An Approach towards Automatic Workflow Composition through Information Retrieval

David Chiu
School of Engineering and Computer Science
Washington State University
Vancouver, WA 98686
david.chiu@wsu.edu

Travis Hall
School of Engineering and Computer Science
Washington State University
Vancouver, WA 98686
trhall@wsu.edu

Farhana Kabir
School of Engineering and Computer Science
Washington State University
Vancouver, WA 98686
farhana.kabir@wsu.edu

Gagan Agrawal
Department of Computer Science and Engineering
Ohio State University
Columbus, OH 43210
agrawal@cse.ohio-state.edu

ABSTRACT

Understanding how to design, manage, and execute scientific workflows has become increasingly esoteric. Yet, despite the development of scientific workflow management systems, which have simplified workflow planning to some extent, a means to reduce the complexity of user interaction without forfeiting some robustness has been elusive. We believe that a keyword interface may be highly beneficial to common users in need of information which requires workflow planning and execution. In this paper, we describe a system that can automatically compose a set of relevant workflows, which may or may not have been previously defined by other users, given only a keyword query. We present a way to index data sets and Web services (utilized to compose workflows in our system) on their ontological attributes. This ontology allows us to facilitate an IR-based workflow retrieval model. We conducted a case study in geoinformatics with a set of real geospatial Web services, data, and their metadata annotations. our system was capable of answering six keyword queries with fast search times (2.16ms on average) and relatively high Top-N precision values: 78%, 77.3%, and 76.2% for the Top 3, 5, and 10 retrieved workflows respectively.

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: General

Keywords

Scientific workflows, service composition

IDEAS11 2011, September 21-23, Lisbon [Portugal]
Editors: Bernardino, Cruz, Desai

1. INTRODUCTION

Modern search engines have become indispensable for locating relevant information about anything that has been published to the Web. Yet, at the same time, we have also grown aware of a search engine's limitations. In various scientific fields it is often difficult to reach information obscured deep within these domains with common search engines. For example, consider an earth science student who needs to identify how much a certain area in a nearby park has eroded since 1940 for a school project. Certainly, if this exact erosion information had previously been published onto a Web page, a search engine could probably locate it quite effortlessly. But due to the exactness of this query, the likelihood that such a Web page exists is slim. Consequently, the student would have to be content with approximate or anecdotal results. This compromise betrays the spirit of access to scientific information and the Web, and we argue that various avenues for obtaining this information probably do exist in the Web, but are simply unknown to the student.

While community question-answering (CQA) systems (e.g., *Yahoo! Answers*[1]) may offer some help, it is well known that user replies often lack any guarantee on relevance, reliability, or response time. The scientific nature of the student's query further restricts the number of users capable of participating in answering. Instead, we believe that the information the student seeks might be one that can be automatically derived from executing a piecemeal composition of available Web services, also referred to as *scientific workflow composition* [24, 14, 26]. For example, one way to derive the erosion data is to first acquire land elevation data sets from 1940 and the present at the desired location. The student may then locate, or provide, a service that can compute the difference between the two sets of data. Another way might be to find a mathematical erosion model that had been deployed as a Web service by a third party, and simply supply the required inputs (dates and location) to generate the estimated result.

We believe that the service-oriented solution is, after all, apropos—the maturation of the semantic Web has prompted,

[1] http://answers.yahoo.com

if not already produced, the mass sharing of scientific Web services and data sets. For instance, the *GeoPortal*[2] is one such site where users can publish and search geographic data sets and services. Elsewhere, *myExperiment* [16] is an online community that enables users to share, find, customize, and execute pre-made scientific workflow plans. However, workflow planning (Web service composition) still requires intrinsic knowledge from end-users in terms of which data sets and services to use, their interdependencies, and domain-level expertise.

In this paper, we describe our system, *Auspice*, which is capable of on-demand workflow planning given a set of keywords. Services and data sets are indexed ontologically on their tagged keywords and metadata. Given the index and a search query, Auspice not only returns previously defined workflows, but also identifies and composes any known services together with appropriate data sets to derive the information sought by the query. This strategy is in contrast to the aforementioned efforts (which unilaterally search predeveloped workflow plans, services, and data sets). Some aspects of our system are inspired by current methods in information retrieval integration [21, 17, 28]. We propose a workflow relevance model that ranks the retrieved composed workflow plans. To the best of our knowledge, this is first system to combine IR techniques with automatic service composition to provide access to scientific information.

Returning back to our example, we propose a system where the earth science student can input a simple query, e.g., `erosion (41.48335,-82.687778) 1940 2011`, and retrieve a set of workflow plans (which may or may *not* have been defined in the past) that may be relevant. Certainly, in this case, returning a workflow plan producing land elevation data sets during the queried times and location would be highly desirable. However, other plans, such as a workflow that produces a current image of the queried area may also be interesting to the user. Given the amount of shared services and data sets that have been made available by the scientific community, our system composes known services together with appropriate data sets to derive information sought by the query. Each composed workflow plan is ranked by a relevance score, and our student can then peruse each plan and its associated metadata, and select a workflow for execution. Auspice would then carry out the execution and deliver the resulting erosion data back to the student.

1.1 Contributions

We have developed keyword querying support for generating service-based workflow plans on the fly. These plans may or may not be previously developed by other users, and we have created an IR-based retrieval model which attempts to capture each workflow's relevance to the submitted keywords. Particularly, the workflows are ranked according to a score computed as a function of the number of scientific concepts relative to the query. An evaluation over a geoinformatics case study has been conducted using the IR metrics. Our collaborators in the geodetic sciences performed a blind relevance feedback on the generated workflows, and we are reporting high precision values: 78%, 77.3%, and 76.2% for the Top 3, 5, and 10 retrieved workflows respectively.

