Workflow Unified as a single Matrix for Processing Unlimited Services (WUMPUS)

Mac Radigan

Abstract

...

Contents

List of Algorithms

1 Introduction

2 Workflow Unified as a single Matrix for Processing Unlimited Services (WUMPUS)

`# workflow productions`

`# raw data services`

$\{d_{y_1 1}, d_{y_2 1}, \cdots, d_{y_Q 1}\} = \rho_1\,()$ # inputs to raw services are implicit (source)

$\{d_{y_1 2}, d_{y_2 2}, \cdots, d_{y_Q 2}\} = \rho_2\,()$ # inputs to raw services are implicit (source)

$\vdots$

`# intermediate services`

$\{d_{y_1 k}, d_{y_2 k}, \cdots, d_{y_Q k}\} = \rho_k\,(\{d_{x_1 k}, d_{x_2 k}, \cdots, d_{x_P k}\})$

$\{d_{y_1 k+1}, d_{y_2 k+1}, \cdots, d_{y_Q k+1}\} = \rho_{k+1}\,(\{d_{x_1 k+1}, d_{x_2 k+1}, \cdots, d_{x_P k+1}\})$

$\vdots$

`# final service`

$\{d_{f_1 M}, d_{f_2 M}, \cdots, d_{f_Q M}\} = \rho_M\,(\{d_{x_1 M}, d_{x_2 M}, \cdots, d_{x_P M}\})$ # this is the prototype service (sink)

Thus, each production is a map $f\colon \mathbf{P} \times \mathbf{D}^P \mapsto \mathbf{D}^Q$. We may re-write the productions as follows:

$$
\begin{aligned}
\left(\{d_{y_1 1}, d_{y_2 1}, \cdots, d_{y_Q 1}\}\right) &\leftarrow (\rho_1, \{\epsilon\}) \\
\left(\{d_{y_1 2}, d_{y_2 2}, \cdots, d_{y_Q 2}\}\right) &\leftarrow (\rho_2, \{\epsilon\}) \\
&\vdots \\
\left(\{d_{y_1 k}, d_{y_2 k}, \cdots, d_{y_Q k}\}\right) &\leftarrow (\rho_k, \{d_{x_1 k}, d_{x_2 k}, \cdots, d_{x_P k}\}) \\
\left(\{d_{y_1 k+1}, d_{y_2 k+1}, \cdots, d_{y_Q k+1}\}\right) &\leftarrow (\rho_{k+1}, \{d_{x_1 k+1}, d_{x_2 k+1}, \cdots, d_{x_P k+1}\}) \\
&\vdots \\
\left(\{d_{f_1 M}, d_{f_2 M}, \cdots, d_{f_Q M}\}\right) &\leftarrow (\rho_M, \{d_{x_1 M}, d_{x_2 M}, \cdots, d_{x_P M}\})
\end{aligned}
\tag{1}
$$

From the workflow productions we can populate an oriented degree incident matrix, $\mathbb{W}$, having a row for each service, and a column for each Datatype connection between services. Entries at row i and column j are positive if the i^{th} service takes the j^{th} Datatype as input, and negative if it is an output. The magnitude of the entry is 1 times the degree of input/output files of the given type.

$\mathbb{W}$	d_0	d_1	$\cdots$	d_N
ρ_0	$\omega_{1,1}$	$\cdots$	$\cdots$	$\omega_{1,N}$
ρ_1	$\vdots$	$\omega_{m,n}$	$\iddots$	$\vdots$
$\vdots$	$\vdots$	$\iddots$	$\ddots$	$\vdots$
ρ_M	$\omega_{M,1}$	$\cdots$	$\cdots$	$\omega_{M,N}$

Table 1: Service Production Oriented Incidence Matrix

$$\omega_{m,n} = \begin{cases} -\alpha, & \text{if } x = \rho_m\left(f_n, \cdots\right) \exists x \in \mathbb{D} \text{ \{ i.e. the file } d_n \text{ is an input to } \rho_m \text{ with multiplicity } \alpha \text{ \}} \\ +\alpha, & \text{if } d_n = \rho_m\left(x, \cdots\right) \exists x \in \mathbb{D} \text{ \{ i.e. the file } d_n \text{ is an output of } \rho_m \text{ with multiplicity } \alpha \text{ \}} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Taking the *sign* of the matrix we have an ordinary oriented incident matrix, $\mathbb{W}_u$.

$$\mathbb{W}_u = signum\left(\mathbb{W}\right) \tag{3}$$

We will define the notation $\mathbb{M}\left(i|j\right)$ to indicate a submatrix of $\mathbb{M}$ having the i^{th} row and j^{th} column removed. We can then identify the potential starting services matrix, $\mathbb{S}$, as the rows of $\mathbb{W}_u$ those which do have any inputs, as shown below.

