrightarrow \dots \Leftrightarrow \varphi_k^{|h|}$$

C. Monotonic Concession Protocol (MCP)

In the negotiation problem we consider a set of n users, each user u_i with proposal φ_i . These n users have to negotiate to agree on a common value v that is the belief value of one of the proposals.

1) *Protocol properties:* A protocol defines the rules of negotiation process between users. It specifies the proposals that each user is allowed to make as a function of prior proposals.

PROBLEM 2. Given a negotiation process, designing a protocol that satisfies the following properties:

- *Termination:* A protocol must ensure that the negotiation process ends after a finite number of rounds. Formally, the protocol must ensure the negotiation process ends after a polynomial rounds in the size of D and U .
- *Success:* A protocol must guarantee success which means an agreement must be reached after the negotiation process ends.

A success-guaranteed protocol is a special case of a terminal protocol where a protocol must also end with an agreement. Given a negotiation process h that follows a protocol, we can check its status as follows: given a function $\pi(r) = \bigcap_{\varphi_i \in r} \varphi_i$ which indicates the agreed values at the round r , a negotiation process h is

- Terminal if $|\pi(r^{|h|})| = 0$
- Successful if $|\pi(r^{|h|})| = 1$
- Running if $|\pi(r^{|h|})| > 1$

Or

- Terminal if $\exists \varphi_1, \dots, \varphi_k \in r, \varphi_1^{|h|} \Leftrightarrow \dots \Leftrightarrow \varphi_k^{|h|} \Leftrightarrow \emptyset$
- Successful if $\exists \varphi_1, \dots, \varphi_k \in r, \varphi_1^{|h|} \Leftrightarrow \dots \Leftrightarrow \varphi_k^{|h|} \Leftrightarrow \{v\}$
- Running if otherwise

Therefore, a protocol must show improvement after each run to guarantee termination. This can be done by limit the proposals that can be made by a user.

Observation 1. A user must

- Show improvement in his proposals. That means the proposal in each round must be more preferable than previous rounds.
- Not contradict himself. He can not propose new proposals that violates previous proposals.

Given this observation, we add a preferable constraint on the proposals for Protocol ?? to ensure termination.

Protocol 1. A monotonic concession protocol is a simple protocol with an additional constraint: given a negotiation process h , a proposal $\varphi_i^{|h|}$ is valid if

$$\forall r \in N, \varphi_i^r \succ \varphi_i^{r-1}$$

In other words, a proposal is valid if it is more preferable than the previous proposals made by the same user. We can prove that the MCP protocol ends after a finite of rounds.

Theorem 1. Protocol 1 is terminal if one of two following conditions is satisfied

- 1) $\forall h, |\pi(r^{|h|})| = 0$
- 2) $\forall h, |\pi(r^{|h|})| = 1$

Proof: Since a user can not make the same proposal twice but the later proposal must be more preferable than previous proposals, we ensure that the protocol ends after a finite set of rounds. ■

However, by ensuring efficiency, we have compensated agreement over efficiency. The theorem below proves that the monotonic concession protocol is terminal but does not guarantee success:

Theorem 2. Protocol 1 does not guarantee success: $\exists h, |\pi(r^{|h|})| \neq 1$

Proof: The proof is straightforward. We need to find two negotiation histories that follows MCP that one is successful and the other terminates but unsuccessful. ■

In order to guarantee success, we need to force an agreement if the negotiation process ends unsuccessfully. We use majority voting to force an agreement between users. Let $P_r = \bigcup \llbracket \varphi_i \rrbracket$ denotes the proposed values of all users at round r and $\sigma_r(v) = \sum_{v \in \llbracket \varphi_i \rrbracket} 1$ indicates the number of users propose v at round r . The forced agreement v of the negotiation process h is the value that has the most proposals: $\forall v' \in P_r, \sigma_r(v') < \sigma_r(v)$.

2) *Relaxed Monotonic Concession (RMC) Protocol*): The observation 1 may not be realistic since it forces a user not to contradict himself and requires his proposals to be monotonic conceding. Therefore, we replace the validity condition in Protocol 1 with a loosening form:

Protocol 2. A relaxed monotonic concession protocol is a simple protocol with an additional constraint: given a negotiation process h and a function $\varsigma(r) = \Sigma \llbracket \varphi_i \rrbracket$, h is valid if

$$\forall r \in [1, |h|], \varsigma(r-1) > \varsigma(r)$$

However, the RMC protocol requires an assessment of the global state which can only be evaluate with a coordinator. In this paper, we assume that a coordinator is available and responsible for providing ς value to each user at each round.