The ensuing sections are organized as follows. A metadata framework for supporting automatic workflow planning, i.e., service composition, is discussed next. In Section 3, we describe our retrieval model and propose algorithms for enabling keyword search under this framework. A system evaluation under a geoinformatics case study is further discussed in Section 4. Related efforts on workflow composition and keyword search systems are discussed in Section 5, and we conclude in Section 6.

[2]http://www.geowebportal.org

2. BACKGROUND AND DEFINITIONS

In this section, we describe our system's architecture, including the scheme for supporting data-driven automatic workflow composition.

Before exploring the details of our system, we first state some important assumptions. First, we note that workflow planning is a well-studied field (see Section 5), where many efforts have been made to ensure correctness. For this reason, we assume that support such as data type, schema matching, and program-level semantics (e.g., pre- and post- conditions) are in-place. The emphasis of this paper is placed directly on enabling a keyword interface to dynamically compose relevant workflows. We further assume that Web services and scientific data sets are accompanied by metadata. Available Web services are *tagged* with descriptive keywords, and data are annotated with general scientific metadata. Indeed, certain guidelines have already been accepted on the coupling of scientific metadata, including geodetic [15] and biological [23] sciences.

An overview of our system is depicted in Figure 1. Auspice maps a set of keywords to concepts within a scientific domain ontology. Once the set of concepts has been identified, it is sent to the workflow planner, which composes services together with data files automatically and returns a ranked list of workflow candidates to the user. The user can then select a suitable workflow plan to execute over the service-enabled Cloud or Grid frameworks. Because the ontology drives our planning algorithm, an important discussion involves how it is built. We allow users to share data sets and Web services with the system through a *registration* process. These registered resources are indexed by our ontology so that they are available for generating future workflow plans. Next, we first define the ontology, then describe the ontology registration process.

2.1 Capturing Concept Relationships

A necessary ingredient toward facilitating automatic workflow planning is scientific domain knowledge. A system must understand the semantic relationships among scientific data sets, Web services, and the information they can derive. In our system, scientific services and data sets are assumed to represent (or derive) certain virtual objects we simply refer to as *concepts*. For example, a *land elevation* concept in a geographic domain ontology may be *derived by* either a digital elevation file or Web services that produce such a file. In a more complicated example, to derive a *coast line*, resources associated with both *land elevation* and *water level* concepts may be required, and both recursively require further concepts toward their own derivation.

The concept derivation chain is analogous to service-based workflow planning. Given a targeted domain concept, c, it may be derived using a data set, d, or a Web service operation s, whose input parameters, $(x_1, \ldots, x_p)$ may again be substantiable by respective scientific concepts $(c(x_1), \ldots, c(x_p))$. Like before, each of these concepts may be further derived by another service or data set. This chaining process contin-

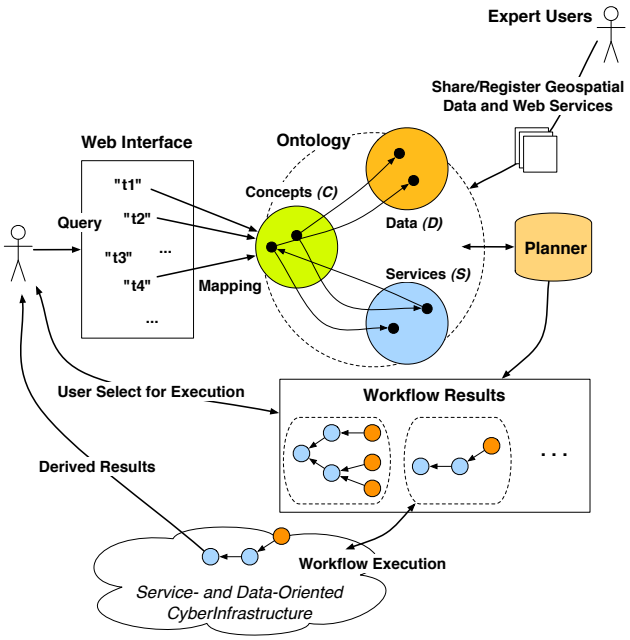

Figure 1: Auspice Interface and Concept

ues until a terminal element (either a service without input or a data file) has been reached on all paths. One can envision then, that when given a target concept along with attributes like date, location, etc., that applying this technique can yield a set of workflow plans simply by considering all derivation paths from the originating concept. In general, a workflow plan, w, can thus be recursively defined,

$$w = \begin{cases} \epsilon \\ d \in D \\ (s, (w'_1, \ldots, w'_p)) \in S \end{cases}$$

such that terminals ϵ and $d \in D$ respectively denote a null workflow and a data instance (file, user input, intermediate data, etc.) belonging to a specific data type in D. In the latter case, the nonterminal, $(s, (w'_1, \ldots, w'_p)) \in S$, is a tuple where s denotes a service operation with a corresponding parameter list, $(w'_1, \ldots, w'_p)$, and each w'_i is itself a workflow planned toward its own derivation. In other words, a workflow is a tuple that either contains a single data instance or a service operation whose parameters are, recursively, (sub)workflows.

