Retrieval of Semantic Workflows with Knowledge Intensive Similarity Measures

Ralph Bergmann[1] and Yolanda Gil[2]

[1] University of Trier, Department of Business Information Systems II
54286 Trier, Germany – Email: bergmann@uni-trier.de
[2] Information Sciences Institute, University of Southern California
Marina del Rey, CA 90292, USA – Email: gil@isi.edu

Abstract. We describe a new model for representing semantic workflows as semantically labeled graphs, together with a related model for knowledge intensive similarity measures. The application of this model to scientific and business workflows is discussed. Experimental evaluations show that similarity measures can be modeled that are well aligned with manual similarity assessments. Further, new algorithms for workflow similarity computation based on A* search are described. A new retrieval algorithm is introduced that goes beyond traditional sequential retrieval for graphs, interweaving similarity computation with case selection. Significant reductions on the overall retrieval time are demonstrated without sacrificing the correctness of the computed similarity.

1 Introduction

Today, workflows are an established means for modeling business processes and to automatically control their execution. Recently, workflows are more widely used for many purposes different from business process automation. *Scientific workflows* are executable descriptions of automatable scientific processes such as computational science simulations and data analyses [18]. Further, workflows are used for modeling medical guidelines, to represent project plans, or to describe information gathering strategies. Such new applications of workflows typically deal with a number of new difficulties, particularly due to an increasing number of workflows potentially relevant, an increasing complexity of the individual workflows, and an increased demand for more flexibility (*agile workflows*). To deal with those new challenges in workflow management, reasoning methods for semantically enriched workflow representations have a high potential to support workflow modeling, composition, adaptation, analysis, and optimization. This is what we call *semantic workflow reasoning*. Also case-based reasoning (CBR) has already demonstrated its high potential in semantic workflow applications [19, 15, 3, 14, 9, 7, 17]. Repositories of workflows can be used as case bases, enabling the application of case retrieval and adaptation methods.

In this paper we focus on case retrieval of semantic workflows as one method that contributes to semantic workflow reasoning. Workflow cases clearly belong

to the class of highly structured case representations, for which graph-based representations are promising [8, 6, 14–16]. A particular challenge is the *similarity-based retrieval* of semantic workflow descriptions, as workflow similarity enables to retrieve workflows that do not match the query exactly, but that are a good starting point for adaptation. However, previous work on workflow retrieval either ignores the graph structure but uses a simple flat case representation [3], use (parts of) the graph structure, but only uses textual labels instead of semantic descriptions [16, 14, 11, 8, 7], or uses semantically annotated graph structures, but a crisp matching approach rather than similarity [9].

In this paper, we present a new general framework for workflow and related similarity modeling. Workflows are represented as semantically annotated graphs, extending previous proposals for graph-based workflow representations [16, 8, 17]. The well-known local/global principle to modeling similarity measures [4, 2] is extended to these graph representations, providing a flexible means for workflow similarity modeling. This framework is domain independent and general in the sense that it covers control-flow oriented workflows such as business processes as well as data-flow oriented workflows such as scientific workflows. As with all graph-based representations, the computational complexity of the similarity computation becomes a critical issue during retrieval. As a second contribution of this paper, we developed and analyzed several approaches for similarity computation and retrieval based on different search algorithms. The third contribution is the application and experimental evaluation of the framework and the retrieval algorithms in two different application areas.

2 Semantic Workflows and Workflow Retrieval

Workflow representations typically reflect the dataflow or control flow structure among the tasks of a process. Today, various workflow representation formats are used, depending on the kind of workflow. Representation approaches for business workflows have a strong focus on the control flow, usually implementing (some of) the workflow patterns proposed by van Aalst. Typical control flow patterns are *sequence*, *and-split*, *and-join*, *xor-split*, *xor-join*, and possibly *loops*. Figure 1a shows an example of a business workflow within a University administration, according the representation used in the CAKE project [3, 16] at the University of Trier. On the other hand, scientific workflows have a strong focus on the dataflow, typically restricting the control flow to a partial ordering of the tasks. Such a simple control structure offers several advantages and has been sufficient to support a variety of applications [18]. Each task (or software component) in the scientific workflow can have input datasets and input parameters, as well as output datasets. Dataflow links indicate what data are output by a task and consumed by another task. Figure 1b shows an example of a scientific workflow describing some data mining process according to the representation in the Wings project [10] at the Information Sciences Institute. Semantic workflow representations enrich the above described workflow formats by adding meta data and constraints to the individual elements of the workflow. Ontologies are used