$$\mathbb{S} = \mathbb{W}_u\left(|j\right) \quad \text{where } j = \{x | x \in \mathbb{W}_u \wedge \min(\mathbb{W}_u) = 1\} \tag{4}$$

We will designate the final (prototype) service $\mathbb{F}$, and specify that it is known at the time of static analysis. Now identify the intermediate services $\mathbb{P}$ knowing that $Intermediate = All \setminus (Starting \cup Final)$.

$$\mathbb{P} = \mathbb{W}_u\left(|j\right) \quad \text{where } j = \{x | x \in \mathbb{W}_u \wedge \min(\mathbb{W}_u) = 1 \vee \max(\mathbb{W}_u) = -1\} \tag{5}$$

We have thus partitioned $\mathbb{W}_u$ into a starting submatrix, $\mathbb{S}$ intermediate submatrix, $\mathbb{P}$ final submatrix, $\mathbb{F}$.

$$\mathbb{W}_u = \left[\begin{array}{c} \mathbb{S} \\ \hline \mathbb{P} \\ \hline \mathbb{F} \end{array}\right] \tag{6}$$

The Laplacian matrix is the product of an oriented incident matrix, say $\mathbb{M}$, and its transpose. Note that

this matrix is Hermitian, which is a property we will exploit in the numerical computation of eigenvalues.

$$\mathbb{L} = \mathbb{M}\mathbb{M}^{\intercal} \tag{7}$$

We can determine if all nodes within a submatrix of an oriented incident matrix is connected from its algebraic connectivity. The second eigenvalue of the Laplacian matrix of a subgraph is the algebraic connectivity, and thus the subgraph is connected if the second eigenvalue is greater than zero.

$$\Delta_{\mathbb{L}}(s) = |s\mathbb{I} - \mathbb{L}| \tag{8}$$

$$\underline{\lambda} = \Delta_{\mathbb{L}}(0) \tag{9}$$

$$\lambda_2 > 0 \Leftrightarrow \text{connected} \tag{10}$$

Algorithm 1 Connected

function CONNECTED($\mathbb{M}$)
  {Create the Laplacian matrix}
  $\mathbb{L} = \mathbb{M} \times \mathbb{M}^{\intercal}$
  {Solve for the eigenvalues}
  $\underline{\lambda}$ =EIGW($\mathbb{L}$)
  if $\lambda_2 > 0$ **then**
    {The second eigenvalue is the algebraic connectivity}
    return True
  else
    return False
  end if
end function

We may exploit the fact that our Laplacian matrix is Hermitian, and use Kung's Algorithm for QR factorization (qrDecomp). We may then use the QR Algorithm (eigW) to compute the eigenvalues of the Laplacian matrix.

To support Kung's Algorithm, we introduce a notation for transvections, $T_{i,j}^{N}(x)$. The transvection is an $N \times N$ matrix with ones along the diaginal, the value x at row i and column j, and zeros elsewhere.

$$\mathbb{T}_{i,j}^{N}(x) = \mathbb{I}_N + x\mathbb{E}_{i,j} \tag{11}$$

Kung's QR Algorithm then computes a strictly positive diagonalization matrix, $\mathbb{E}$, and resultant diagonal matrix $\mathbb{D}$. We define $\mathbb{C} = \sqrt{\mathbb{D}}$, and can write the factorization of a matrix $\mathbb{M}_{N\times N}$ as follows.

$$\mathbb{E} = \prod_{i=1}^{N}\prod_{j\neq i} T_{i,j}^{N}\left(\mathbb{M}_{i,j}\right)^{*} \tag{12}$$

$$\mathbb{D} = \left(\mathbb{E}\mathbb{M}\right)^{*}\left(\mathbb{M}\mathbb{E}\right) = \mathbb{E}^{*}\mathbb{M}^{*}\mathbb{M}\mathbb{E} \tag{13}$$

$$\mathbb{C} = \sqrt{\mathbb{D}} \tag{14}$$

$$\mathbb{Q} = \mathbb{A}\mathbb{E}\mathbb{C}^{-1} \tag{15}$$

$$\mathbb{R} = \mathbb{C}\mathbb{E}^{-1} \tag{16}$$

Algorithm 2 Transvection

function TRANSVECTION($\mathbb{M}_{N\times N}, i, j$)
  {Initialize an identity matrix}
  allocate $\mathbb{T}_{N\times N} \leftarrow 0$
  for $k \leftarrow 1 \ldots N$ **do**
    $\mathbb{T}_{k,k} \leftarrow 1$
  end for
  {Conjugate and copy at i,j}
  $\mathbb{T}_{i,j} \leftarrow \mathbb{M}_{i,j}^{*}$
  {Return the transvection}
  return $\mathbb{T}$
end function

The QR Algorithm iteratively computes the eigenvalues of a matrix using a QR decomposition. The algorithm converges when the elements in the subdiagonal approach zero.