In our previous effort, we proposed an ontology to capture these concept derivation relationships [6]. Ontology $O = (V_O, E_O)$ is a directed acyclic graph where a set of vertices, V_O, comprises a disjoint set of classes: scientific concepts C, service operations S, and data sets, D, i.e., $V_O = (C \cup S \cup D)$. Each directed edge $(u, v) \in E_O$ must represent one of the following relations:

$(u\ \delta^{s \to c}\ v)$ – Service $u \in S$ derives concept $v \in C$

$(u\ \delta^{d \to c}\ v)$ – Data type $u \in D$ derives concept $v \in C$

$(u\ \delta^{c \to s}\ v)$ – Concept $u \in C$ derives service $v \in S$

For convenience, we also define the following functions: $f_S(v) = \{u | (u\ \delta^{s \to c}\ v) \in E_O\}$ and $f_D(v) = \{u | (u\ \delta^{d \to c}\ v) \in E_O\}$ are the corresponding sets of services and data sets directly responsible for deriving concept c. We also define $f_C(v) = \{u | (u\ \delta^{c \to s}\ v) \in E_O\}$ to be the set of concepts which are responsible for deriving service v, i.e., the concept mappings of each of v's parameters.

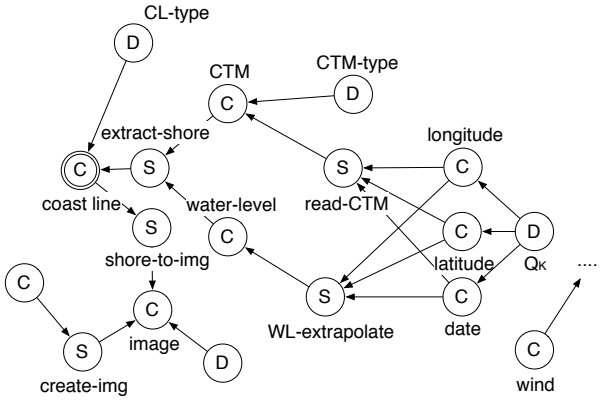

Figure 2: Example Ontology

Consider the example ontology illustrated in Figure 2. If *coast_line* is the targeted concept in the query, one could first traverse its edges in reverse, which is equivalent to identifying $f_S(coast\_line)$ and $f_D(coast\_line)$. Focusing on $extract\_shore \in f_S(coast\_line)$, we know that one set of workflows ends with executing *extract_shore*. We further identify *extract_shore*'s parameter concepts, $f_C(extract\_shore)$, and apply the traversal recursively. Each concept vertex which has more than one derivation edge results in a variation in the workflow, that is, a different workflow plan. The following is a list of workflow plans that can be enumerated to derive *coast line* concept:

$$w_1 = CL\text{-}type$$
$$w_2 = extract\_shore(CL\text{-}type, WL\text{-}extrapolate(Q[long], Q[lat], Q[time]))$$
$$w_3 = extract\_shore(read\text{-}CTM(Q[long], Q[lat], Q[time]), WL\text{-}extrapolate(Q[long], Q[lat], Q[time]))$$

where $Q[\ldots]$ are values given by the user query. Recall from the workflow definition that data type nodes, such as $CL\text{-}type$, in workflows are terminals. The product generated from w_1 for instance, would require that all known $CL\text{-}type$ data files be retrieved given $Q[\ldots]$ as attributes. We refer the interested reader to our previous work for more on this process [6].

We will revisit this workflow enumeration/generation process in further detail in Section 3 as we discuss the query planning algorithm, but first, we describe the process in which Auspice aids users in populating the ontology.

2.2 Facilitating Ontology Construction

To populate the ontology with available services and data sets, Auspice supplies a metadata registration framework. A brief description of data set registration is given here, but we refer the interested reader to our previous works [6] for a more nuanced discussion. Before describing the registration

process, it is important to note that each concept, $c \in C$, is assumed to be associated with a set of descriptive terms.

We first present the process for making data sets available for workflow planning. To register a data set with the ontology, a user inputs the file, its metadata, and optionally, a set of keywords that describe the data. Again, we assume that the accompanying metadata follows some well-defined standard, such as CSDGM [15] or NBII [23]. From the metadata, a set of predefined attributes are extracted and used for indexing. This is an important step, as it facilitates fast file identification (such as the file's capture time, dates, location properties, etc.). Indeed, the selection of indexable attributes are domain-dependent, and are themselves concepts in the ontology.

After the file has been indexed on its metadata, the ontology must be updated. The initial step in this process is to deduce the concepts which can be derived by the data set. A set of descriptive key terms are either input by the user or mined from the file's metadata descriptions (e.g., those with high term frequency) are first returned. Then given these terms, Auspice retrieves a set of concepts from the ontology that have a high likelihood of matching the given keywords. The user selects the concepts from this list which best summarize the new data set. The user also has the option create a new concept altogether. For each of the matching concepts, a $\delta^{d \to c}$ edge is added into the ontology.

A similar approach is taken for registering Web services into Auspice. An expert user initially inputs the service's description file given in the WSDL (Web Service Description Language) standard [9]. This document contains metadata descriptors of the service, which includes the operations' interface (input and return types, high level description). For each service operation given in the document, our system asks for a set of descriptive tags for each parameter, as well as keywords describing the service operation's output. Like before, these keywords should help describe scientific concepts.

Upon receiving these inputs, Auspice registers each prescribed service operation in the following way. First, the system's ontology must be updated to reflect the new service operation. A new service operation node, v, is created added into the service class, S. Tantamount to the data registration process, the edges for the new service are computed next. The keywords associated with the service's output are queried against the keywords annotating concepts. Again, the users select the concepts which best describe the service's output, and $\delta^{s \to c}$ edges are added into the graph. Next, for each service parameter, $\delta^{c \to s}$ edges are computed a similar fashion. With these ontology updates in place, the new Web service operation is now available to the workflow planner. Next, we describe the planning algorithm, as well as the keyword search approach to this algorithm.