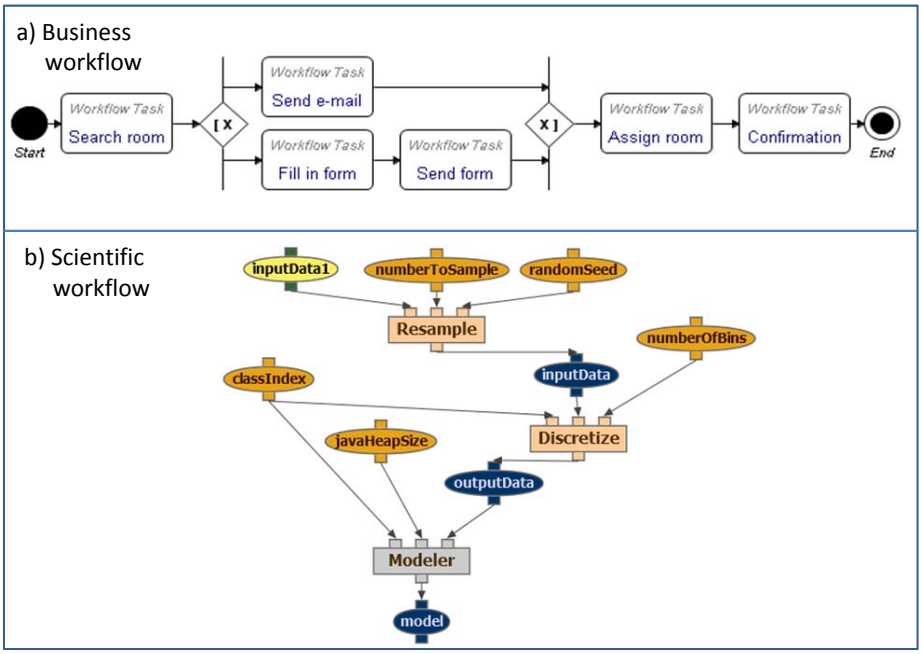

Fig. 1. Example of an administrative business workflow.

to formalize the respective workflow domain by organizing the metadata descriptions of the occurring workflow elements. This allows to capture the semantics of tasks, data items, and control flow items.

The workflow representation in the CAKE system [3] uses an object-oriented representation (originally developed for cases) for ontology representation and metadata annotation. Tasks can be organized in a hierarchy of classes, each of which contains certain properties of a task, which can be inherited from the super class. For example, the *Assign room* task includes a role description stating that the assignment must be performed by the responsible administrative department. Another property may state the average duration for task execution. The semantic workflow representations in Wings augment traditional workflow representations with semantic constraints that indicate metadata properties of workflow tasks and datasets [9, 10]. Each workflow constituent has a workflow variable associated with it, and the semantic constraints express constraints over those variables. In the case of data variables, the constraints are expressed in terms of metadata properties. For example, an input dataset may be restricted to be a discrete dataset, and if so then its variable would have a constraint stating that it has the property of being discrete. In the case of components, the constraints express how the execution of the software component changes the metadata properties of the input datasets into output datasets. Wings uses workflow reasoning

algorithms to enrich workflows by propagating the constraints throughout the workflow structure [9]. This algorithm takes simple workflow graphs created by users and automatically augments them with semantic information by adding and propagating semantic constraints that can be used for subsequent retrieval as we propose in this paper.

For semantic workflow representation some languages that include semantic constraints expressed in OWL have been proposed, such as OWL-S, WSMO, SWSL, and SWSF. These languages include task decomposition structures and complex precondition expressions. Our CAKE and Wings approaches could be used with any of those languages, though only a small subset of the constructs allowed in those languages are used in our systems.

For workflow retrieval, several CBR and related approaches exist today. The CODAW system [15] supports incremental modeling of workflows by a similarity-based reuse of the workflow templates using an HTN planner that employs an inexact, graph-based matching. Leake et al. [14] evaluate the execution paths of past workflows in order to support user extension of workflows that are under construction. Recently, a probabilistic similarity model for workflow execution paths was proposed [1]. The myGrid project[3] focuses on the discovery, reuse and repurposing of bioinformatics workflows [12]. They distinguish between direct *re-use* of an existing workflow as is and *re-purposing* an existing workflow by modifying some aspect of it. Goderis [12] defines requirements and bottlenecks for workflow re-use based on many user interviews and practical experiences.

Previous work investigated structure-less workflow retrieval based on query and workflow representation being plain textual descriptions or sets of (social) tags [12], or abstract workflow features [3]. Queries including structural properties of the workflows are treated using graph matching approaches such as subgraph isomorphism or edit-distance measures [12, 16, 14, 8]. Matching techniques based on semantic types of data has been done for the discovery of individual services or software components [13], but not for the more complex structures that workflows represent. Semantically annotated graph structures are used in workflow retrieval in Wings [9], but retrieval is restricted to crisp matching, not incorporating similarity measures.

3 Graph-based Framework for Workflow Similarity

3.1 Representation of Semantic Workflows

In line with previous work on graph-based workflow representation [5, 8], we represent workflows as semantically labeled directed graphs $W = (N, E, S, T)$ where N is a set of nodes and $E \subseteq N \times N$ is a set of edges. $T : N \cup E \rightarrow \Omega$ associates to each node and each edge a *type* from Ω. $S : N \cup E \rightarrow \Sigma$ associates to each node and each edge a *semantic description* from a semantic meta data language Σ. While Ω is fixed for the used workflow representation, Σ is domain dependent. We do not demand a particular language for Σ, but just assume some

[3] www.mygrid.org.uk

language for semantic metadata for which similarity measures can be defined (see Sect. 3.2). This general graph structure can be used to represent different kinds of workflows, such as scientific or business workflows. It is further specialized for the representation of workflows by distinguishing different types of nodes and edges, enumerated in Ω and assigned through T:

Each workflow consists of exactly one **workflow node**. The semantic description associated to a workflow node represents some overall properties of the entire workflow. This can be a workflow classification in some ontology, a set of semantic tags, or other properties related to the quality or execution requirements of the workflow.

