2013

Using Tarjan's algorithm to organize and schedule the computational workflow in a federated system of models and databases

Gabriel Sean McNunn
Iowa State University

Using Tarjan's algorithm to organize and schedule the computational workflow in a federated system of models and databases

by

Gabriel Sean McNunn

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Kenneth Mark Bryden, Major Professor
Richard LeSar
Xinwei Wang

Iowa State University

Ames, Iowa

2013

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor Mark Bryden for his guidance and patience throughout the course of my research and graduate education. His knowledge and experience helped to guide me through this process and focus my research goals. I would also like to thank Kris Bryden for her time and attention to detail that helped me to write this thesis.

I would like to thank my grandmother Sonja for her endless encouragement and support throughout my life and my education. You have always been an influential part of my life, and I am grateful for the support you have given me.

Lastly, I would like to thank my mother Tonia for the unconditional love and encouragement she has provided me throughout the course of my life and my education. I dedicate this thesis to you for all the support you have given me through the ups and the downs. You have always been there to push me when I've needed it, and your patience, dedication, and insights have been influential in helping me achieve my goals in both academics and life. I will always be grateful for everything you have done for me.

ABSTRACT

This thesis examines the use of Tarjan's algorithm for finding strongly connected components as a mechanism for organizing and scheduling the computational workflow in a federated system of computational models and databases. Often the emergent behavior of large-scale complex engineered and natural systems is the consequence of the interaction between the subsystem components that compose the system. As a result, modeling and simulating these large-scale complex systems requires a large number of heterogeneous computational models and databases to be assembled into a federated architecture that supports interoperability and information sharing between components. Although the local data exchanges that are needed to simulate the connectivity between the components of these federated systems is often well understood, identifying the workflow that is needed to accurately propagate data through the overall system is challenging and often not practical using a hands-on or brute-force approach. As a result, a novel method is needed that identifies the computational workflow required to accurately propagate data through a large-scale federated system of models and databases, ensuring each component receives the correct input data from other components in the system prior to it being solved or queried. This thesis develops a methodology that utilizes Tarjan's algorithm as a mechanism for identifying data-related interdependencies and for scheduling the necessary workflow that is needed to solve federated systems of models and databases based on the local input and output data associated with each of the components. The methodology is applied to identify the computational workflow needed to solve a system of one-dimensional models and databases representing the heat transfer and thermal stress in a gas turbine blade, and a system of models and databases representing the performance of a hybrid gas turbine, solid oxide fuel cell energy system. The goal is to extend the use of the algorithm as a tool for

organizing and scheduling the computational workflow in federated systems models and databases that are developed, validated, revised, and executed independently using distributed or cloud-based resources.

CHAPTER 1

INTRODUCTION

1.1 Overview

The emergent behavior of large-scale complex engineered and natural systems is often not discernable from the behavior or performance of the independent subsystem components that compose the system, but instead is dependent on the interaction between these components. However, many existing modeling and simulation tools are developed as stand-alone applications to describe a system with respect to a particular set of physics and scale while relying on fixed or transient boundary conditions to accurately describe the interconnectivity of the model within a larger system domain. As a result, investigating the impact of multi-physics and multi-scale interactions on the overall performance or behavior of systems has traditionally relied on simplified model sets, or the construction of a physical prototype and empirical methods. Although several computational methods and frameworks for integrating and coupling high-fidelity multi-physics and multi-scale models have been proposed and implemented, the ability to assemble and link the large number of models and databases needed to fully describe large-scale complex systems remains a significant barrier (Joppich and Kurschner 2006, Larson et al. 2005, Hill et al. 2004, Bitz et al. 2012, David et al. 2013, Ford et al. 2003, Patzak et al. 2013). One particular issue that must be addressed is the lack of ability to identify the computational workflow that is needed to accurately propagate information or data through the system of models and databases. The workflow must identify components of the system sharing data-related interdependencies that require an implicit solution and the sequence of data exchanges, model runtimes, and database queries that is needed to ensure each component in the system receives the necessary input information prior to being called or queried. Although the

local data exchanges that are needed to simulate the connectivity between any two models and databases are often well understood, the system-level workflow is less apparent, particularly when a large number of models and databases are assembled.

Existing high-fidelity model coupling frameworks require a manual or hands-on approach to defining a computational workflow and limit the number of components that may be included in a simulation. Although this approach works well for a small set of models, as the detail and scope of a simulation increases, the number of component models that is needed and the complexity of the workflow tend to grow. By utilizing a federated or cloud-based architecture, limits on the computational resources that are needed to handle a large-scale system of models and databases can be distributed, allowing large networks of models and databases that are developed, validated, revised, and executed independently from one another to be assembled and exchange data to simulate their connectivity and interaction. However, a mechanism that enables the computational workflow to automatically be identified based on the local input and output data associated with each of the independent system components is still needed.

Tarjan's algorithm for finding the strongly connected components of a directed graph has been adapted and utilized for a variety of organizational tasks including its implementation in the Engineering Equation Solver (EES) software as a tool for blocking and scheduling the solution sequence to solve large, sparse systems of equations (Klein and Alvarado 1998). Specifically, the algorithm is used to identify implicit equation sets and schedule the sequence in which the sets must be evaluated to solve the whole system. Similar to equations, models and databases require input information and respond with the output of new information that can be used to gain insight or as the input of another equation, model, or database. Consequently, like large, sparse systems of equations, large-scale federated systems of models and databases require

a scheduling approach to satisfy all of the information-related dependencies and interdependencies between components. Based on this characteristic, the use of Tarjan's algorithm can be extended to organize and schedule the workflow in systems of models and databases. Specifically, the algorithm identifies components with data-related interdependencies that require an implicit solution and schedules the necessary computational workflow to indicate the sequence of data exchanges, model runtimes, and database queries that is needed to accurately solve the system.

1.2. Thesis Summary

This thesis develops a methodology in which Tarjan's algorithm is used as a mechanism for organizing and scheduling the computational workflow in a large-scale federated system of models and databases. Details of the methodology, its implementation, and its application to two disparate systems of models are presented in a journal article to be submitted to the journal *Advances in Engineering Software*. The goal of this paper is to introduce a solution to one of the challenges associated with the assembly of large-scale federated systems of models and databases.

CHAPTER 2

THE USE OF TARJAN'S ALGORITHM FOR ORGANIZING AND SCHEDULING THE COMPUTATIONAL WORKFLOW IN A FEDERATED SYSTEM OF MODELS AND DATABASES

A paper to be submitted to the journal *Advances in Engineering Software*

Gabriel S. McNunn, Kenneth M. Bryden, Richard LeSar

Abstract

This paper proposes a novel application of Tarjan's algorithm for finding strongly connected components as a mechanism for organizing and scheduling the computational workflow in a federated system of heterogeneous models and databases. Often the emergent behavior of these systems is a consequence of the interaction between the subsystem components that compose the system rather than the independent behavior or performance of the components. However, existing models and simulation tools are primarily developed as stand-alone applications, simulating a system based on specific physics and scale. As a result, methods and tools are needed that enable large numbers of heterogeneous, distributed computational models and databases to be assembled into large-scale system simulations. Several high-fidelity model integration and coupling frameworks have been developed; however, the ability to couple a large number of models and handle complex workflows remains a significant challenge. This paper demonstrates the use of Tarjan's algorithm as a mechanism for identifying the computational workflow needed to solve a large-scale federated system of models and databases. The algorithm identifies interdependent models that require an implicit solution, and schedules the sequence of data exchanges, model runtimes, and database queries that are needed to propagate information and ensure each component receives the correct input data prior to its execution or query as part of the federated system.

2.1 Introduction

Modeling large-scale complex systems, products, and processes is becoming increasingly important. However, the difficulty of modeling larger and more interconnected systems with a high level of fidelity is a significant barrier. Often this barrier is due to the behavior of these systems that is unapparent from the independent behavior or performance of the subsystem components that compose the system, but instead is a result of the interaction and interconnectivity between these components. Consequently, the use of independent models may not provide the scope or level of detail that is needed to accurately simulate the behavior and dynamics of large complex systems as a whole (Voinov and Cerco 2010). For example, the performance of a wind farm depends on the performance of the individual wind turbines as well as the interaction between the wind turbines and the local topography. Consequently, simulating the performance of a wind farm may require a system of models that includes a boundary layer model to simulate the flow dynamics at the surface of a turbine blade, a near field fluid dynamics model to capture the vortex shedding from the individual blades, and a far field fluid dynamics model to simulate the impact of wind towers, downstream wakes, and topography (Calaf et al. 2010, Singh and Ahmed 2013, Lanzafame et al. 2013, Chatelain et al. 2013). In addition to these models, structural models describing the loads and elasticity experienced by the wind turbine blades (Brazilevs et al. 2011), gear-train models predicting the stresses and strains experienced by the drive shafts, bearings, bushings, gear teeth, and spline couplings in the gearboxes of the wind turbines (Xing and Moan 2013), models predicting the electrical output of the wind turbines based on empirical generator power curves, economic models describing the levelized cost of electricity generated by the wind farm, and other models and databases describing different aspects of the system and providing greater detail.