Theorem 3. *Protocol 2 does not guarantee success:*
 $\exists h, |\pi(r^{|h|})| \neq 1$

Proof: The proof is straightforward. We need to find two negotiation histories that follows MCP that one is successful and the other terminates but unsuccessful. ■

Since the RMC protocol does not guarantee success, we must also apply majority voting to get a forced agreement as described in Section ??.

VI. EXPERIMENTS

A. Experimental setup

We have used 4 real-world datasets (see Table II) for our experiments. The first three datasets (BusinessPartner, PurchaseOrder, UniversityAppForms) have ground truth but the THALIA one does not. The datasets are available at <http://lsirpeople.epfl.ch/qvnhnguye/smart/>. In the following we shortly describe these datasets.

- *Business Partner:* The Business Partner are 3 schemas originated from SAP that model business partners in SAP ERP, SAP MDM and SAP CRM systems. Each schema has 80 attributes and 3 levels of hierarchy.
- *PurchaseOrder:* We collected, extracted and normalized purchase order e-business documents from various resources, including COMA evaluations [?], openTRANS¹, xCBL², RosettaNet³, SAP-PO⁴, CRF⁵.
- *UniversityApplicationDataset:* University Application Form (UAF): We collected and extracted XML schemas representing the web interface of American universities' application form. Although we visited around 50 university websites but only be able to obtain 15 schemas since most of them use the same platforms, such as CommonApp⁶, UniversalApp⁷, Embark⁸, CollegeNet⁹.
- *THALIA:* This dataset [8] is available at <http://www.cise.ufl.edu/research/dbintegrate/thalia/>. It provides 44 XML schemas of learning courses of 44 famous universities all over the world. Each schema has around 10 attributes.

We worked with a state-of-the-art Hypergraph Partitioning tool, namely the hMETIS system, <http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview>, release 2007 – 05 – 25. We have used the schema matcher Coma++ [4].

We have obtained the candidate correspondences using a schema matcher (Coma++) (threshold=0.3, topK=10,

TABLE II
DATASETS

| Dataset | Nr. of schemas | Min/Max. nr. of attributes | Nr. of violations |
|--------------------|----------------|----------------------------|-------------------|
| BusinessPartner | 3 | 80/106 | 252 |
| PurchaseOrder | 10 | 35/408 | 11688 |
| UniversityAppForms | 15 | 65/228 | 70347 |
| THALIA | 44 | 3/18 | 5063 |

delta=0.5) for each involved pair of schemas. In the experiments, the interaction graph of schema matching network is a complete graph (i.e. clique graph) and we assume the one-to-one constraint and the circle constraint to hold. All the violations are detected with respect to these constraints. The task assignment starts with the complete set of violations. We obtain the figures by repeating the experiments in 5 runs.

B. Experimental Metrics

In our experiment we are particularly interested in measuring the processing time of the task assignment process and the costs of negotiation process based on partitioning result.

Processing time. We measure the processing time in terms of partitioning process performed by hMETIS. We ignore all other I/Os time since the partitioning time essentially illustrates the task assignment process to assign violations to each user.

Communication cost. After partitioning, users participate in negotiation process to reach agreements. We measure the communication cost in terms of exchanging messages (i.e., overlapping correspondences) between each pair of users. The formula was given in Definition 1: $msg(T) = \sum |C_i \cap C_j|$. Note that this metric could be greater than the total number of correspondences in schema matching network.

Negotiation cost. We measure the number of conflicts; that is, the number of common correspondences shared by two or more users to be resolved by negotiation protocol. The formula is given by $conf(T) = \bigcup (C_i \cap C_j)$. Note that this metric must be less than the total number of correspondences in schema matching network.

C. Experimental results

1) *Trade-off between workload-balancing and communication-minimizing:* In this section we study the relation between workload balance and communication minimizing in fair task assignment (definition 1). We show that in order to reach the workload balance between users, it is necessary to sacrifice the cost of communication between users for negotiation.