Each task in a workflow is represented by a **task node**. Its semantic description typically classifies the task in some task ontology and may provide additional functional properties of the task. It also includes the description of workflow roles (i.e. human agents or services) for the execution of those tasks.

Each data item in a workflow is represented by a **data node**. Its semantic description classifies the data item within some data type ontology and may specify certain data properties (e.g. a data set has missing values) relevant for retrieval. Control-flow centric workflows (Fig. 1a) may have no data nodes.

Each control flow element in a workflow, such as split/join elements for and/xor blocks, are represented as a **control-flow node**. The semantic description of a control flow node specifies which control flow element is represented. Data-centric workflows (e.g. Fig. 1b) may have no control flow nodes.

The workflow node is linked to each of the other nodes by a **part-of edge**. The semantic description of such an edge specifies the role of the associated node within the overall workflow. Data nodes can be linked with the workflow node via edges having one of the following semantic description: *is-input*, *is-output*, *is-intermediate*, *is-parameter*. For example, in Fig. 1b *inputData1* is overall workflow input data and hence linked with *is-input*. *inputData* in Fig. 1b is data produced and further processed within the workflow and is hence linked with *is-intermediate*. Further, the semantic descriptions for part-of edges include *has-task* for task nodes linked to a workflow node and *has-control* for links to control-flow nodes.

The dataflow among tasks is represented using **dataflow edges**. A dataflow edge links data nodes to task nodes and vice versa. The semantic description of such an edge indicates whether the data item was *consumed* as input data or *produced* as output data by the task. All arrows shown in Fig. 1b are turned into dataflow edges in the graph.

The control-flow among tasks is represented using **control-flow edges**. Such an edge connects two task nodes or a task node with a control-flow node. An edge from node $n1$ to $n2$ indicates that node $n2$ may be executed after node $n1$. All arrows shown in Fig. 1a will be turned into control-flow edges in the graph.

Semantic constraints among properties of data nodes are represented using **constraint edges** that connect two data nodes. The semantic description includes the definition of a constraint. For example, a constraint may state that test and training data must come from the same domain.

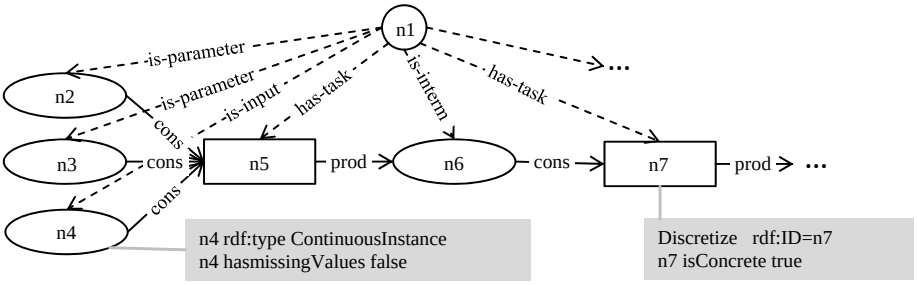

Fig. 2. Fraction of a workflow graph: the Resample and Discretize task are shown.

Figure 2 shows the graph representation of a fraction of the workflow from Fig. 1b. It shows one workflow node (circle), four data nodes (ovals), and two task nodes (boxes) as well as some of the edges. A simplified fraction of the semantic descriptions of a data and a task node are shown in the grey boxes.

In the following, we assume that a repository of workflows is given, which we consider as case-base $CB = \{CW_1, ...CW_n\}$. We aim at retrieving workflows from this case base given a particular query. This query QW is also a workflow in the above described representation. It just specifies some workflow elements together with the semantic description that are considered requirements on the workflows to be retrieved. For example, a query could state that someone is looking for workflows that first discretize some continuous data and then use the discrete data in a decision tree modeler. Hence, the query would contain a workflow node, two data notes: one specifying the continuous data and one specifying the discretized intermediate data. Further, it includes two task nodes, one that specifies a task that can discretize data and one that specifies a task which is from the class decision tree modeler. The respective edges are included in the query as well. The similarity-based retrieval we aim at, could then retrieve for example the workflow shown in Fig. 1b, although it is not a perfect match as the modeler task in this workflow does not specify which concrete modeler to be used. However, it can be potentially specialized to a decision tree modeler. A different workflow from the repository containing a J48 modeler might be a better match with respect to this task.