3. QUERY PROCESSING AND WORKFLOW RETRIEVAL MODEL

We propose to automatically compose and return all workflows relevant to the most number of keywords in a user query, Q. We describe our algorithms using the ontology defined earlier in Figure 2, and further we interweave following keyword query in our description: "`wind coast line CTM image (41.48335, -82.687778) 8/20/2009`"

3.1 Planning with Keywords

The data and service metadata registration procedure, discussed in the previous section, allowed users to tag these resources with descriptive terms. Upon receiving a query, K, these supplied keywords are used to identify the concepts in which the new resource derives, and if such a concept does not exist, the user is given an option to create one in the ontology. Thus, we can assume that each concept, c, associates a set of keywords. For instance, the concept of *coast* might associate with terms $\{shore, shoreline, coastline, coast, \ldots\}$.

Some terms can be simply matched by their patterns. For example, 13:00 should be associated with the concept, *time*. Others require further processing. A given coordinate, $(41.48335, -82.687778)$, is parsable assigned to corresponding concepts independently, i.e., $(41.48335\ \delta^{d \to c}\ long)$ and $(-82.687778\ \delta^{d \to c}\ lat)$. Because Auspice is currently implemented over the geospatial domain, only a restricted set of such patterns can be expected. Users can also specify value assignments directly in their queries, using a *keyword=value* pattern.

In much the same way as in our metadata registration scheme, concepts are mapped onto the given keywords to generate two sets: C_K and Q_K, where C_K is the set of *unsubstantiated* concepts, and Q_K is the set of *value-substantiated* concepts. Specifically, Q_K is the set of concepts that have been assigned a value from the query. Therefore, the system assumes that Q_K is set of query parameters, where C_K contains the targeted concept element. The reason is that users are unlikely to supply a value to a concept for which they are querying. In our running example query, we may expect the following query decomposition:

$$C_K = \{wind, coast, line, CTM, image\}$$
$$Q_K = \{long \leftarrow 41.48335,\ \ lat \leftarrow -82.687778,\ \ date \leftarrow 8/20/2009\}$$

It is important to reiterate the assumption that the user's query target must be an element, or involve a combination, of C_K, which is identified with the support of Q_K.

These two sets, along with the ontology, are input directly into the query processor that comprises two main methods: *Query Execution* (`QueryExec`), which invokes *Workflow Enumeration* (`WFEnum`), as a subprocedure. The `WFEnum` procedure composes and retrieves all workflows that derive some user-targeted concept, $c_t \subseteq C_K$, under the support of the given attributes, Q_K. The `QueryExec` procedure generates the power set, $\mathcal{P}(C_K)$, which contains all concept combinations, and invoking `WFEnum` repeatedly using the various $c_t \in \mathcal{P}(C_K)$. Opportunities for pruning $\mathcal{P}(C_K)$ elements are abundant, but we first describe the `WFEnum` method.

3.1.1 Automatic Workflow Planning

We can now lead into the discussion of the workflow planning algorithm, *Workflow Enumeration*, which is shown in Algorithm 1. `WFEnum`'s inputs include $c_t \subseteq C_K$, which denotes the targeted concept (or set of concepts). The next input, Π, is a set of required concepts. Every concept element in Π must appear somewhere in any derived workflow. A set of query parameters, e.g., user given attributes, Q_K, is also given to this algorithm. These would include the value-substantiated concepts, such as the longitude, latitude, and the date given by the user in our example query. Q_K is used to identify the correct data files and also as input into services that require these particular concept values. Finally,

the ontology, O, is given to supply the algorithm with the derivation graph. The algorithm exhaustively retrieves and returns a set of workflow plans, R.

Algorithm 1 WFEnum(c_t, Π, Q_K, O)

```
1: static R
2: ▷ Traverse all data nodes responsible for deriving c_t
3: for all d ∈ f_D(c_t) do
4:    R ← R ∪ {σ_<Q_K>(d)} ▷ Select files using the supporting query parameters, Q_K.
5: end for
6: ▷ Traverse all service nodes responsible for deriving c_t
7: ▷ Any workflow enumerated must be reachable within Π
8: for all s ∈ f_S(c_t) do
9:    cand ← true; R_s ← ()
10:   ▷ Remove target, c_t, from requirement set
11:   Π ← Π \ {c_t}
12:   ▷ Get all concepts, c', that are mapped to s's parameters
13:   for all c' ∈ f_C(s) do
14:      if Π ⊆ ψ(c') then
15:         R_s ← WFEnum(c', Π ∩ ψ(c'), Q_K, O)
16:         R ← R ∪ R_s
17:      else
18:         cand ← false
19:         break ▷ Prune s
20:      end if
21:   end for
22:   if cand = true then
23:      ▷ construct service invocation plan for each p ∈ R_s, and append to R
24:      for all p ∈ R_s do
25:         R ← R ∪ {(s, p)}
26:      end for
27:   end if
28: end for
29: return R
```

On (Alg1:Lines 3-5), the algorithm first considers all data-type derivation possibilities within the ontology, e.g., ($d\ \delta^{d \to c}\ c_t$), by traversing all nodes $d \in F_D(c_t)$. On (Alg1:Line 4), data files are retrieved from the data store with respect to data type d and to the parameters given in Q_K, i.e., each returned file record, f, becomes a file-based workflow deriving c_t. Next, the algorithm handles service-based derivations, e.g., ($s\ \delta^{s \to c}\ c_t$) edges. For each service operation, $s \in F_S(c_t)$, s's parameters must first be recursively planned, and the set, R_s (Alg1:Lines 15-16), is used to contain all workflows recursively generated to derive service s. Recall that before we can compose s, we must first derive all of s's parameters, which by prior definition, are sub-workflows. Only afterwards can s be ultimately be composed with each sub-workflow from R_s. To perform recursive derivation, c_t, the current targeted concept, is first removed from Π (Alg1:Line 11). Next, traverse through each of s's parameter concepts $c' \in F_C(s)$ (Alg1:Line 13) and generate its workflows by recursively calling `WFEnum` on c' (Alg1:Line 15).