Algorithm 3 Kung's Algorithm

function QRDECOMP($\mathbb{M}_{N\ timesN}$)
  {Initialize an identity matrix}
  allocate $\mathbb{E}_{N\times N} \leftarrow 0$
  for $k \leftarrow 1 \dots N$ **do**
    $\mathbb{E}_{k,k} \leftarrow 1$
  end for
  {Create the diagonalizer by applying conjugate pairs of transvections to remove off-diagonal elements}
  for $i \leftarrow 1 \dots N$ **do**
    for $j \leftarrow 1 \dots N$ **do**
      if $i \neq j$ **then**
        $\mathbb{E} = \mathbb{E} \cdot$ TRANSVECTION$(\mathbb{M}, i, j)$
      end if
    end for
  end for
  {Apply the diagonalizer to create the diagonalization}
  $\mathbb{D} \leftarrow \mathbb{E}^*\mathbb{M}^*\mathbb{M}\mathbb{E}$
  $\mathbb{C} \leftarrow \sqrt{\mathbb{D}}$
  {Compute the $\mathbb{Q}$ and $\mathbb{R}$ matrices}
  $\mathbb{Q} \leftarrow \mathbb{A}\mathbb{E}\mathbb{C}^{-1}$
  $\mathbb{R} \leftarrow \mathbb{C}\mathbb{E}^{-1}$
  {Return the orthogonal and upper triangular decomposition}
  return $\mathbb{Q}, \mathbb{R}$
end function

Algorithm 4 QR Algorithm

function EIGW($\mathbb{A}_0^{N\times N}$)
  {Set the initial conditions}
  $k \leftarrow 0$
  $\mathrm{Q}_0, \mathbb{R}_0 \leftarrow qrDecomp(\mathrm{A}_0)$
  {Iterate until the subdiagonal converges to zero}
  while $\mathbb{A}_{k_{i,j}} < \epsilon, \forall j = i-1 \forall i \in 1 \dots N$ **do**
    $k \leftarrow k + 1$
    $\mathbb{Q}_{k-1}, \mathbb{R}_{k-1} \leftarrow$ QRDECOMP$(\mathbb{A}_{k-1})$
    $\mathbb{A}_k \leftarrow \mathbb{R}_{k-1}\mathbb{Q}_{k-1}$
  end while
  {Create a vector from the diagonal elements of $\mathbb{A}_k$}
  $\underline{\lambda} \leftarrow \{a_{i,j} | \forall a_{i,j} \in \mathbb{A}_k \wedge i = j\}$
  {Return eigenvalues}
  return $\underline{\lambda}$
end function

Algorithm 5 On File Processed **CONCURRENT**

procedure onFileProcessed(fileset, parameters)

{Compute the state vector}

$\underline{\mathrm{Z}}_k \leftarrow$ stateVector$(fileset)$

{Compute the state change}

$\Delta\underline{\mathrm{Z}}_k \leftarrow \underline{\mathrm{Z}}_k - \underline{\mathrm{Z}}_{k-1}$

{Identify the production candidates}

$\underline{\alpha}_k \leftarrow \mathbb{W}^+\underline{\mathrm{Z}}_k \oplus \Sigma\mathbb{W}^+$

{Filter the production candidates to yeild the production set}

$\underline{\rho} \leftarrow \underline{\alpha}_k \oplus$ sign$(\mathbb{W}^+\underline{\mathrm{Z}}_k)$

for all $\rho \in \underline{\rho}$ **do**

{Create the input file set}

$\underline{\mathrm{X}} \leftarrow$ fileset $\left(\mathbb{W}^+_{\underline{\rho},*}\right)$

{Call each service in the production set}

fileset.push $\rho(fileset, parameters)$

end for

end procedure

2.0.0.1 Workflow Execution The workflow is executed using the $executeWorkflow\,()$ algorithm.

3 Summary

Algorithm 6 Execute Workflow

procedure EXECUTEWORKFLOW(fileList, parameters)
  for all $file \in fileList$ IN PARALLEL **do**
    for all $\rho \in rows(\mathbb{S})$ **do**
      {Construct the augmented graph with starting service (source) ρ}
      $\mathbb{M} = \begin{bmatrix} \frac{\rho}{\mathbb{P}} \\ \hline \mathbb{F} \end{bmatrix}$
      if CONNECTED($\mathbb{M}$) and $\rho.canCreate$(x) **then**
        fileset.push $\rho(file, parameters)$
      end if
    end for
  end for
end procedure

References