To capture the richness, fullness, and complexity of large-scale complex engineered and natural systems, methods and tools are needed that enable large numbers of heterogeneous models and databases to be assembled and linked. The goal is to allow disparate models and databases that are developed, validated, revised, and executed independently from one another to be assembled into a federated architecture that supports their interoperability and facilitates the exchange of data between components to simulate their interconnectivity and interaction. Although the local exchange of data between two models or databases is often well understood, identifying how data must be propagated through a large-scale system of models and databases to ensure each component receives the correct input information prior to being called or queried is less intuitive. As a result, existing high-fidelity model integration and coupling frameworks often limit the type and number of models that can be included in a simulation and do not enable the components to be coupled in different ways. However, in some cases the frameworks rely on the manual specification of the information flow through the system of coupled models. That is, the user must explicitly specify the local data that must be exchanged between component models, the nature of the data exchange, and the workflow needed to accurately propagate to each of the components in the system. Although this approach works well for a small number of component models, as the scope and detail of a simulation increases the number of models needed to fully describe a system quickly increases. As a result, the number of component interactions and the complexity of the workflow needed to solve the system grow rapidly.

This paper proposes a novel use of Tarjan's algorithm for finding strongly connected components as a mechanism for identifying the computational workflow in federated systems of models and databases that are assembled to represent large-scale complex engineered and natural systems. Based on the local input and output data associated with each of the models and

databases included in the system, the algorithm identifies components with data-related interdependencies requiring an implicit solution and schedules the sequence of data exchanges, model runtimes, and database queries that are needed to evaluate the system as a whole and ensure each component receives the necessary input information. Following a description of Tarjan's algorithm and a discussion of its application for organizing and scheduling the computational workflow in federated systems of models and databases, two alternative systems consisting of several disparate component models and databases are examined. The first is a one-dimensional system of models and databases representing the heat transfer and thermal stress in the wall of a convection-cooled gas turbine blade with thermal barrier coating. The second is a system of models and databases that is assembled to represent the performance of Hyper (Hybrid Performance Project), a hybrid energy generation system consisting of a 120 kW gas turbine and a simulated 300 kW solid-oxide fuel cell (SOFC). The system, located at the National Energy Technology Laboratory in Morgantown, West Virginia, is used to investigate strategies for controlling the pressure drops and surges that are characteristic of the hybrid system. (Traverso et al. 2012, Winkler et al. 2006).

2.2 Background

Existing methods for assembling disparate computational models follow a monolithic or partitioned approach to integration and coupling. The monolithic approach results in a single aggregate source code and executable that encompasses all of the independent source codes of the component models included in the system (Walhorn et al. 2005, Ryzahkov et al. 2010, Heil 2008). Alternatively, the partitioned approach maintains the modularity of the component model source codes, relying on an external infrastructure to exchange the necessary data between the different components and coordinate the model runtimes and database queries. Each approach

has advantages and disadvantages. The monolithic approach requires no external communication between software components and ensures model interoperability by avoiding issues related to conflicting data structures, protocols, languages, and code semantics, e.g., parameter names and variable definitions (Holzworth et al. 2010). However, the benefits of this approach are often lost due to the extensive time and effort required during the development stages of the monolithic code, particularly when large numbers of models are to be incorporated into the system. As additional models are integrated, the complexity and size of the code can often become unmanageable, limiting the ability to make revisions or repurpose the code (David et al. 2013). As a result, the simulation of a different system or even the addition, substitution, or removal of new models often requires the development of an entirely new monolithic code.

In contrast, the partitioned approach reduces the software development challenges associated with the construction of a monolithic code by maintaining the modularity of the component models (Sternel et al. 2008, Tabiei and Sockalingam 2012, Jaio et al. 2006, Sonntag et al. 2013, Kattke et al. 2011, El Khoury et al. 2013, Branger et al. 2010, Malleron et al. 2011). This enables the component source codes to be developed and validated in isolation from one another by experts and allows the potential to integrate and couple legacy models, in-house research codes, and commercial modeling and simulation packages. The partitioned approach typically requires some direct alteration of the component model source codes to use a framework specific API (Joppich and Kurschner 2006) or include specific data structures, classes, and functions (Larson et al. 2005). However, developing model specific wrappers that provide an interface between the models and the framework can minimize the “invasiveness” or degree to which the source codes must be altered. The wrappers function to translate incoming and outgoing data to the necessary structure and format (Lloyd et al. 2011, Joppich and

Kurschner 2006). An infrastructure that supports the exchange of data between the components is also required as well as a method of reconciling the different temporal and spatial scales associated with each of the models (Brandmeyer and Karimi 2000). Often the communication infrastructure includes mesh-association tools and interpolation stencils to assist with exchanging data between models with disparate spatial discretizations with respect to structure, resolution, or dimensionality (Johnson et al. 2011).

The main advantages of the partitioned approach are the flexibility and scalability that it offers compared to the monolithic approach (David et al. 2013). Specifically the flexibility to add, substitute, and remove models without affecting the overall architecture of the system and the scalability to couple large numbers of models that are developed, validated, revised, and executed independently from one another across a network of distributed or cloud-based resources.

Methods and standards for establishing model interoperability, or the ability of two or more computational models to exchange information and use the information, are essential to enabling the assembly of large-scale systems of disparate models and databases (Rezaei et al. 2014). Often the terms *integration* and *coupling* are used synonymously when referring to the process of establishing the interoperability between two or more computational models. However for clarity, this paper considers the two terms to be separate and distinct aspects of model interoperability. Integration refers to the process of resolving the inconsistencies between the disparate models including numerical aspects (e.g., units, dimensionality, and order of accuracy) and software aspects (e.g., semantics, language, data structures, and I/O protocols). To simplify the generally rigorous process of integrating heterogeneous models, development standards (Kumfert et al. 2006) and API's (Knapen et al. 2013) have been created and accepted within

some communities. The standards provide a common structure and semantics for building new stand-alone models that can more easily be integrated with other models that are developed using the same standards.

In contrast, coupling refers to the exchange of data between two or more models to simulate their interaction or connectivity. The process of coupling models requires the identification of all data-related dependencies and interdependencies that exist between the models and an infrastructure for exchanging the data. The nature of the coupling (i.e., the manner in which information is exchanged) between any two models dictates the interaction between the models and can significantly impact the accuracy or behavior of the simulation as a whole. One-way coupling refers to the use of output data from one model or database as the input data of another model and results in an explicit sequence in which the models or databases must be solved or queried. Alternately, two-way coupling refers to the bi-directional exchange of data between two models as a consequence of a data-related interdependence that must be resolved, for example, the conditions at a shared spatial boundary separating two models (Patzak et al. 2013). Similarly, interdependencies can also arise due to a series of one-way couplings that exist between three or more models and consequently require an implicit solution to be resolved.

Several frameworks and tools have been created to support the integration and coupling of disparate models and simulate a variety of multi-physics and multi-scale systems. These include MpCCI (Joppich and Kurschner 2006), the Model Coupling Toolkit (MCT) (Larson et al. 2005), the Earth System Modeling Framework (ESMF) (Hill et al. 2004), the Community Climate System Model (CCSM) (Bitz et al. 2012), the Object Modeling System (OSM) (David et al. 2013), VE-Suite (McCorkle and Bryden 2007), and the General Coupling Framework (GCF) (Ford et al. 2003). However, existing high-fidelity model integration and coupling frameworks

often limit the type and number of models that can be included in a simulation as well as the manner in which the components may be coupled. As a result, the workflow needed to solve the system is entirely concurrent, requiring an implicit solution to solve all of the models, or be entirely sequential, requiring the independent solution of each component in an explicit order. In more flexible frameworks, the user must explicitly specify the local data that is exchanged between all component models, the nature of each data exchange, and the overall computational workflow needed to ensure each model receives the correct data prior to being solved. Although this works well for a small number of components, as the scope and detail of a simulation increases, the number of models and databases needed to describe the system grows rapidly. As a result, methods are needed that identify the computational workflow needed to accurately propagate information to each of the components in a large-scale federated system of models and databases.

2.3 Tarjan's algorithm

This paper focuses on a novel extension of Tarjan's algorithm as a mechanism for identifying the computational workflow in a large system of heterogeneous, distributed models and databases. Specifically, the goal of this work is to enable disparate computational models and databases that are developed, maintained, revised, and executed in independently from one another to be assembled into a federated system that supports interoperability and information sharing between components. The concept is similar to federated database management systems, conceived by Hammer and McLeod (Hammer and McLeod 1981) to describe a system of distributed and autonomous component databases that are integrated and assembled to various degrees. The component databases continue operation and maintenance outside of the federated system; however, users are now allowed to query both local information from a specific database

or global information from the overall federated system (Hiembigner and McLeod 1985, Sheth and Larson 1990). Like a federated database management system, a federated system of models and databases supports interoperability and information sharing but maintains the development autonomy of the components, allowing maintenance, revision, and validation of the models and databases to be done independently from one another. However, the assembly of a federated system of models and databases requires a scheduled approach to solve the system and to satisfy all of the data-related dependencies and interdependencies that link the disparate components. Although the local data exchanges between any two models or databases are often well understood, explicitly defining the flow of information through the overall large-scale system is typically unclear and difficult. As a result, a mechanism is needed for organizing and scheduling the computational workflow in a federated system of models based on the local input and output data associated with each of the component models included in the system.