Opportunities for pruning are revealed at this stage of the algorithm. To support pruning, we define the notion of *concept-reachability*. We say that c_i is reachable by c_j if and only if there exists a path in ontology O from c_j to c_i. We let the set, $\psi(c_i)$, denote all such concept-reachable nodes to c_i:

$$\psi(c_i) = \{c_j | \exists \text{ path } (c_j \rightsquigarrow c_i) \text{ in } O\}$$

The pruning logic follows that, if the set of required concepts, Π, does not overlap in the derivation paths of c', then c' is underivable. Because c' is a parameter of service, s, it would also imply that s is underivable. Thus, any workflow involving s can be pruned from the retrieved set (Alg1:Line 18-19). Conversely, if s is promising, and its parameters' concepts are used as targets to recursively generate workflow plans toward the ultimate realization of s (Alg1:Line 15). Recalling the workflow's recursive definition from Section 2, this step is tantamount to deriving the nonterminal case where $(s, (w'_1, \ldots, w'_p)) \in S$. The generated sub-plans for each of s's parameters are stored in R_s and can be added to the result set, R (Alg1:Line 16). The final pairing of s with all of its parameters' sub-plans in R_s is completed (Alg1:Lines 24-26). Each pair represents a disparate workflow deriving c_t, and finally, R is returned as the results (Alg1:Line 29).

3.1.2 Keyword Querying

The query execution method (Algorithm 2) finds promising input configurations to invoke the previous algorithm, `WFEnum`. This algorithm assumes that the user query has already been parsed into the set of value-instantiated concepts, Q_K, and C_K, set of unsubstantiated concepts.

Algorithm 2 QueryExec(C_K, Q_K)

```
1: R ← ()
2: ▷ compute the power set, P(C_K), of C_K
3: for all ρ ∈ P(C_K), in descending order of |ρ| do
4:    ▷ ρ = {c_1, ..., c_n}, {c_1, ..., c_{n-1}}, ..., {c_1}
5:    ▷ check for ψ-reachability within ρ, and find successor
6:    for all c_i ∈ ρ do
7:       if (ρ − {c_i}) ⊆ ψ(c_i) then
8:          ▷ enumerate all plans with c_t as target
9:          c_t ← c_i
10:         Π ← (ρ − {c_t})
11:         R ← R ∪ WFEnum(c_t, Π, Q_K, O)
12:         ▷ prune all subsumed elements from P(C_K)
13:         for all ρ' ∈ P(C_K) do
14:            if ρ' ⊆ ρ then
15:               P(C_K) ← P(C_K) − {ρ'}
16:            end if
17:         end for
18:         break
19:      end if
20:   end for
21: end for
22: return R
```

Recall from earlier discussion that we assume the user's desired query target is an element of, or a subset of unsubstantiated elements in C_K. The power set, $\mathcal{P}(C_K)$ is first computed for C_K. If we let ρ denote a single element in $\mathcal{P}(C_K)$, then input configuration into `WFEnum` is the tuple $(c_t, \{\rho - \{c_t\}\}, Q_K, O)$.

However, not every $c_t \in \rho$ is justifiable as the desired target concept. If a target, c_t, cannot be reached by any of the required concepts in $\{\rho - \{c_t\}\}$, then it can be pruned as a possible input configuration (Alg2:Line 7). In the example ontology in Figure 2, when $\rho = \{wind, coast, line,$

image, *CTM*}, then c_t = *image* and Π = {*wind*, *coast*, *line*, *CTM*} is the only acceptable configuration due to $(\rho - \{image\}) \subseteq \psi(image)$. When a reachable configuration has been determined, the planning method, `WFEnum` can be invoked inputting the corresponding c_t and $\{\rho - \{c_t\}\}$ as input to the required set, Π (Alg2:Line 11).

After (possibly) repeated invocations of `WFEnum`, the workflow plans are ranked and returned. Next, we discuss the scoring function by which workflows are ranked.

3.2 Workflow Relevance

The retrieved set of workflow plans, R, should be ranked according to their *relevance* to the query. We define relevance in our context as a function on the number of keyword-mapped concepts that appear in each workflow plan, w. We, for instance, would expect that any workflow rooted in *wind* be less relevant to the user than the plans which include significantly more keyword-concepts: *coast*, *image*, etc., if the query in the running example was given to the system. Given a workflow plan, w, and set of query terms, $K = (k_1, \ldots, k_n)$, we first define $f_C(w)$ to be the set of concepts corresponding with each service and data file in the workflow, w. Formally,

$$f_C(w) = \bigcup_{s,d \in w} \{c_i | (s\ \delta^{s \to c}\ c_i)\} \cup \{c_i | (d\ \delta^{d \to c}\ c_i)\}$$