2) *Effects of network size:* We are interested in how does the processing time of task assignment depends on the number of violations. In this experiment we apply our fair task assignment method in different datasets, using same set of constraints to detect violations. In the other words, we repeat the task assignment method in the case of networks of different size, but of the same topology, the complete graph. Finally, we measure the processing time, relative to the number of all violations.

¹openTRANS E-business document standards <http://www.opentrans.de/>

²XML Common Business Library

³The RosettaNet Standard, <http://www.rosettanet.org/>

⁴SAP Purchase Order Standard, <http://www.sap.com>

⁵Centro Ricerche Fiat <http://www.crf.it>

⁶Common Application <https://www.commonapp.org>

⁷Universal College Application <https://www.universalcollegeapp.com/>

⁸Embark, <http://www.embark.com>

⁹CollegeNet <http://www.collegenet.com/elect/app/app>

D. Notes

Metrics:

- *Overlapping correspondence*: is the correspondence which is shared by two or more users.
- *Workload balance*: check the workload of each user is balance or not.
- *Number of users*: is the number of partitions k in k -way hypergraph partitioning problem.
- *Processing time*: is the elapsed time of task assignment, that is, the elapsed time of hypergraph partitioning.

Parameters:

- *The number of violations*: affected by the number of correspondences in schema network. How does the processing time of hypergraph partitioning scale with the number of violations (i.e., the number of violation-nodes in hypergraph).
- *The number of correspondences in one violation*: affected by the number of schemas in schema network. How does the processing time scale with the number of hyper edges of hypergraph.

Experiments:

- Finding appropriate number of users
- Effects on total exchanging messages
- Processing time

VII. RELATED WORK

We model the collaborative reconciliation on schema matching network. In this context, the user inputs are encoded as propositional formulae. Following logical argumentation, we can infer consequences from encoded formulae. Each inference is represented by an argument, in which the *claim* is a consequence (indirect approval or disapproval of a correspondence) and the *support* is the respective explanation. On top of the arguments, we analyze the relationships (*attacks*) between them and construct an *argumentation framework* to detect conflicts in inputs. Second, we propose a method to guide conflict resolution. From arguments, we generate all possible decisions and rank them according to different criteria. Moreover, to enhance the trust of users in their own decisions and those of the others, we support what-if analysis by showing the effects of decisions. Third, we develop ArgSM, an *argumentation-based negotiation support* framework for schema matching. In ArgSM, the schema matching problem is modeled in terms of Answer Set Programming (ASP). Based on this model, we generate arguments and compute attacks. Moreover, our framework provides two insight views: *Schema view* (human-oriented), and *Argumentation view* (technical). They are displayed side by side in a unified graphical user interface (GUI). This helps the users to review the inputs and make decisions effectively [20], [19], [10], [21], [13], [12], [14].

VIII. CONCLUSION

We presented a negotiation protocol to enable negotiation within our tool. We would like to extend the notion of proposed constraints and consider further integrity constraints

that are relevant in the praxis (e.g., functional dependencies, domain-specific constraints). We would like to apply our methods to other problems. While our work focuses on schema matching, our techniques, especially the argumentation-based reconciliation, could be applicable to other tasks such as entity resolution or business process matching.