Tarjan's algorithm (Tarjan 1972) for finding strongly connected components (SCC) was originally developed as an efficient algorithm for identifying the SCCs of a directed graph composed of a set of vertices connected by a set of directed edges. Each SCC represents a subgraph of the original directed graph in which a path, consisting of a series of directed edges and vertices, connects each vertex to every other vertex in the subgraph. That is, a cycle links all the vertices included in the SCC, unless the SCC contains only a single vertex. Condensing or collapsing the SCCs of a directed graph into single vertices results in a directed acyclic graph (DAG) lacking any directed cycles. It is this type of directed graph to which traditional task scheduling or topological sort algorithms are applied (Datla et al. 2011). However, as a result of the depth-first search utilized by Tarjan's algorithm to traverse a directed graph, the order in which the algorithm identifies the SCCs corresponds to the reverse topological sort of the

resulting DAG that is produced after all SCCs have been identified (Tarjan 1972). In addition to Tarjan's algorithm, other algorithms have been developed to identify the SCCs of directed graphs including the Kosaraju-Sharir algorithm (Sharir et al. 1981) and the path-based strong component algorithm (Gabow 2000). However Tarjan's algorithm is considered to be one of the most efficient and simple to implement of these, requiring a linear search time on the order of O(n) where n is the number of vertices in the directed graph (Tarjan 1972). Comparatively, the Kosaraju-Sharir and path-based strong component algorithms require a search time on the order of O(n^2) and O(n) respectively. Additionally, these algorithms do not provide the scheduling or topologically sort capabilities associated with Tarjan's algorithm.

A flowchart representing Tarjan's algorithm is shown in Fig. 1. Initialized at an arbitrary vertex, the algorithm selects an unexplored edge leading to a new vertex. From this vertex the process is repeated, and an unexplored edge is selected and traversed to a new vertex. The order in which the vertices are visited is stored on a stack, referring to an ordered list of vertices that represents the search path of the algorithm. In addition to being placed on the stack, as each vertex is visited it is assigned a *lowlink,* or pointer, indicating the lowest vertex on the stack that is found to be connected with the vertex to which the *lowlink* belongs. By default the *lowlink* of each vertex is initially set to reference itself (i.e., *vertex* $1_{\text{lowlink}} \rightarrow$ *vertex* 1) as no edges emanating from the vertex have yet been explored. As the algorithm progresses, the *lowlink* of a vertex is only altered if the algorithm encounters an edge leading from the vertex to a vertex that is already on the stack and lower on the stack than the vertex that is currently referenced by the *lowlink.*

The depth first search utilized by Tarjan's algorithm prohibits backtracking to previously visited vertices until a path has been followed as far as possible. That is, the search is always

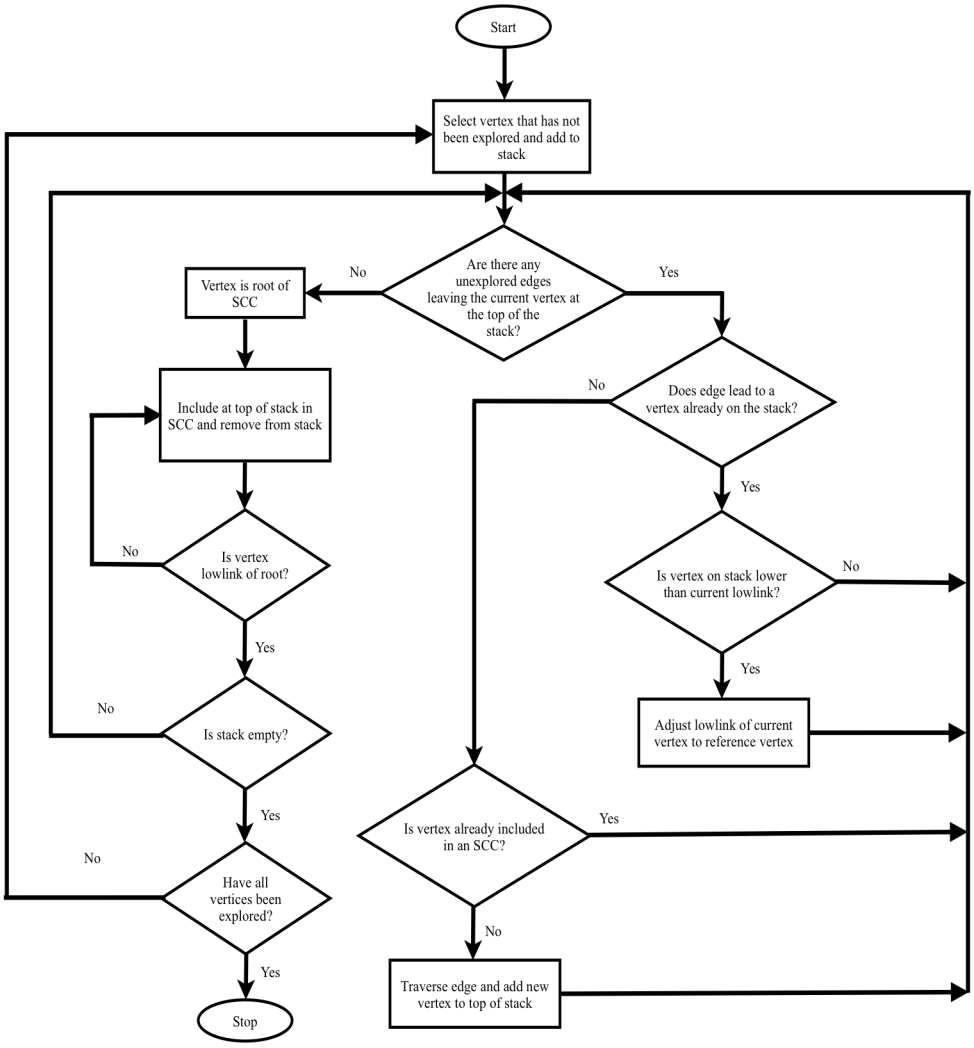

Fig. 1. A flowchart representation of Tarjan's algorithm (Duff and Reid 1978).

advanced from the vertex at the top of the stack until no unexplored edges, or available search paths, remain. The end of a search path indicates the *root*, or the last component that belongs to the an SCC has been reached. As a result, the *lowlink* associated with the *root* represents the earliest vertex that is included in the SCC; and therefore, all vertices in the stack positioned between the *root* and the vertex are also included in the SCC. In cases where the *lowlink* of the *root* indicates the *root* itself, the resulting SCC contains only the single component. The vertices included in the SCC are removed from the stack, and the algorithm continues the search from the vertex now positioned at the top of the stack. If the resulting stack is empty, a new search begins from an arbitrary unexplored vertex. The algorithm continues until all vertices have been explored and assigned to an SCC, at which point it is terminated.

Tarjan's algorithm has been utilized as a tool for a variety of organizational tasks including the triangulation of matrices (Duff 1978), linear temporal logic (LTL) verification (Geldunhuys and Valmari 2005), and blocking and scheduling the solution sequence for solving large, sparse systems of equations in the Engineering Equation Solver (EES) (Klein and Alvarado 1998) software package. Specifically, EES applies the algorithm to a directed graph or connection matrix representing a system of equations. The algorithm reduces the system into an ordered series of blocks, or implicit equation sets, that can be solved more quickly and efficiently than the larger system as a whole. For example, the system of eight equations in Table 1 can be represented as the connection matrix shown in Fig. 2 or as the equivalent directed graph shown in Fig. 3. Each equation is associated with a single unknown variable that must be solved. Therefore, the vertices of the directed graph represent the equations of the system and the directed edges represent shared variables that have been determined from the equations from which they emanate. Applying Tarjan's algorithm to the directed graph representing the system

Table 1. A system of eight equations.

#	Equation
1	$a + b = -c$
2	$b + c = 4 - h$
3	$2b + d = 0$
4	$c + e = 10 + f$
5	$d - e + 5f = a + 3$
6	$g = 1$
7	$3b + c - g = -2h$
8	$h = c + 3$

equation	connection matrix
1	$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix}$
2	
7	
3	
4	
5	
6	
8	

Fig. 2. A directed graph representing the system of equations in Table 1.

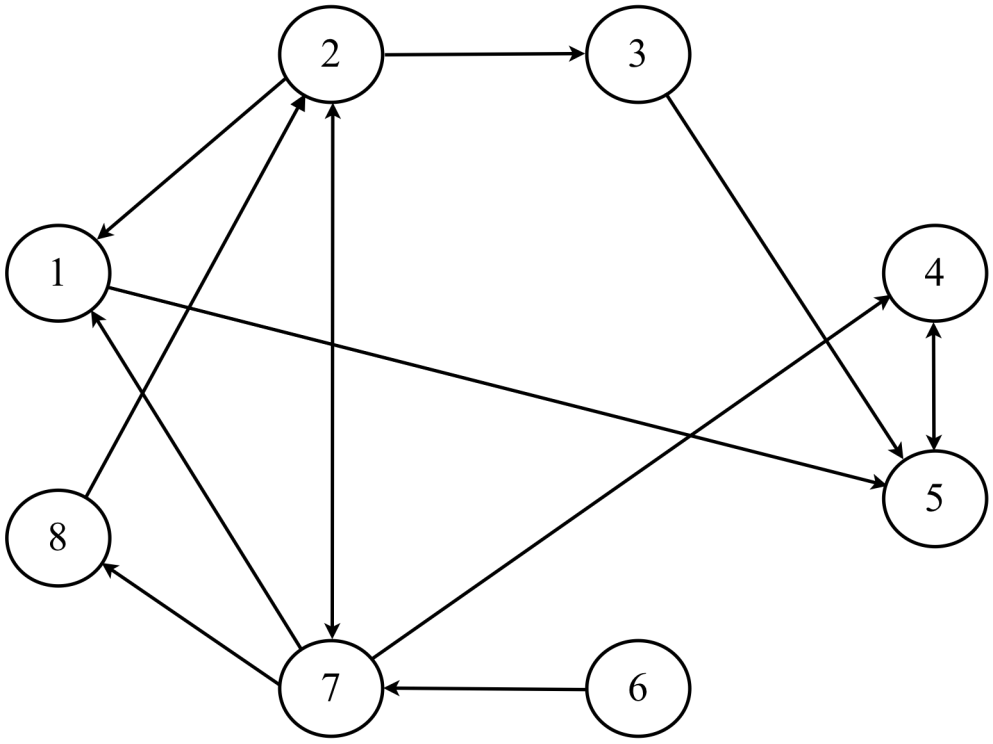

Fig. 3. A directed graph representing the system of equations in Table 1.

of equations results in the blocked solution sequence shown in a sorted connection matrix and a directed graph in Fig. 4 and Fig. 5, respectively. The sorted connection matrix shows a solution sequence beginning at the top of the matrix and moving downward. Blocks indicate the equations and corresponding unknown variables that must be solved for at each sequential step in the solution sequence, and the circled elements indicate variable values that have been determined at previous steps in the sequence and are now needed to allow the current block to be solved. Specifically, the connection matrix shown in Fig. 4 depicts a solution sequence in which the unknown value of *g* is initially determined from equation 6 and subsequently used within equations 7, 2, and 8 to implicitly solve for *c*, *b*, and *h*. The values of *c* and *b* are then used within equation 1 to determine the value of *a*, and the solution sequence continues until reaching the bottom of the matrix and solving for the values of *e* and *f*.