Next, because w may contain many concepts that may or may not be relevant to the user, we want to assess some form of an *overfit penalty*. The penalty, $\delta(w, K)$, is computed from the concepts in w that are superfluous to the user query. In our case studies, we have used the following,

$$\delta(w, K) = \log(|f_C(w) - C_K| + 1)$$

Recalling that C_K denotes the unsubstantiated query concepts, $|f_C(w) - C_K|$ simply denotes the number of superfluous concepts in w. We can now define w's scoring function as follows,

$$rank(K, w) = \frac{\sum_{k_i \in K} |\{k_i \in f_C(w)\}|}{((1-\beta) + \beta\delta(w, K)) \times |f_C(w)|}$$

where $|\{k_i \in f_C(w)\}|$ is the total number of times a query term k_i occurs in w's derivation graph and $|f_C(w)|$ is the number of concepts pertaining to w. β is a tunable parameter that controls the impact of the overfit penalty. When $\beta = 0$, overfitting is ignored, and when $\beta = 1$, the full penalty of δ is assessed in the score. Our ranking function is a variation of typical probabilistic term-frequency models in IR, and inspired especially by BM25 proposed by Jones, *et al.* [20].

4. A CASE STUDY IN GEOINFORMATICS

We present a case study of our keyword search functionality in this section. Due to the subjective nature on workflow relevance feedback, we worked directly with the Department of Civil and Environmental Engineering and Geodetic Sciences at Ohio State University. Our collaborators provided us with the following resources:

1. A set of 25 Web services they had previously developed to process coastal data.
2. 2248 data files (and 2248 accompanying metadata files given in CSGDM [15]) representing various remote sensing information in the Lake Erie region, including satellite images, wind speed, water levels, temperature, etc.
3. A set of keyword queries (shown in the Table below) that may be representative and useful to geodetic scientists.

Together, we extracted 29 geodetic concepts representing the outputs of the given services and data sets. The services and data files were then registered with the ontology based on their accompanying metadata. While we concede that the set of resources seems restricted, we believe it justifiable. Given that we are ultimately interested in our system's *effectiveness*, we must evaluate our system on its relevance feedback. Thus, we believe that a focused case study using a familiar set of services and data would give our scientists a baseline for evaluating our system.

Query ID	
1	"`coast line CTM 7/8/2003 (41.48335,-82.687778)`"
2	"`bluff line DEM 7/8/2003 (41.48335,-82.687778)`"
3	"`(41.48335,-82.687778) 7/8/2003 wind speed`"
4	"`waterlevel=174.7cm water surface 7/8/2003 (41.48335,-82.687778)`"
5	"`waterlevel (41.48335,-82.687778) 13:00:00 3/3/2009`"
6	"`land surface change 7/8/2003 (41.48335,-82.687778) 7/7/2004`"

4.1 Evaluating Effectiveness

In traditional information retrieval, given a keyword query K, the retrieved set R, and a set of relevant results known a priori, REL, the recall and precision of R are defined $recall(R, K) = |REL \cap R|/|REL|$ and $prec(R, K) = |REL \cap R|/|R|$. One known issue with this classic definition is that it assumes a binary model—a result is either relevant or not. Because our retrieval model also outputs any partial workflows that may match a subset of keywords, we do not believe this form of evaluation is robust. Given a workflow plan, w, and keywords, K, we asked our collaborators to assign a relevance-feedback, $fb(w, K) \in [0, 1]$ (1 being most relevant) to each retrieved plan[3]. We then compute a Top-N normalized precision as follows,

$$prec(R, K, N) = \frac{1}{N} \sum_{i=0}^{N-1} fb(w_i \in R, K)$$

which evaluates our retrieval model for the first N workflows displayed.

In Figure 3, we show the Top-N and Mean precision values for each of the queries. The `Top-3`, `Top-5`, `Top-10`, and

[3]Our collaborators were asked to do this blindly, i.e., without knowledge of the ranking order our system had output.

`Mean` (i.e., $N = |R|$) precisions are plotted. The high precision rates for the `top-k` modes are encouraging signs for our retrieval algorithm due to the fact that users are typically interested in the first few results. The figure shows high precision for `Top-3` across all six queries, and then expectedly drop as N increases to 5 and 10, with the exception of Query 6, which found better results towards the middle of the returned list. Overall, across the six queries, we observe an average precision of 78%, 77.3%, and 76.2% for Top 3, 5, and 10, respectively. These results are promising, as it suggests that our ranking model is effective.