REFERENCES

- [1] Charles J. Alpert, Jen-Hsin Huang, and Andrew B. Kahng, *Multilevel circuit partitioning*, Proceedings of the 34th annual Design Automation Conference (New York, NY, USA), DAC '97, ACM, 1997, pp. 530–533.
- [2] Charles J. Alpert and Andrew B. Kahng, *Recent directions in netlist partitioning: a survey*, Integr. VLSI J. **19** (1995), no. 1-2, 1–81.
- [3] Jason Cong and Sung Kyu Lim, *Multway partitioning with pairwise movement*, Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design (New York, NY, USA), ICCAD '98, ACM, 1998, pp. 512–516.
- [4] H.H. Do and Erhard Rahm, *COMA: a system for flexible combination of schema matching approaches*, Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 610–621.
- [5] Avigdor Gal, Michael Katz, Tomer Sagi, Matthias Weidlich, Karl Aberer, Hung Quoc Viet Nguyen, Zoltán Miklós, Eliezer Levy, and Victor Shafraan, *Completeness and ambiguity of schema cover*, CoopIS, 2013, pp. 241–258.
- [6] Avigdor Gal, Tomer Sagi, Matthias Weidlich, Eliezer Levy, Victor Shafraan, Zoltán Miklós, and Nguyen Quoc Viet Hung, *Making sense of top-k matchings: A unified match graph for schema matching*, 2012, p. 6.
- [7] Michael R. Garey and David S. Johnson, *Computers and intractability: a guide to the theory of np-completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.
- [8] Joachim Hammer, Mike Stonebraker, and Oguzhan Topsakal, *THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches Introduction to THALIA*, Thalia Studies In Literary Humor (2004), no. August, 1–11.
- [9] S. Hauck and G. Borriello, *An evaluation of bipartitioning techniques*, Proceedings of the 16th Conference on Advanced Research in VLSI (ARVLSI'95) (Washington, DC, USA), ARVLSI '95, IEEE Computer Society, 1995, pp. 383–.
- [10] Nguyen Quoc Viet Hung, Saket Sathe, Duong Chi Thang, and Karl Aberer, *Towards enabling probabilistic databases for participatory sensing*, CollaborateCom, 2014, pp. 114–123.
- [11] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Zoltan Miklos, and Karl Aberer, *On leveraging crowdsourcing techniques for schema matching networks*, DASFAA, 2013, pp. 139–154.
- [12] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Chau Vinh Tuan, Tri Kurniawan Wijaya, Zoltan Miklos, Karl Aberer, Avigdor Gal, and Matthias Weidlich, *Smart: A tool for analyzing and reconciling schema matching networks*, ICDE, 2015, pp. 1488–1491.
- [13] Nguyen Quoc Viet Hung, Duong Chi Thang, Matthias Weidlich, and Karl Aberer, *Erica: Expert guidance in validating crowd answers*, SIGIR, 2015, pp. 1037–1038.
- [14] Nguyen Quoc Viet Hung, Duong Chi Thang, Matthias Weidlich, and Karl Aberer, *Minimizing efforts in validating crowd answers*, SIGMOD, 2015, pp. 999–1014.
- [15] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, *Multilevel hypergraph partitioning: applications in vlsi domain*, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **7** (1999), no. 1, 69–79.
- [16] George Karypis and Vipin Kumar, *Multilevel k-way hypergraph partitioning*, Proceedings of the 36th annual ACM/IEEE Design Automation Conference (New York, NY, USA), DAC '99, ACM, 1999, pp. 343–348.
- [17] Hung Quoc Viet Nguyen, Tri Kurniawan Wijaya, Zoltán Miklós, Karl Aberer, Eliezer Levy, Victor Shafraan, Avigdor Gal, and Matthias Weidlich, *Minimizing human effort in reconciling match networks*, ER, 2013, pp. 212–226.
- [18] Quoc Viet Hung Nguyen, XuanHoai Luong, Zoltan Miklos, ThoThanh Quan, and Karl Aberer, *Collaborative schema matching reconciliation*, CoopIS, 2013, pp. 222–240.
- [19] Quoc Viet Hung Nguyen, Thanh Tam Nguyen, Ngoc Tran Lam, and Karl Aberer, *Batc: a benchmark for aggregation techniques in crowdsourcing*, SIGIR, 2013, pp. 1079–1080.

- [20] Quoc Viet Hung Nguyen, Tam Nguyen Thanh, Tran Lam Ngoc, and Karl Aberer, *An evaluation of aggregation techniques in crowdsourcing*, WISE, 2013, pp. 1–15.
- [21] Thanh Tam Nguyen, Quoc Viet Hung Nguyen, Matthias Weidlich, and Karl Aberer, *Result selection and summarization for web table search*, ICDE, 2015, pp. 231–242.
- [22] Jeffrey S. Rosenschein and Gilad Zlotkin, *Rules of encounter: designing conventions for automated negotiation among computers*, MIT Press, Cambridge, MA, USA, 1994.
- [23] Sabyasachi Saha and Sandip Sen, *An efficient protocol for negotiation over multiple indivisible resources*, Proceedings of the 20th international joint conference on Artificial intelligence (San Francisco, CA, USA), IJCAI'07, Morgan Kaufmann Publishers Inc., 2007, pp. 1494–1499.
- [24] Horst D. Simon and Shang-Hua Teng, *How good is recursive bisection?*, SIAM J. Sci. Comput. **18** (1997), no. 5, 1436–1445.
- [25] Sverre Wichlund, *On multilevel circuit partitioning*, Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design (New York, NY, USA), ICCAD '98, ACM, 1998, pp. 505–511.