Similar to equations, computational models and databases can be considered as information nodes that provide a source of new usable information when provided with specific input information. Consequently, solving large-scale systems of models and databases requires a scheduled approach similar to that used by EES when solving large, sparse systems of equations. That is, a solution sequence or computational workflow is required that ensures each component receives the necessary input information from other components in the system prior to being called or queried. As a result, Tarjan's algorithm can be used as a mechanism for organizing and scheduling this workflow when applied to a directed graph or connection matrix representing all of the local data exchanges within a federated system of models and databases.

Applied to an *N*x*N* connection matrix in which the rows and corresponding columns represent the disparate models and databases included in the system. The algorithm is initialized at an arbitrary row (vertex) of the connection matrix, and a non-zero element (edge) within that

equation	blocked matrix								
6	[1	0	0	0	0	0	0	0	*g*
7	(1)	1	1	0	0	0	0	0	*c*
2	0	1	1	1	0	0	0	0	*b*
8	0	1	0	1	0	0	0	0	*h*
1	0	(1)	(1)	0	[1]	0	0	0	*a*
3	0	0	(1)	0	0	[1]	0	0	*d*
4	0	(1)	0	0	0	0	[1	1]	*e*
5	0	0	0	0	(1)	(1)	[1	1]	*f*

(1) solved [1] unsolved

Fig. 4. A directed graph showing the solution sequence identified by Tarjan's algorithm to solve the system of equations in Table 1 (blocks represent implicit equation sets).

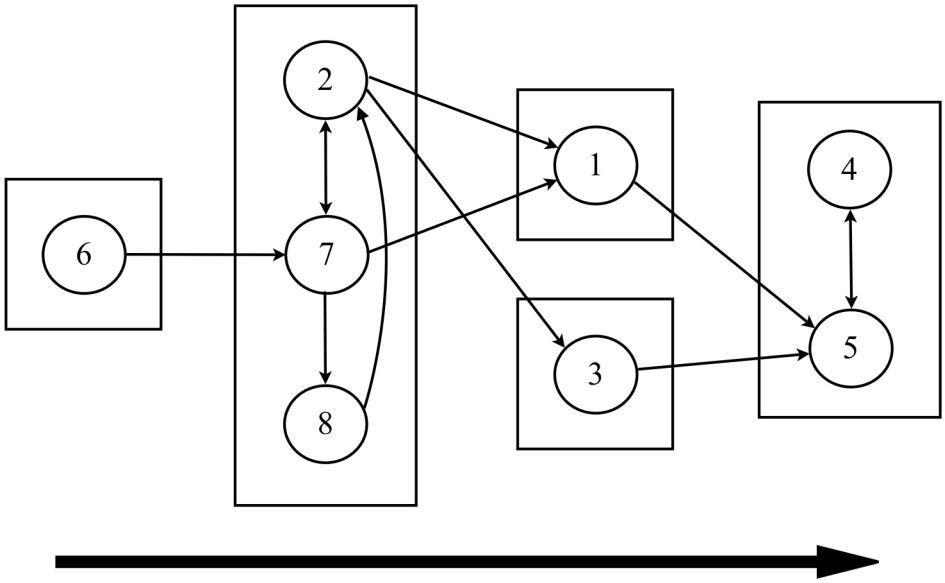

Fig. 5. A sorted connection matrix showing the solution sequence identified by Tarjan's algorithm to solve the system of equations in Table 1 (equivalent to those shown in Fig. 3).

row is randomly selected. The position of a non-zero element in the n^{th} column of the current row indicates an edge leading to the n^{th} row (new vertex) of the matrix. Traversing to the new row, the algorithm again selects a non-zero element, and the process is repeated. As previously noted, the order in which Tarjan's algorithm visits the rows is stored on a stack, and a *lowlink* is assigned to each row indicating the earliest visited row on the stack that is found to be connected to the current row. If a non-zero element within the current row leads to a row already on the stack, the *lowlink* of the current row may need to be adjusted to reflect the newly found connection, but only if the row on the stack was visited earlier than the row currently referenced by the *lowlink*. Regardless of any change to the *lowlink*, the algorithm advances the search from the row at the top of the stack. A *root* is reached when no unexplored non-zero elements remain within a row. This signifies a block of interdependent models has been found, and all rows included in the block are removed from the stack and the connection matrix before the algorithm continues. If the stack becomes empty, a new search begins from an arbitrary row remaining in the matrix. Once all rows within the matrix have been visited, the algorithm is terminated. The identified blocks require special consideration to resolve the data-related interdependencies. Often these interdependencies require an implicit solution method to be resolved; however, the addition of new models or databases can be used to avoid the interdependencies. The schedule of the workflow is indicated by the reverse order in which the model blocks are discovered by the algorithm.

2.4 Application to a 1-D heat transfer model of a gas turbine blade

As a result of the high temperature and pressure environment within a gas turbine, gas turbine blades are prone to a variety of thermo-mechanical issues such as creep, fatigue, corrosion, and thermal cycling. To minimize the temperature within a gas turbine blade, a variety

of cooling technologies have been developed including convection cooling within the internal structure of the blades and the application of thermal barrier coatings (TBC) to the exterior of blades. The internal convective cooling removes heat from the gas turbine blade, and the TBC provides a low thermal conductivity barrier that protects the blade against high-temperature oxidation, corrosion, and thermal cycling (Gupta et al. 2013).

Figure 6 shows a one-dimensional system representing the wall of a convection-cooled gas-turbine blade with thermal barrier coating. Similar to the method used in (El Khoury et al. 2013), the system is divided into six computational domains, including a cold air domain representing the internal cooling air within the gas-turbine blade; a nickel-based super alloy domain representing the gas turbine blade material, a domain representing the bonding layer (CoNiCrAlY) of the TBC, a domain representing the alumina (Al_2O_3) thermally grown oxide layer of the TBC, a domain representing the ceramic zirconia (ZrO_2) topcoat of the TBC, and a hot air/gas domain representing the high-temperature environment within the gas turbine, as shown in Table 2. To simulate the steady state heat transfer and thermal stress in the one-dimensional gas turbine blade, a system of disparate computational models and databases may be assembled. A list of the specific models included in the system is provided in Table 2. In this case, the models are assumed to be integrated and interoperable, avoiding software or model inconsistencies that would impact the exchange of data from one model to another. Figure 7 shows an aggregate directed graph containing all of the local data exchanges or couplings between the component models and databases within the system. This includes the exchange of the thermal conductivity values that are needed to determine the heat flux through each material layer of the TBC and blade substrate, and the exchange of temperature distribution data that is needed to evaluate the thermal stress in each distinct layer.

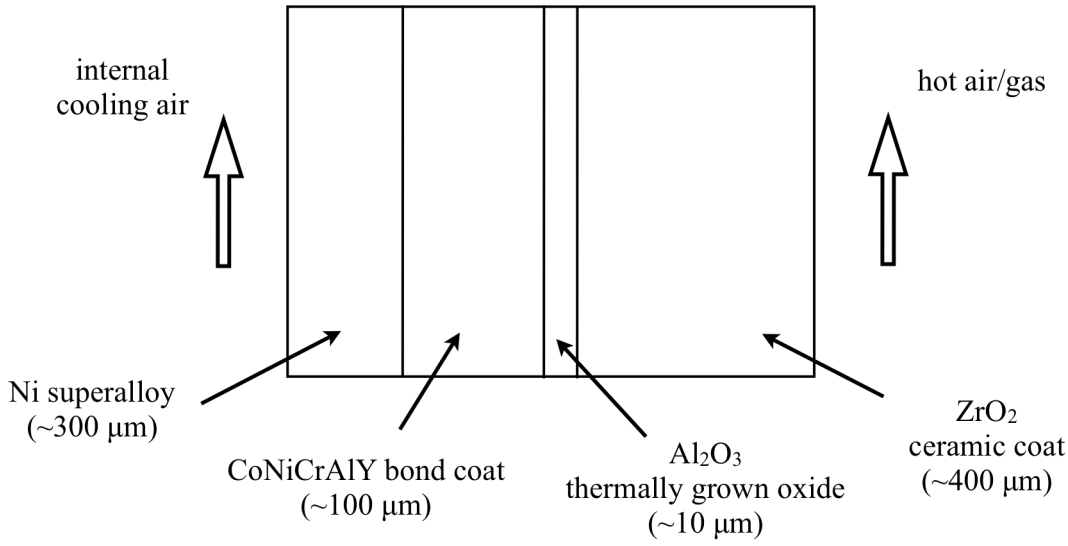

Fig. 6. A diagram of a convection-cooled gas turbine blade wall with a thermal barrier coating.