3.2 Modeling Workflow Similarity

In order to assess the similarity of a case workflow $CW = (N_c, E_c, S_c, T_c)$ wrt. a query $QW = (N_q, E_q, S_q, T_q)$, we need to consider the constituents of the workflow as well as the link structure. This requires similarity models that enable the comparison of workflow elements. We now present a new similarity model, which can be considered an enhancement of the well-known local/global approach for structural CBR [4, 2], particularly for object-oriented similarity measures [2]. The local similarity measures assess the similarity between two nodes or two

edges of the same type. The global similarity for workflows is obtained by an aggregation function combining the local similarity values within a graph mapping process. The local similarity assessments are based on the semantic descriptions of the nodes and edges. Therefore, we assume a similarity function:

$$sim_\Sigma : \Sigma \times \Sigma \to [0, 1].$$

The detailed formalization of this similarity measure depends on the language Σ and can itself be modeled using the local/global approach aggregating local similarity measures for the individual properties in the semantic description. In our work, we treat the semantic descriptions in an object-oriented fashion and use the well established similarity measures described in [2]. Thereby, class hierarchies of tasks and data, as well as properties like those shown in Fig. 2 can be appropriately considered in the similarity model.

Nodes of equal type have a similarity value assigned, based on the similarity of the semantic descriptions. Based on sim_Σ, we define node similarity $sim_N(n_q, n_c)$ for $n_q \in N_q$ and $n_c \in N_c$ as follows:

$$sim_N(n_q, n_c) = \begin{cases} sim_\Sigma(S_q(n_q), S_c(n_c)) & \text{if} \quad T_q(n_q) = T_c(n_c) \\ 0 & \text{otherwise} \end{cases}$$

Edge similarity is more sophisticated as it should consider not just the semantic description of the edges being compared, but also the nodes that are linked by the edges. For example, two dataflow edges should only be considered similar if they link similar data objects to similar tasks. To reflect such considerations, we define edge similarity $sim_E(e_q, e_c)$ for $e_q \in E_q$ and $e_c \in E_c$ as follows:

$$sim_E(e_q, e_c) = \begin{cases} F_E \begin{pmatrix} sim_\Sigma(S_q(e_q), S_c(e_c)), \\ sim_N(e_q.l, e_c.l), \\ sim_N(e_q.r, e_c.r) \end{pmatrix} & \text{if} \quad T_q(e_q) = T_c(e_c) \\ 0 & \text{otherwise} \end{cases}$$

For an edge e, the expression $e.l$ denotes the left node of the edge and $e.r$ denotes the right node. The function F_E is an *aggregation function* that combines the semantic similarity between the edges and the similarity of the connected nodes to the overall similarity value. In our implementations we define F_E as follows: $F_E(S_e, S_l, S_r) = S_e \cdot 0.5 \cdot (S_l + S_r)$.

The overall workflow similarity between QW and CW is now defined by means of a *legal mapping*, i.e., a type-preserving, partial, injective mapping function $m : N_q \cup E_q \to N_c \cup E_c$ that satisfies the following five constraints:

$$\begin{array}{lll} T_q(n_q) = T_c(m(n_q)) & T_q(e_q) = T_c(m(e_q)) & \\ m(e_q.l) = m(e_q).l & m(e_q.r) = m(e_q).r & \forall x, y\ m(x) = m(y) \to x = y \end{array}$$

Please note that in such a legal mapping m a case node or edge can only be the target of one query node or edge, respectively. Since the mapping can be partial, not all nodes and edges of the query must be mapped to the respective case elements. Also an edge can only be mapped if the nodes that the edge connects

are also mapped to the respective nodes which are linked by the mapped edge. The similarity between QW and CW wrt. a mapping m is now defined using a second aggregation function F_W for workflows as follows:

$$sim_m(QW, CW) = F_W \begin{pmatrix} (sim_N(n, m(n))|n \in N_q \cap \mathrm{Dom}(m)), \\ (sim_E(e, m(e))|e \in E_q \cap \mathrm{Dom}(m)), \\ |N_q|, |E_q| \end{pmatrix}$$

The aggregation function combines the individual similarity values for mapped nodes and edges to an overall similarity value wrt. the mapping m. $\mathrm{Dom}(m)$ denotes the domain of m. The parameters $|N_q|$ and $|E_q|$ enables F_W to consider partial mappings, i.e. nodes and edges not mapped should not contribute to the overall similarity. In our implementation we define F_W as follows:

$$F_W((sn_1, \ldots, sn_i), (se_1, \ldots, se_j), n_N, n_E) = \frac{sn_1 + \cdots + sn_i + se_1 + \cdots + sn_j}{n_N + n_E}$$

Please note that each mapping m can be interpreted as a particular suggestion for the reuse of the case workflow: the mapped case nodes and edges should be considered solution elements for the respective nodes and edges in the query. The similarity value for the mapping assesses the utility of this reuse opportunity. For a case there are usually several mappings that reflect different ways of reusing the case. Hence, the overall workflow similarity $sim(QW, CW)$ is determined by the best mapping (see also [5]), i.e the mapping with the highest similarity:

$$sim(QW, CW) = \max\{sim_m(QW, CW)| \text{ mapping } m\}.$$

To summarize, the presented similarity model for workflows is defined by three *model parameters*, 1^{st} the similarity function sim_Σ for semantic descriptions, 2^{nd} the aggregation function for edges F_E, and 3^{rd} the aggregation function for workflows F_W.