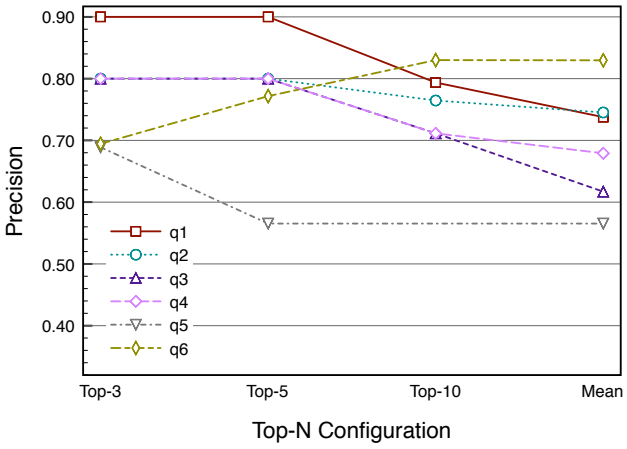

Figure 3: Top-N Precision

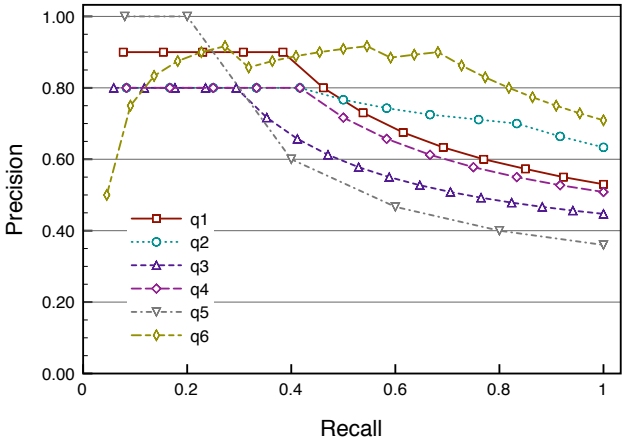

Figure 4: Recall-Precision Graph

Figure 4 depicts the Recall-Precision graph for the same six queries, a more nuanced, perspective than the Top-N plots. We can observe the behavior of precision as increasingly more plans are retrieved (recall). To plot this graph, we assume that our system only composes and retrieves relevant workflows, i.e., $R = REL$. Because of this assumption, our precision typically decrease, as user feedback becomes lower for lower ranked results. This is again, with the slight exception for Query 6, but now we are afforded an orthogonal view that it is only a mis-ranking of the first workflow plan (albeit the *most important ranking*) that is causing the poor results early. After around 60% of all plans have been retrieved for this query, the remaining become quite irrelevant for the user, which is our expectation.

4.2 Query Latency

Here we present the search time of the six queries issued to the system. In this experiment, we executed the search using two versions of our querying algorithms. The search time is the latency from the time a keyword query was submitted to displaying the results. Most of the overhead is attributed to planning time.

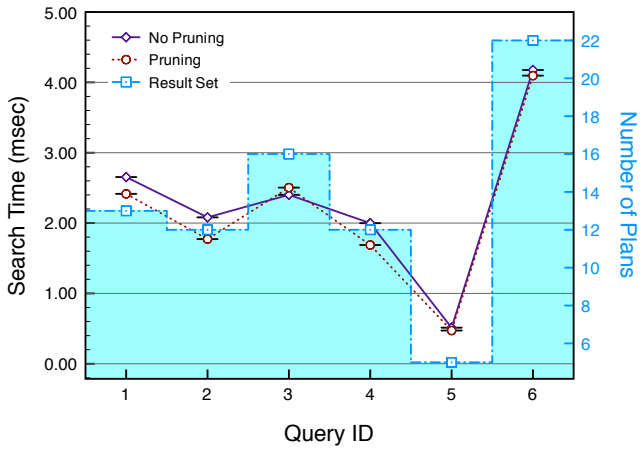

Figure 5: Query Latency

The first version consists of the pruning logic that was described in the previous section, and the second version does not perform pruning of unlikely workflow plans until the end. The results of this experiment are shown in Figure 5, and as we can see, a typical search executes on the order of several milliseconds. Along the right-hand y-axis, the result size (number of workflow plans generated) is shown. We can also see that the pruning version results in slightly faster search times in almost all queries. These pruning results are promising for such a small testbed. In a much larger ontology and data/service set, we would expect much higher benefits via pruning. The exception of Query 3, however, was at first vexing, but it was later determined that this query does not in fact benefit from pruning with the given services and data sets in the ontology, and the pruning logic became an overhead.

5. RELATED WORKS

There has been an abundant amount of work on scientific workflow systems. However, to the best of our knowledge, we believe that our system is the first to offer an IR-based keyword querying interface for automatic composition of scientific workflows. In this section, we initially discuss some pioneering works on scientific workflow management systems. We follow this with related efforts on IR over relational databases, which are loosely related, but inspirational to this work. Finally, we present and contrast works on the role that Web 2.0 has had on the scientific workflow community.

Early workflow systems include ZOO [19] and Condor [10]. ZOO uses an object-oriented language to model the invocation of processes and the relationships between them. Condor was originally proposed to harvest the potential of idle CPU cycles. Soon after, dependent processes (in the form of directed acyclic graphs) were being scheduled on Condor systems using DAGMan. Pegasus [12] creates workflow plans in the form of Condor DAGMan files, which then uses the DAGMan scheduler for execution. Other systems allow users to guide workflow designs. To complement the growing need for interoperability and accessibility, many prominent workflow managers, including Pegasus [12], Taverna [24], Kepler [2] (the service-enabled successor to the actor and component-based Ptolemy II [5]), and Triana [22] have become attuned with service-oriented systems. These works require careful workflow design by users. For instance, either using a declarative file to describe the processing dependencies (as in Condor) or tools such as a GUI, a workflow can be constructed. Our system, in contrast, offers a keyword query interface and attempts to compose workflows automatically.

Search engines have ushered the keyword paradigm into the mainstream, as it requires no a priori knowledge about a system's underlying structure to construct queries. Typically, keyword search queries derive a list of results ranked against their relevance to the user's needs. While only very loosely related to our work, it is worthwhile to note the efforts made toward keyword search over relational databases. In DataSpot [11] and BANKS [3], the database is represented as a directed graph where nodes denote tuples and directed edges represent "links" from foreign keys to primary keys across tables. To answer a query in DataSpot, it returns the connected subgraphs where the nodes contain all keyword terms. Because it the information represented by these subgraphs are sometimes ambiguous, BANKS differs in that the results to a query are those directed trees (Steiner trees) whose nodes correspond to the given keywords. The trees are rooted on a node that connects to all other keyword nodes, and therefore it conveys the strongest information. The nodes in BANKS's structure employ a PageRank-style [4] weighting, which affects the ranking of results.