Table 2. Component models and databases assembled to represent the heat transfer and thermal stress in a 1-D gas turbine blade.

#	**Turbine blade models**
1	convective heat transfer model for internal cooling air (e.g. $q = hA(T_{surf}-T_{air})$)
2	nickel superalloy properties database (e.g. thermal conductivity, thermal expansion coefficient)
3	temperature distribution model for nickel super-alloy substrate (e.g. finite volume method (FVM) solver of Laplace equation)
4	thermal stress model for nickel superalloy substrate (e.g. finite element analysis (FEA) solver thermal stress $\sigma = E\alpha(T_{ref} - T_{hot})$)
5	bonding coat (CoNiCrAlY) material properties database (e.g. thermal conductivity k, thermal expansion coefficient α)
6	temperature distribution model for CoNiCrAlY bonding coat
7	thermal stress model for CoNiCrAlY bonding coat
8	thermally grown oxide (Al_2O_3) material properties database
9	temperature distribution model for Al_2O_3 thermally grown oxide
10	thermal stress model for Al_2O_3 thermally grown oxide
11	ceramic topcoat (ZrO_2) material properties database
12	temperature distribution model for $ZrCO_2$ ceramic coat
13	thermal stress model for ZrO_2 ceramic coat
14	convective heat transfer model for external hot air/gas
15	thermal barrier coating cost estimation model

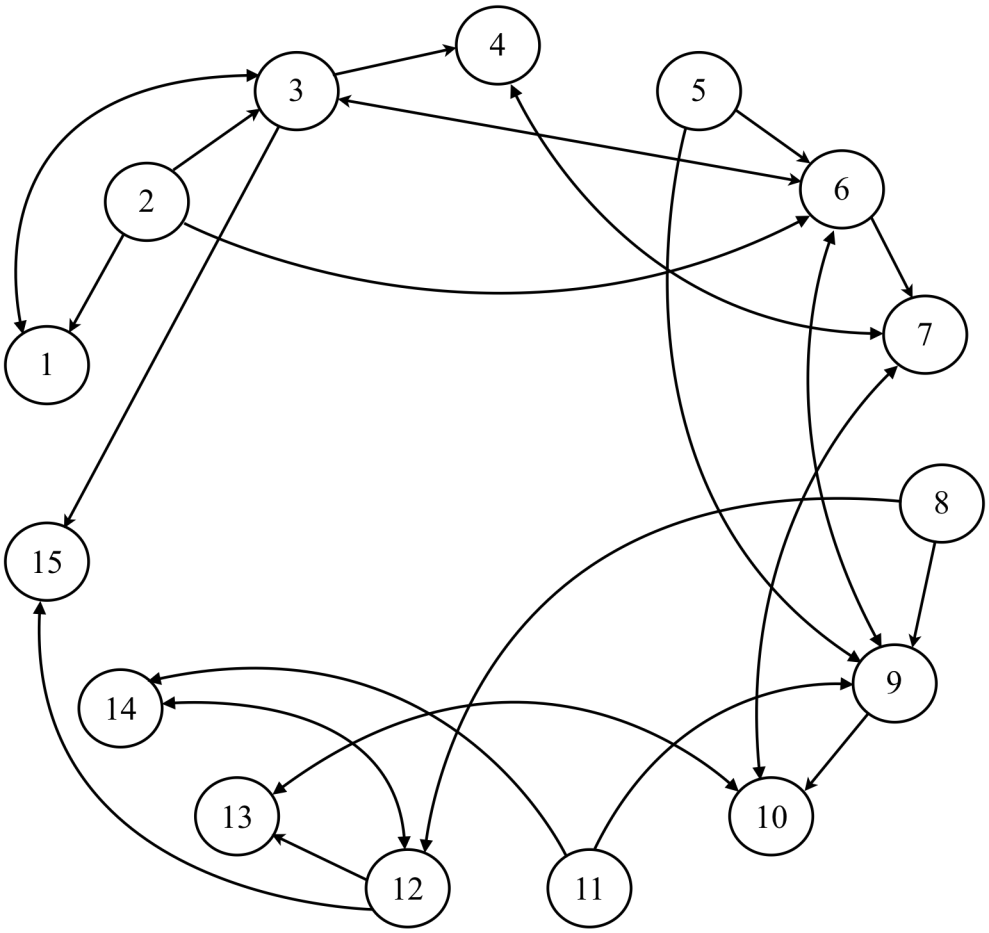

Fig. 7. An aggregate directed graph representing all of the local data exchanges in the system of models and databases included in Table 2 that describe the heat transfer, thermal stress, and cost.

Applying Tarjan's algorithm to the system results with the workflow shown in Fig. 8, which includes seven distinct blocks, or component groups, that can be executed and queried over three sequential steps. Specifically, the workflow depicts an initial exchange of thermal conductivity values from each of the independent material databases (2, 5, 8, 11) to the corresponding temperature distribution solvers (3, 6, 9, 12) and the cold air (1) and hot air (14) boundary models. The thermal conductivity is needed to calculate the heat transfer through the blade substrate and TBC layers. The temperature distribution solves and convection boundary models are organized into a single block due to the interdependence related to energy conservation at the shared boundaries. Dimensional data indicating the thickness of the overall TBC is then transferred to the simplified cost model (15) to estimate the cost-effectiveness of the TBC, and temperature data from each of the thermal distribution models is transferred to the corresponding thermal stress models (4, 7, 10, 13) associated with the substrate and TBC layers. As a consequence of the tensile and compressive forces that act on each layer by the adjacent layers, the stress experienced in each layer is interdependent with the stress in the alternative layers. Therefore, an implicit solution similar to that used to resolve the interdependencies between the temperature distribution models is needed.

2.5 Application to a system of models representing the Hyper energy system

To increase efficiency and lower energy costs, existing energy technologies can be integrated with fuel cells to create hybrid power generation systems. One example of a hybrid systems being considered is integrated gas turbine, solid oxide fuel cell (SOFC) systems, which can achieve an efficiencies of up to 70% based on the lower heating value of natural gas fuel (Winkler and Lorenz 2002). However, before these systems are commercially viable several issues must be addressed including the ability to control the interaction between the high-

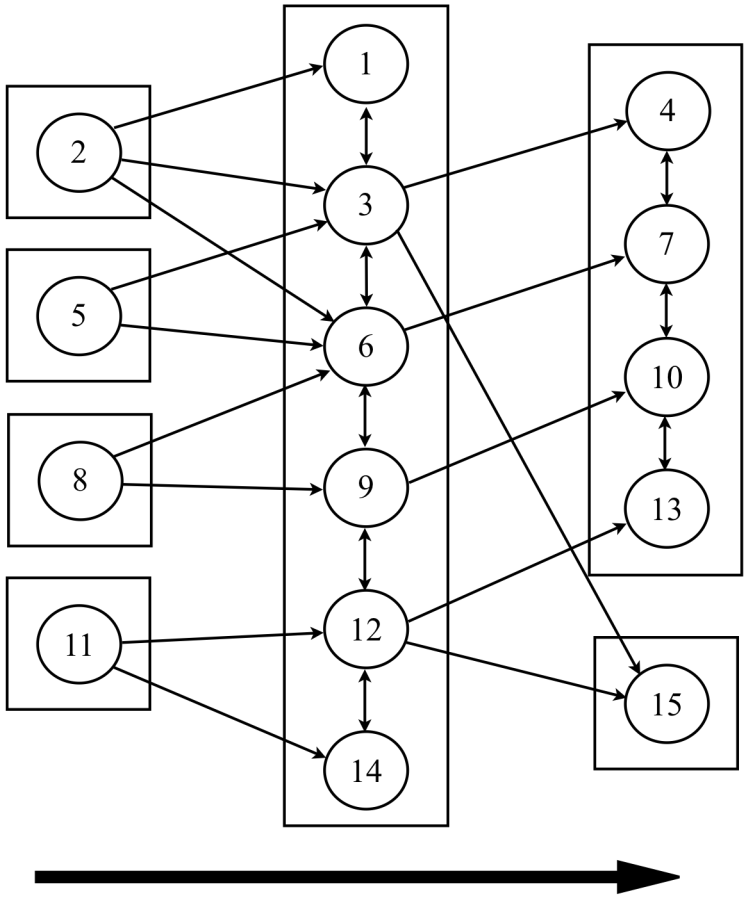

Fig. 8. The resulting workflow identified by Tarjan's algorithm to solve the system of models and databases shown in Fig. 7.

pressure flow of a gas turbine and the fragile fuel cell materials that if not regulated, can lead to severe equipment damage or malfunction (Winkler et al. 2006). To investigate strategies for controlling these types of hybrid systems, a prototype system has been developed at the National Energy Technology Lab (NETL) in Morgantown, West Virginia. The system is composed of a 120 kW gas turbine and several components (air plenum, combustor, and post-combustor) that simulate the heat effluence and stack volume of a 300 kW SOFC. To control the pressure drops and surges within the system during operation, a cold-air bypass valve, a hot-air bypass valve, and a bleed-air valve are manipulated. The position of these valves relative to the other components in the system is shown in the airflow schematic of the system in Fig. 9.