3.3 Experimental evaluation

We now focus on the question, whether the computed similarity values are appropriate within the context of a CBR application aiming at retrieving workflows for reuse. For this purpose, the similarity measures should be able to approximate the utility of the cases wrt. the problem formulated within the query (see [2], p.94*ff.*) in the context of the concrete application. Moreover, a good and broadly applicable similarity model should enable to flexibly determine appropriate model parameters for many applications. Whether this is the case, can of course only be assessed by analyzing a series of real-world applications. As a first step towards this goal, we developed two initial workflow retrieval applications in two complementary domains: *administrative business workflows* (similar to Fig. 1a) and *scientific data mining workflows* (similar to Fig. 1b). Based on our previous work [17, 9] we developed for each domain an ontology for workflows, tasks (and data) as well a case base of 20 workflows. For both domains, the average number of nodes in the cases is 10; the average number of edges is 18.

The aim of the first experiment is to show, whether it is possible to model similarity measures that approximate the utility assessment of a human user. More precisely, the hypothesis is:

H1: *The similarity model enables to define model parameters such that the computed similarity is better aligned with an expert's assessment than a lexical similarity measure.*

For each domain we determined 10 queries that are related to the 20 cases in the case base. For each query, an expert was asked to select the 5-6 best suitable cases in the case base and to determine a partial order among those best cases. For each domain, a similarity model was developed by hand: similarity values within sim_Σ have been adjusted manually and the functions F_E and F_W are those mentioned before in this section. For each query, we computed the similarity of each case in the case base by applying an exhaustive search algorithm that computes all mappings, thereby producing an optimal solution. As baseline for the comparison, we computed the similarity by a lexical similarity measure, in particular a Levenshtein similarity measure on the task and data names. Besides this, the same functions F_E and F_W were used.

For both similarity measures, we compared the ordering of the cases with those of the human expert. As a measure of correctness we used the ranking measure proposed by Cheng et al. [6]: They define the *correctness* and *completness* of an ordering $\sqsupset$ with respect to a reference order $\sqsupset_*$ based on the concordance C of the two orders, i.e., by the number case pairs ordered equivalently by both orders and by the discordance D, i.e. by the number of case pairs for which both orders differ. Correctness and completeness are defined as follows: $Corr = (C - D)/(C + D)$ and $Compl = C/| \sqsupset_* |$.

In our evaluation we determine for each of the 10 query workflows QW the order on the 20 cases through: $CW_1 \sqsupset CW_2$ iff $sim(QW, CW_1) > sim(QW, CW_2)$. The partial reference order $\sqsupset_*$ is given by the human ranking for the same query. The value for correctness is within the interval [-1,1]. If it has a value of 1 then every ordering stated in $\sqsupset$ is correct, if it has a value of -1, every ordering stated in $\sqsupset$ contradicts the reference order. A value of 0 indicates that there is no correlation between both orders. The completeness value is within the interval [0,1]. The higher the value the more orderings are contained in $\sqsupset$ (compared to situations in which $\sqsupset$ does not distinguish the cases due to equal similarity). Figure 3 shows the results of the evaluation for the semantic similarity measure

	Administrative business workflows			Scientific data mining workflows		
Similarity	Correct	Complete	Retrieval Time	Correct	Complete	Retrieval Time
Semantic	0.708	0.910	24.00 sec	0.840	0.693	8.00 sec
Lexical	0.542	0.911	24.36 sec	0.593	0.751	7.78 sec

Fig. 3. Similarity Measure Evaluation.

compared to the Levenshtein measure for both domains. The averaged correctness and completeness values over all 10 queries are displayed. The figure clearly shows an improved correctness for the semantic similarity measure in both domains, while the completeness slightly suffers for the data mining workflows. This clearly confirms the hypothesis H1.

Additionally, Fig. 3 shows the average retrieval time over all 10 queries using linear retrieval over the full case base, applying exhaustive search for similarity assessment. These figures clearly show that this approach is computationally unacceptable for practical applications. This motivates our work on a more scalable framework, described in the next section.

4 Similarity Computation and Workflow Retrieval

While similarity computation by exhaustive search guarantees to find the optimal match, it is computationally not feasible. Greedy search is proposed as alternative search algorithm for similarity assessment of workflow graphs [14, 8, 5] as it enables to quickly find a local optimum. However, the resulting error can hardly be controlled as the local optimum can differ significantly from the global optimum, hence producing too low similarity values. The A* search algorithm promises to be good alternative over the above mentioned approaches, given a well-informed admissible heuristic function can be determined. In the following, we will develop several A* search variants and demonstrate their performance and similarity error in experimental evaluations.