Still in the scope of keyword querying over databases, DISCOVER [18] and DBXplorer [1] generate relevant SQL on-the-fly and returns the tuple trees (from potentially joined tables) that contain all keywords. These efforts ranked the results based on the number of table joins needed to generate each result, i.e., the more joins needed, the fuzzier (and thus, less relevant) the results are. This is similar to the ranking model taken by our system. For instance, more service and data nodes in a composed workflow implies fuzzier results, and will be scored less. In other works, Hristidis et al. [17] and Liu et al. [21] later proposed IR-style ranking strategies which, among other things, leverages the underlying database's native relevance methods for text search within attributes. Saddyadian et al.'s Kite [27] extends these efforts to distributed, heterogeneous databases. In some systems [25, 29] the entire database is crawled and indexed offline. Particularly, Qu and Widom's EKSO [29] crawls the database offline and generates *virtual documents* from joining relational tuples. Each virtual document associates with some database object that should be understandable to the user when returned. These virtual documents are then indexed using traditional keyword/IR indexing techniques.

Search engines like *seekda!* [28] can identify usable Web services. The service and their operations are indexed via annotation in the WSDL descriptor. However, these tools cannot compose service plans. Dong, Halevy, et al., described Woogle, a similarity search engine for services [13], which can aid users in identifying relevant services for composition. Unsupervised labeling/clustering is applied to service descriptors to facilitate similarity searches. Woogle can handle *operation matching*, which inputs a service operation and returns a list of similar service operations and *input/output matching*, which returns a set of services with similar inputs and outputs. In contrast, our system plans services and data sets together to derive scientific information.

Wolfram|Alpha[4] is a Web-based expert system which generates impressively exact answers to highly specific queries. Massive volumes of *curated data* from trusted sources are organized and computed on-the-fly. myExperiment [16] is a forum inspired by social networking and Web 2.0 for users to communicate, share, and use bioinformatics workflows. To facilitate workflow discovery, workflows are tagged with descriptive keywords and attributed to certain scientists/groups. Keyword search can be used to identify relevant workflows in the myExperiment catalogue. However, our system does not require users to share well-defined workflow structures. Workflow plans are retrieved through automatic service and data composition.

6. CONCLUSION AND FUTURE WORK

We have described an approach toward supporting keyword search in our workflow system, *Auspice*. Available data sets and services are indexed on the concepts they represent, and an ontology is prescribed to represent derivation relationships. We proposed algorithms which return workflow plans ranked by order of their relevance to the user's keyword query. This relevance is a function on the number of keywords that a workflow's concept-representation can match. We were able to show that our method renders relatively high precision for several keyword queries. We believe that this is a promising initial step towards realizing our vision for a dynamic, scientific Web.

We wish to include the quality of workflow as a factor in relevance calculations. In our previous works [7, 8], we developed a framework for estimating a workflow's execution time costs and its accuracy as a way for supporting QoS. Extensions to this work are underway to improve our relevance metric by incorporating our cost models.

7. REFERENCES

[1] S. Agrawal. Dbxplorer: A system for keyword-based search over relational databases. In *In ICDE*, pages 5–16, 2002.

[2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows, 2004.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, pages 431–440, 2002.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[4]http://www.wolframalpha.com

[5] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng. Heterogeneous concurrent modeling and design in java (volume 2: Ptolemy ii software architecture). Technical Report 22, EECS Dept., UC Berkeley, July 2005.

[6] D. Chiu and G. Agrawal. Enabling ad hoc queries over low-level scientific datasets. In *Proceedings of the 21th International Conference on Scientific and Statistical Database Management (SSDBM'09)*, 2009.

[7] D. Chiu, S. Deshpande, G. Agrawal, and R. Li. Composing geoinformatics workflows with user preferences. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'08)*, New York, NY, USA, 2008.

[8] D. Chiu, S. Deshpande, G. Agrawal, and R. Li. Cost and accuracy sensitive dynamic workflow composition over grid environments. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (Grid'08)*, 2008.

[9] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1.

[10] Condor dagman, http://www.cs.wisc.edu/condor/dagman.

[11] S. Dar, G. Entin, S. Geva, and E. Palmon. DtlÕs dataspot: Database exploration using plain language. In *In Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases*, pages 645–649. Morgan Kaufmann, 1998.

[12] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. C. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[13] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *In Proc. of VLDB*, pages 372–383, 2004.

[14] S. Dustdar and W. Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30, 2005.

[15] Metadata ad hoc working group. content standard for digital geospatial metadata, 1998.

[16] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure. myexperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucl. Acids Res.*, pages gkq429+, May 2010.

[17] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[18] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.

[19] Y. E. Ioannidis, M. Livny, S. Gupta, and N. Ponnekanti. Zoo: A desktop experiment management environment. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 274–285, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[20] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. In *Information Processing and Management*, pages 779–840, 2000.

[21] F. Liu, C. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2006. ACM.

[22] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 514–524. IEEE Computer Society, 2004.

[23] Biological data working group. biological data profile, 1999.

[24] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[25] P. Raghavan. Structured and unstructured search in enterprises. *IEEE Data Eng. Bull.*, 24(4):15–18, 2001.

[26] J. Rao and X. Su. A survey of automated web service composition methods. In *SWSWPC*, pages 43–54, 2004.

[27] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano. Efficient keyword search across heterogeneous relational databases. In *ICDE*, pages 346–355, 2007.

[28] seekda web services search engine, http://webservices.seekda.com/.

[29] Q. Su and J. Widom. Indexing relational database content offline for efficient keyword-based search. In *IDEAS '05: Proceedings of the 9th International Database Engineering & Application Symposium*, pages 297–306, Washington, DC, USA, 2005. IEEE Computer Society.