The performance of the Hyper system is non-linear and as a result, modeling and simulating the system with a high level of fidelity presents a significant challenge. Although the performance of the individual components that compose the system is well understood, it is the interaction between the components that defines the overall behavior and dynamics of the system. As a result, simulating the performance of Hyper requires the integration and coupling of several disparate models and databases that describe smaller subsystem components within the overall system. A description of the potential models and databases that are included to represent the Hyper system are listed in Table 3. As a result of the directional airflow within the physical system, the models representing the subsystem components must exchange information, or data, relating to the mass flow, temperature, and pressure of the incoming airflow. Specifically, one-way coupling is used to transfer mass-flow, temperature, and pressure data between adjacent components in the downstream direction of the airflow, and two-way coupling is used to resolve the interdependence between the work of the compressor and turbine. An aggregate directed graph representing all the local data exchanges between the component models and databases is

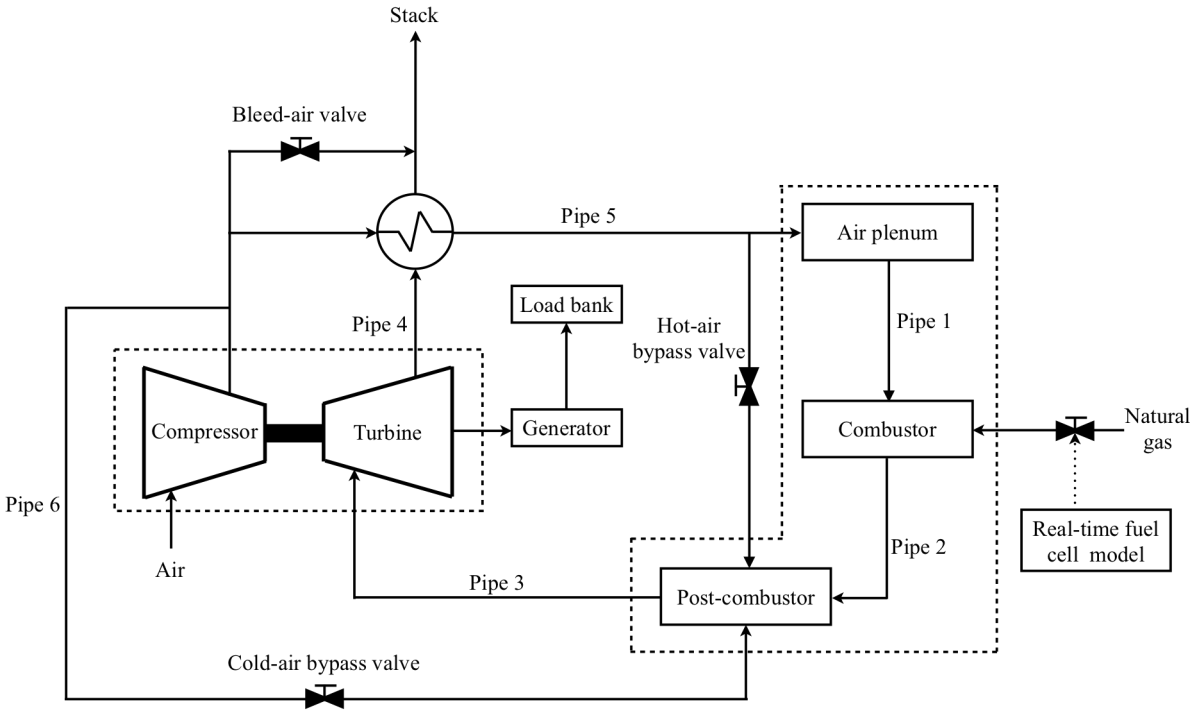

Fig. 9. An airflow schematic of the Hyper system.

Table 3. Component models and databases assembled to represent the performance of the Hyper system.

#	HyPer models
1	air plenum model
2	combustor model
3	post-combustor model
4	gas turbine model
5	empirical turbine back pressure database
6	compressor model
7	heat exchanger model
8	generator model
9	stack model
10	bleed air valve model
11	cold-air bypass valve model
12	hot-air bypass valve model
13	ambient conditions model
14	fuel cell controller model
15	pipe 1 model
16	pipe 2 model
17	pipe 3 model
18	pipe 4 model
19	pipe 5 model
20	pipe 6 model

shown in Fig. 10. Applying Tarjan's algorithm to the system of models results in the sorted workflow shown in Fig. 11. The workflow depicts a predominate block containing a majority of the component models. Due to the number of models included in the block, an implicit solution may require extensive computational time and resources to resolve all of the data-related interdependencies and as a result may be impractical. However, Tarjan's algorithm can be used to examine a variety of coupling scenarios between the components of a system. Altering the local data exchanges between two or more specific component models or databases and reapplying Tarjan's algorithm to the subsequently altered directed graph results in a an entirely new workflow. In this way, the algorithm provides a means of gaining insight and identifying the impact that a specific local component-to-component coupling has on the ability to solve the overall system. This includes substituting new component models and changing the nature of coupling between models. Therefore, the usefulness of Tarjan's algorithm is further extended, enabling users to understand how the flow of information at a local level in the system of models and databases impacts the ability to solve the system model as a whole. This includes substituting new component models and changing the nature of coupling between models. For example, removing the mass-flow, temperature, and pressure data dependence between the Gas Turbine Model (4) and Pipe 3 Model (17*)* and substituting an alternative information source such as a database results in the adjusted directed graph shown in Fig. 12. Applying Tarjan's algorithm to the new graph results in the dramatically simplified workflow shown in Fig. 13. The new workflow consists of 19 model blocks compared to the 7 blocks found within the original system. This indicates that the substitution of a new information source, such as an empirical database, previous simulation results, or real-time sensor data to provide the Gas Turbine Model and Pipe 3 Model with the necessary mass-flow, temperature, and pressure data, would

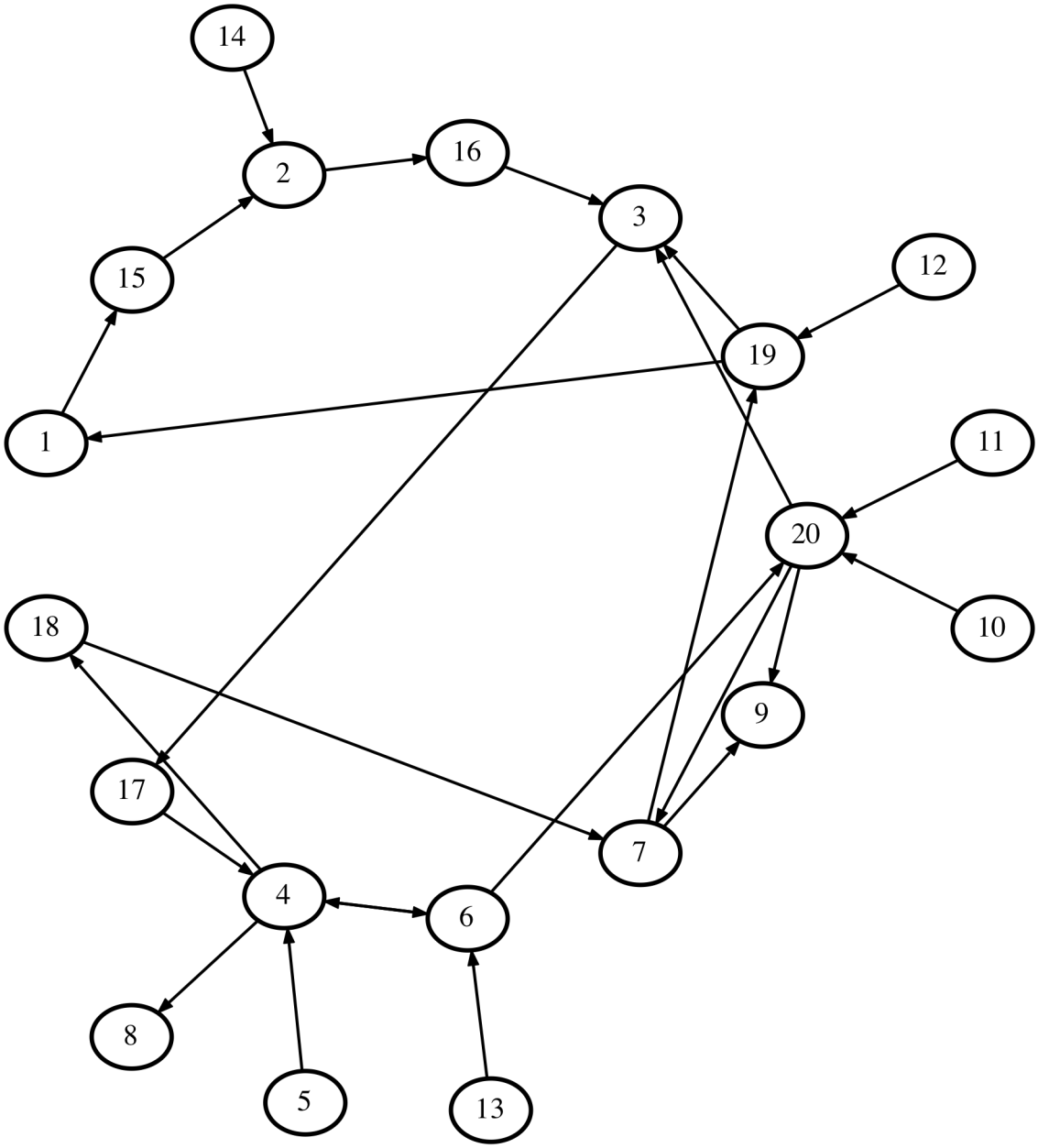

Fig. 10. A directed graph representing the system of models and databases included to represent the performance of the Hyper energy system.