4.1 Similarity Computation by A* Search

The A* algorithm maintains a priority queue of open search nodes. In each step, the first (best) node in the queue is removed. If it represents a solution, A* terminates. Otherwise this node is expanded, i.e. each successor node in the search is determined and inserted into the priority queue. When applied for finding the best match m between a query and a case graph elements, a search node S basically represents the current mapping $S.m$ as well as the not yet mapped nodes $S.N$ and edges $S.E$. During node expansion, the mapping $S.m$ is extended in all possible ways by one additional mapping of a node or edge. The order in which the expanded nodes are inserted into the priority queue is essential for A*. Therefore, each search node S is evaluated by a function $f(S) = g(S) + h(S)$. In the traditional formulation, A* aims at minimizing cost, hence $g(S)$ are the cost already occurred and $h(S)$ is a heuristic estimation function for the remaining cost to the solution. As we apply A* for maximizing the similarity value, the functions must be interpreted differently, i.e. $g(S)$ is the similarity of the current mapping S, while $h(S)$ is an heuristic estimation of the additional similarity that can be achieved through the mapping of the remaining nodes and edges. Nodes are inserted into the priority queue in decreasing order of $f(S)$, i.e. the search node with the highest f-value is expanded first. To achieve an admissible heuristic estimation function, $h(S)$ must be an upper bound of

the similarity. In the A* algorithm shown below, the value $f(S)$ is also stored in the search node, noted as $S.f$. The following A* search algorithm (divided into the top level looping algorithm and the separate expansion function) is called with a query workflow QW and a case workflow CW and returns the similarity value.

*A*Search*$(QW = (N_q, E_q, T_q, S_q), CW = (N_c, E_c, T_c, S_c))$
{ $S_0.N = N_q$; $S_0.E = E_q$; $S_0.m = \emptyset$; $S_0.f = 1$; $Q =< S_0 >$;
while *first*$(Q).N \neq \emptyset$ **and** *first*$(Q).E \neq \emptyset$ **do** { Q = *Expand*(Q) };
return(*first*$(Q).f$); }

Expand(Q)
{ S = *first*(Q); Q = *rest*(Q); x_q =*select*(S);
forall $x_c \in E_c \cup N_c$ s.th. the mapping $S.m \cup (x_q, x_c)$ is legal **do**
{ $S'.m = S.m \cup (x_q, x_c)$; $S'.N = S.N \setminus \{x_q\}$; $S'.E = S.E \setminus \{x_q\}$;
$S'.f = sim_{S'.m}(QW, CW) + h(S')$;
Q =*insert*(S', Q); }
Return Q; }

Here, *first*(Q) is the first priority queue node, *rest*(Q) removes the first node and returns the rest and *insert*(S', Q) inserts the new node S' into Q according to the f-value. During insert, the maximum size of the queue can be restricted (maximum queue size) to cut some branches of the search tree to improve performance on the risk of loosing global optimality. The *select* function determines the next node or edge to be expanded. x_q can be either a query node or edge.

We developed two versions of A* with different estimation and select functions. The first version *A* I* uses a naive estimation function which assesses the similarity of each not yet mapped node or edge as 1. Assuming the aggregation function F_W shown in Sect. 3.2, the contribution of each not mapped element to the overall similarity is computed by h_I. The select function first matches all nodes, before the edges are selected. Which element from $S.N$ or $S.E$ is selected is not determined; we choose randomly according to an internal node/edge id.

$$h_I(S) = \frac{|S.N|+|S.E|}{|N_q|+|E_q|}$$

$$select_I(S) = \begin{cases} n_q \in S.N & \text{if } S.N \neq \emptyset \\ e_q \in S.E & otherwise \end{cases}$$

The second version *A* II* uses a better informed admissible heuristic. For each not mapped query node or edge it determines the maximum possible similarity a mapping can achieve independent of the mapping of the other nodes or edges. These values can be computed prior to search and cached. Also it aims at matching edges as soon as possible. As matching edges requires the connecting nodes being matched already, the edge expansion does lead to a low branching factor. It is 1 if between two nodes there is at most one edge per type. Hence the size of the queue does not increase, while the accuracy of $f(S')$ increases.

$$h_{II}(S) = \sum_{x \in S.N \cup S.E} \left(\max_{y \in N_c \cup N_e} \{sim_{E/N}(x,y)\} \right) \cdot \frac{1}{|N_q|+|E_q|}$$

$$select_{II}(S) = \begin{cases} e_q \in S.E & \text{if } e_q.l \notin S.N \text{ and } e_q.r \notin S.N \\ n_q \in S.N & otherwise \end{cases}$$

4.2 Parallelized A* Retrieval

While the described A* algorithms aim at improving performance of a single similarity assessment, linear retrieval with large case bases will still require one run of the search algorithm per case in the case base. To ensure better scalability with growing case bases, we now propose a parallelized A* II variant, called *A*P*. It enables to compute the top k cases from the case base without fully computing the similarity for all cases. Therefore the search process is parallelized for all cases, maintaining one queue Q_i for each case. In every step, the node from the queue with the highest f-value from all queues of not already finished search processes is expanded. Search terminates, when at least k searches have terminated and when the similarity of the k-best case is higher than all f-values of the remaining queues. Since the f-values are upper bounds for the final similarity, it is ensured that none of the remaining cases can ever exceed the similarity of the known k-best case. Hence, completeness of k-best retrieval is guaranteed. The following algorithm returns the list of the k-best cases (and possibly some more) together with its similarity value.