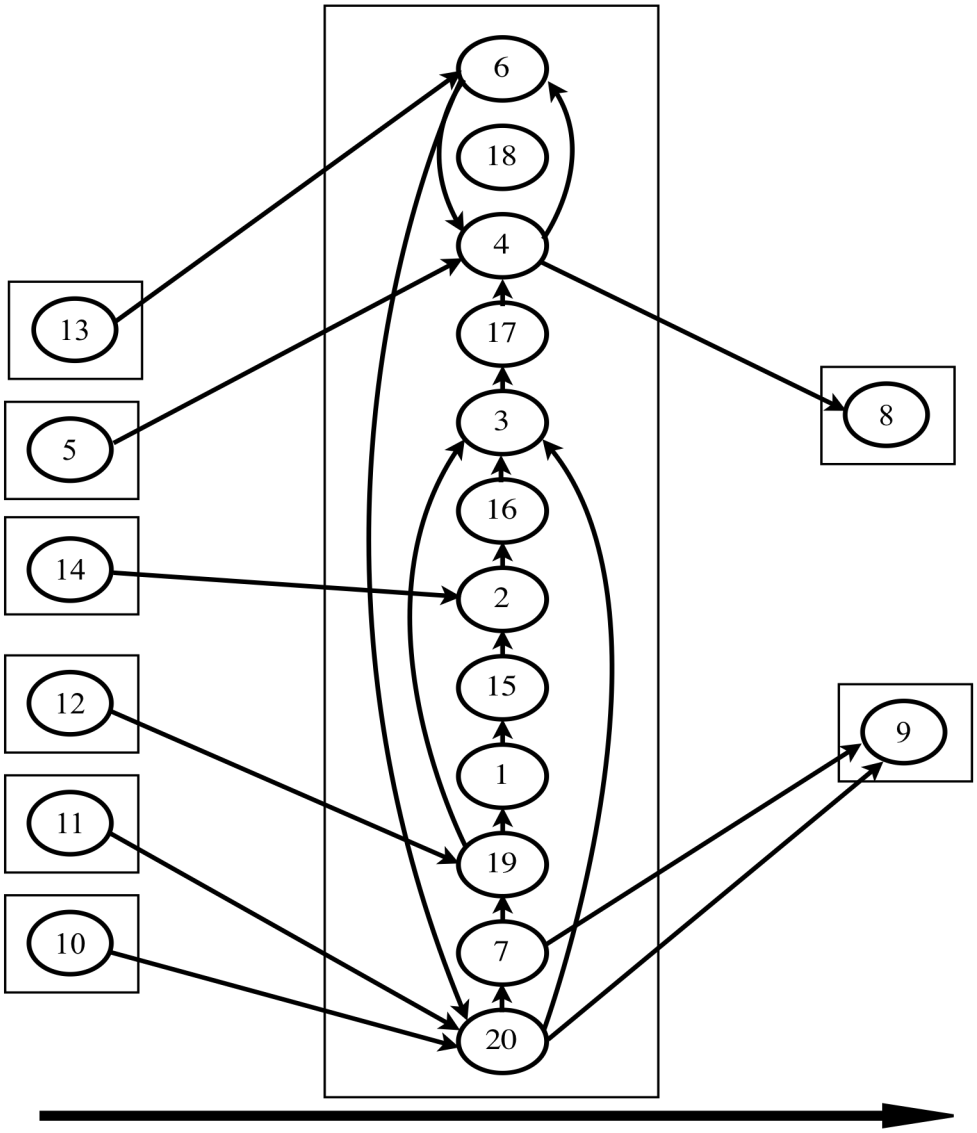

Fig. 11. The resulting workflow identified by Tarjan's algorithm to solve the system of models and databases shown in Fig 10.

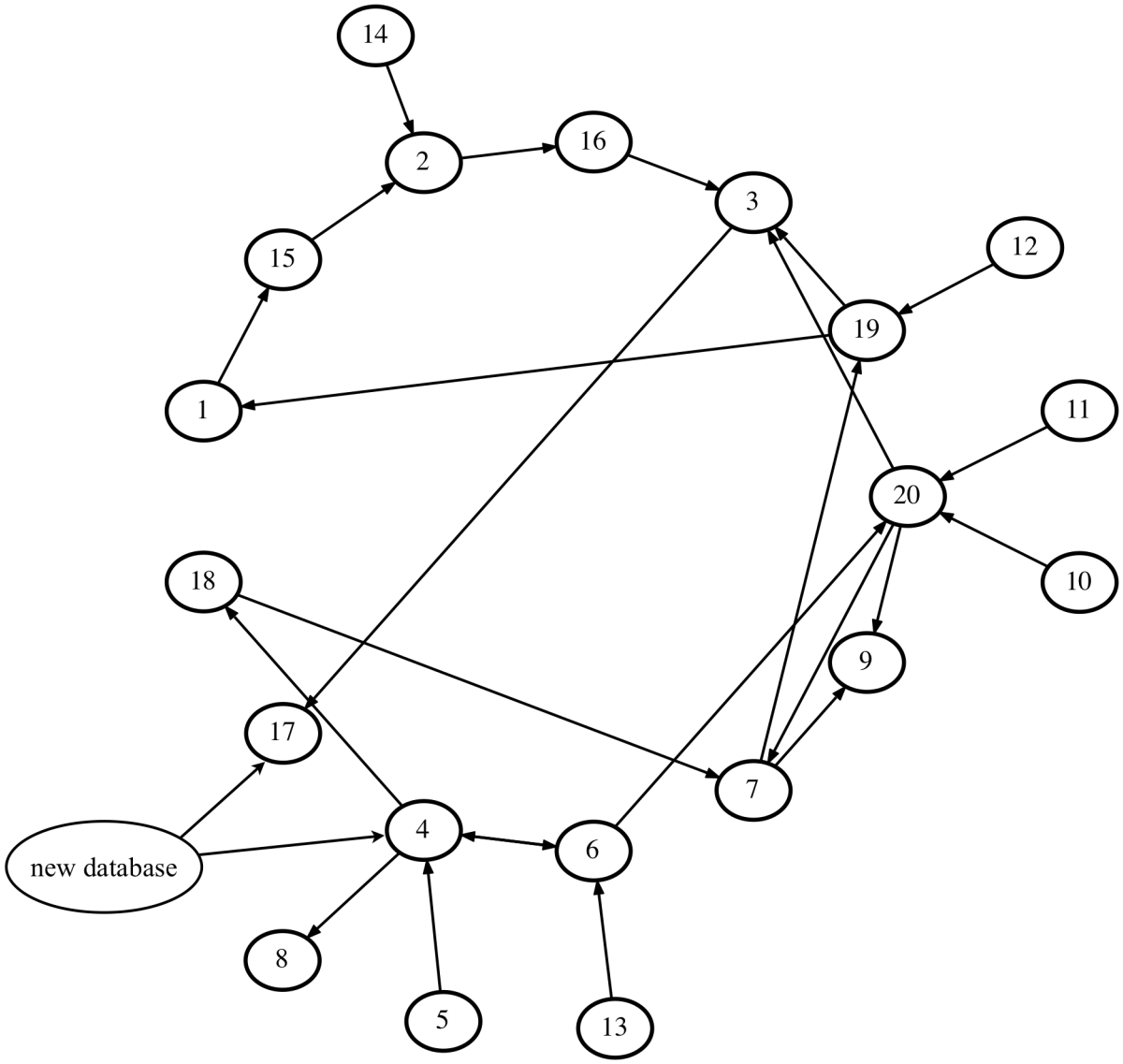

Fig. 12. The resulting system directed graph after removing the coupling between the Pipe 3 Model and the Gas Turbine Model and substituting a database.