*A*P Retrieval*$(QW = (N_q, E_q, T_q, S_q), CB = \{CW_1, \ldots, CW_n\}, k)$
{ $S_0.N = N_q$; $S_0.E = N_E$; $S_0.m = \emptyset$; $S_0.f = 1$;
for $i = 1 \ldots n$ **do** { $Q_i =< S_0 >$ } ;
res $= \emptyset$;
repeat
{ $j = \arg\max_{i \notin res}\{$*first*$(Q_i).f\}$;
$Q_j =$*Expand*(Q_j) ;
if *first*$(Q_j).N = \emptyset$ **and** *first*$(Q_j).E = \emptyset$ **then** res = res $\cup \{j\}$;
} **until** $|res| \geq k$ **and** k-th(*first*$(Q_i).f | i \in$res$) \geq \max_{j \notin \text{res}}\{$*first*$(Q_j).f\}$;
return $\{($*first*$(Q_i).f, i) \mid i \in$res$\}$ }

4.3 Experimental Evaluation

We evaluated the performance of the three A* variants also in relation to exhaustive search and greedy search. The hypotheses to be evaluated are:

H2a: *The average retrieval time of A*I is shorter than exhaustive search.*
H2b: *The average retrieval time of A*II and A*P is shorter than of A*I.*
H2c: *The average similarity error of A*I,II,P are lower than of greedy search.*
H2d: *The average retrieval time of A*P is lower than A*II, if* $k << |CB|$.

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			795.269	
Greedy	1	20	0.112	0.221
A* I	10	20	0.279	0.092
	100	20	3.829	0.102
	300	20	21.075	0.091
A* II	1	20	0.254	0.023
	10	20	0.792	0.011
	100	20	15.051	0.005
A* P	1	5	0.226	0.018
	10	5	0.584	0.008
	100	5	9.332	0.004

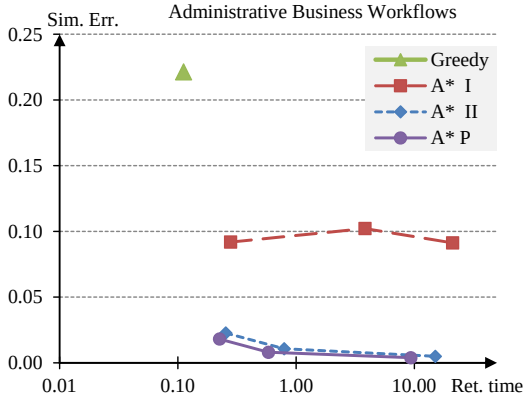

Fig. 4. Retrieval performance: Administrative business workflows

Retrieval Method	Qsize	k	Retrieval time [sec]	Similarity Error
Exhaustive			937.500	
Greedy	1	20	0.028	0.100
A* I	10	20	0.415	0.090
	100	20	5.244	0.069
	300	20	19.937	0.055
A* II	1	20	0.345	0.037
	10	20	0.852	0.020
	100	20	7.093	0.007
A* P	1	5	0.311	0.005
	10	5	0.667	0.007
	100	5	5.663	0.004

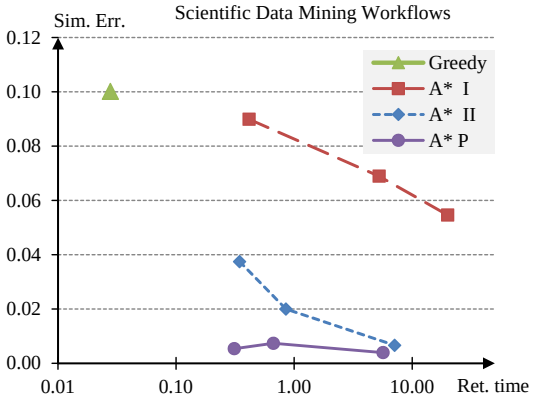

Fig. 5. Retrieval performance: Scientific data mining workflows

We tested the hypotheses for the two workflow domains. The similarity models and the case base of 20 cases from Sect. 3.3 are used. To assess the retrieval performance we used 30 queries for each domain: the 10 queries from Sect. 3.3 plus the 20 cases of the case base itself. We determined the average retrieval time per query as well as the average similarity error of the retrieved cases. As a base line for this we determined for each query and case the optimal similarity value by running the search algorithms over a very long time period.

Figures 4 and 5 show the results for the two domains for the different A* variants with different limits on the queue size. A*P is evaluated for $k = 5$ (out of 20 cases). For the other algorithms the performance does not depend on the number of cases to be retrieved. The graphs plot the similarity error over the retrieval time (logarithmic scale). Increasing the queue size limit clearly leads

to an increased retrieval time, but also to a reduced error. The figures clearly indicate that the hypotheses H2a,b and c are confirmed. Concerning H2d, the advantage of A*P over A*II is not very dominant. Therefore, we performed an additional experiment with a case base of 203 cases from the scientific data mining domain used in [9]. Figure 6 shows the average retrieval time for different values of k. It clearly confirms hypothesis H2d.

	Retrieval Time [sec]		
Qsize	A* P, k=5	A* P, k=10	A* II
10	0,386	0,483	3,04
100	0,667	3,98	14,15

Fig. 6. Retrieval performance: Scientific data mining workflows, 203 cases

5 Conclusion and Future Work

We generalize and extend previous approaches for workflow similarity using graph-based representations in several ways. We explicitly cover data centric as well as control-flow centric workflows. We link with semantic representations and introduce knowledge intensive similarity measures according to the local/global principle. Further extensions are desirable to represent hierarchical workflows. The proper treatment of workflow agility requires representing workflow instances (rather then templates as in the current approach) including the execution state. Also, more practical experience with applications of the model is needed, involving semantic models of a larger scale. This also raises the question of methodologies for developing appropriate similarity models. New methods for learning similarity measures are demanded as a tool for this purpose.

The developed similarity assessment and retrieval algorithms show a satisfying performance in terms of computation time and retrieval error. We demonstrated scalability to medium sized case bases, but an extension to case base sizes $\gg 1000$ may require a course-grained pre-selection of cases according to the MAC/FAC idea, as proposed by Leake et al. [14]. A case retrieval net over the semantic descriptions seems a suitable approach for this purpose to be investigated in future research. Also, the extension of A*P towards a multi-threaded variant exploiting the parallelization of multi-core CPUs seems promising. Future work could also explore whether kernel methods for labeled graphs are a suitable alternative approach for similarity assessment of workflows.

Acknowledgement This work is the outcome of a sabbatical visit of the first author at the Information Sciences Institute in 2010. This research was funded in part under grant IIS-0948429 from the US National Science Foundation. The authors appreciate the fruitful discussion on the topic of the paper by the various groups' members in Trier and Marina del Rey.

References

1. Becker, J., Bergener, P., Breuker, D., Räckers, M.: On measures of behavioral distance between business processes. In: Proceedings of the 10th International Conference on Wirtschaftsinformatik. vol. Vol. 2, pp. 665–674 (2011)
2. Bergmann, R.: Experience Management - Foundations, Development Methodology, and Internet-Based Applications, vol. LNAI 2432. Springer (2002)
3. Bergmann, R., Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Case-based support for collaborative business. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, A.H. (eds.) Advances in CBR. vol. 4106, pp. 519–533. Springer (2006)
4. Burkhard, H.D., Richter, M.M.: On the notion of similarity in case based reasoning and fuzzy theory. Soft computing in case based reasoning (2000)
5. Champin, P.A., Solnon, C.: Measuring the similarity of labeled graphs. CBR Research and Development p. 1066–1067 (2003)
6. Cheng, W., Rademaker, M., Baets, B.D., Hüllermeier, E.: Predicting partial orders: ranking with abstention. In: Machine Learning and Knowledge Discovery in Databases. p. 215–230 (2010)
7. Chinthaka, E., Ekanayake, J., Leake, D., Plale, B.: CBR based workflow composition assistant. In: World Conference onServices-I,. p. 352–355 (2009)
8. Dijkman, R., Dumas, M., Garcia-Banuelos, L.: Graph matching algorithms for business process model similarity search. In: Business Process Management. p. 48–63 (2009)
9. Gil, Y., Kim, J., Florez, G., Ratnakar, V., González-Calero, P.A.: Workflow matching using semantic metadata. In: Proceedings of the 5th International Conference on Knowledge Capture. p. 121–128 (2009)
10. Gil, Y., Ratnakar, V., Kim, J., González-Calero, P., Groth, P., Moody, J., Deelman, E.: Wings: Intelligent Workflow-Based design of computational experiments. IEEE Intelligent Systems 26(1), 62–72 (2011)
11. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-science and a graph-based solution. International Journal of Web Services Research 5(4) (2008)
12. Goderis, A.: Workflow Re-use and Discovery in Bioinformatics. Ph.D. thesis, University of Manchester (2008)
13. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding semantic matching of stateless services. In: Proceedings Of The National Conference On Artificial Intelligence. vol. 21, p. 1319 (2006)
14. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based support for e-Science workflow generation by mining provenance. In: Althoff, K.D., Bergmann, R., Minor, M., Hanft, A. (eds.) Advances in CBR. pp. 269–283 (2008)
15. Madhusudan, T., Zhao, J.L., Marshall, B.: A case-based reasoning framework for workflow model management. Data & Knowledge Engineering 50(1), 87–115 (2004)
16. Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. In: Weber, R., Richter, M.M. (eds.) CBR Research and Development. p. 224–238 (2007)
17. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards Case-Based adaptation of workflows. In: Bichindaritz, I., Montani, S. (eds.) CBR Research and Development. LNAI, vol. 6176, pp. 421–435. Springer (2010)
18. Taylor, I.J., Deelman, E., Gannon, D.B.: Workflows for e-Science. Springer (2007)
19. Weber, B., Wild, W., Breu, R.: CBRFlow: enabling adaptive workflow management through conversational Case-Based reasoning. In: Funk, P., Gonzlez-Calero, P.A. (eds.) Advances in CBR. pp. 434–448. Springer (2004)