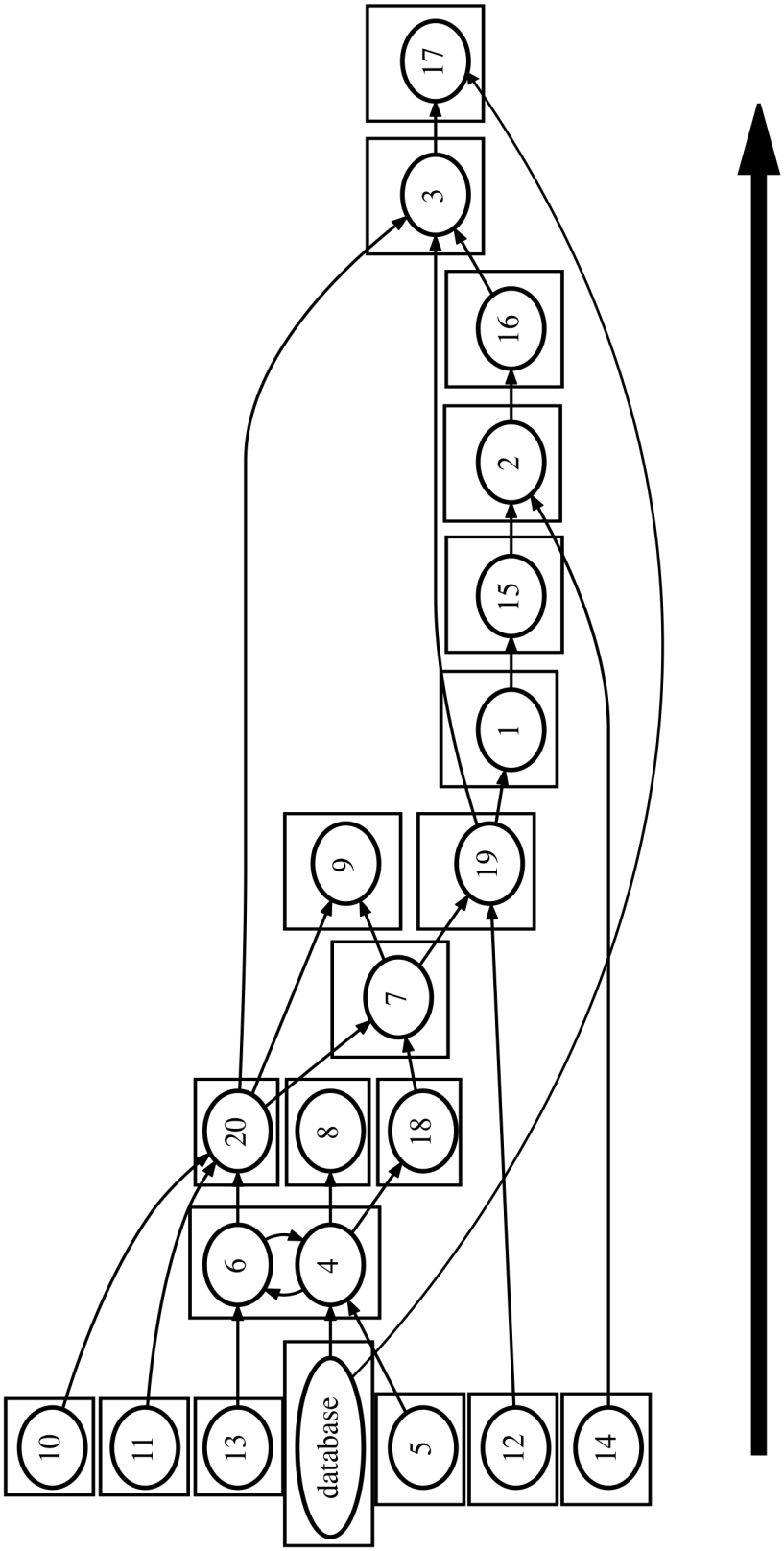

Fig. 13. The resulting workflow identified by Tarjan's algorithm to solve the altered system of models shown in Fig. 12.

significantly reduce the number of interdependent components and result in a more explicit workflow. Consequently, the ability to model and simulate the system using the new configuration of models and data exchanges is greatly simplified.

2.6 Conclusions and future work

This paper proposes and demonstrates the use of Tarjan's algorithm for finding strongly connected components as a mechanism for organizing and scheduling the computational workflow in large systems of heterogeneous, distributed models and databases. As the desired scope and detail of simulations increase, new methods are needed that enable large numbers of models and databases to be integrated and coupled to build comprehensive and detailed system simulations. To avoid limitations on computational resources, these methods must rely on the flow of information between models and databases that are developed, maintained, and executed in isolation from one another across a network of distributed and cloud-based computational resources. As a result, solving a federated system of models and databases requires the identification of a computational workflow that accurately propagates information through the system to ensure each model and database receives the required input data prior to being solved or queried. Based on the local input and output data associated with each individual component model and database, Tarjan's algorithm identifies interdependent groups of models that require an implicit solution and schedules the sequence of data exchanges, model runtimes, and database queries that is needed to evaluate the overall system. The algorithm was utilized to organize and schedule the workflow associated with two disparate systems of disparate models and databases. The first system of models and databases was assembled to represent the heat transfer and thermal stress in a one-dimensional gas turbine blade with thermal barrier coating, and the second system was assembled to predict the performance of a hybrid gas turbine, SOFC energy

system.

2.7 Acknowledgements

This research was performed at the Ames Laboratory through the support of the Department of Energy–Fossil Energy Program under Contract No. DE-AC02-07CH11358.

2.8 References

Bazilevs Y, Hsu M-C, Kiendl J, Wüchner R, Bletzinger K-U. 3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades. Int J Numer Meth Fluids 2011; 65:236-53.

Bitz CM, Shell KM, Gent PR, Bailey DA, Danabasoglu G, Armour KC, et al. Climate sensitivity of the Community Climate System Model, Version 4. J Climate 2012; 25:3053-70.

Brandmeyer JE, Karimi HA. Coupling methodologies for environmental models. Environ Model Softw 2000; 15:479-88.

Branger F, Braud I, Debionne S, Viallet P, Dehotin J, Henine H, et al. Towards multi-scale integrated hydrological models using the LIQUID framework. Overview of the concepts and first application examples. Environ Model Softw 2010; 25:1672-81.

Calaf M, Meneveau C, Meyers J. Large eddy simulation study of fully developed wind-turbine array boundary layers. Phys Fluids 2010; 22:015110-1-16.

Chatelain P, Backaert S, Winckelmans G, Kern S. Large eddy simulation of wind turbine wakes. Flow Turbul Combust 2013; 91:587-605.

Datla D, Volos HI, Hasan SM, Reed JH, Bose T. Task allocation and scheduling in wireless distributed computing networks. Analog Integr Circ S 2011; 69:341-53.

David O, Ascough II JC, Lloyd W, Green TR, Rojas KW, Leavesley GH, et al. A software engineering perspective on environmental modeling framework design: The Object Modeling System. Environ Model Softw 2013; 39:201-13.

Duff IS, Reid JK. An Implementation of Tarjan's algorithm for the block triangularization of a matrix. ACM T Math Softw 1978; 4(2):137-47.

El Khoury K, Mouawad G, El Hitti G, Nemer M. The component interaction network approach for modeling of complex thermal systems. Adv Eng Softw 2013, 65:149-57.

Ford RW, Riley GD, Bane MK, Armstrong CW, Freeman TL. GCF: a General Coupling Framework. Concurr Comp-Pract E 2003; 00:1-10.

Gabow HN. Path-based depth-first search for strong and biconnected components. Inform Process Lett 2000; 74:107-14.

Geldenhuys J, Valmari A. More efficient on-the-fly LTL verification with Tarjan's algorithm. Theor Comput Sci 2005; 345:60-82.

Gupta M, Curry N, Nylen P, Markocsan N, Vaben R. Design of next generation thermal barrier coatings - experiments and modeling. Surf Coat Tech 2013, 220:20-6.

Hammer M, McLeod D. Database Description with SDM: A semantic database model. ACM T Database Syst 1981; 6(3):351-86.

Heil M. Solvers for large-displacement fluid-structure interaction problems: Segregated versus monolithic approaches. Comput Mech 2008; 43:91-101.

Heimbigner D, McLeod D. A federated architecture for information management. ACM T Database Syst 1985; 3(3):253-78.

Hill C, DeLuca C, Suarez M, Da Silva A. The architecture of the Earth System Modeling Framework. Comput Sci Eng 2004; 6:18-28.
Holzworth DP, Huth NI, de Voil PG. Simplifying environmental model reuse. Environ Model Softw 2010; 25:269-75.

Jaio X, Zheng G, Alexander PA, Campbell MT, Lawlor OS, Norris J, et al. A system integration framework for coupled multiphysics simulations. Eng Comput 2006; 22(3-4):293-309.

Johnson RW, Hansen G, Newman C. The role of data transfer on the selection of a single vs. multiple mesh architecture for tightly coupled multiphysics applications. Appl Math Comput 2011; 217:8943-62.

Joppich W, Kurschner M. MpCCI - a tool for the simulation of coupled applications. Concurr Comp-Pract E 2006; 18:183-92.

Kattke KJ, Braun RJ, Colclasure AM, Goldin G. High-fidelity stack and system modeling for tubular solid oxide fuel cell system design and thermal management. J Power Sources 2011; 196:3790-802.

Klein SA, Alvarado FL. EES Engineering Equation Solver manual. F-Chart Software, Middleton, WI. 1998.

Knapen R, Janssen S, Roosenschoon O, Verweij P, de Winter W, Uiterwijk M, et al. Evaluating OpenMI as a model integration platform across disciplines. Environ Model Softw 2013; 39:274-82.

Kumfert G, Bernholdt DE, Epperly TGW, Kohl JA, McInnes LC, Parker S, et al. How the common component architecture advances computational science. J Phys 2006; 46:479-93.

Lanzafame R, Mauro S, Messina M. Wind turbine CFD modeling using a correlation-based transitional model. Renew Energy 2013; 52:31-39.

Larson J, Jacob R, Ong E. The Model Coupling Toolkit: A new Fortran90 toolkit for building multi physics parallel coupled models. Int J High Perform C 2005; 19:277-92.

Lloyd W, David O, Ascough II JC, Rojas KW, Carlson JR, Leavesley GH, et al. Environmental modeling framework invasiveness: Analysis and implications. Environ Model Softw 2011; 26:1240-50.

Malleron N, Zaoui F, Goutal N, Morel T. On the use of a high-performance framework for efficient model coupling in hydroinformatics. Environ Model Softw 2011; 26:1747-58.

McCorkle DS, Bryden KM. Using the Semantic Web technologies in virtual engineering tools to create extensible interfaces. Virtual Reality 2007; 11:253-260.

Patzak B, Rypl D, Kruis J. MuPIF - A distributed multi-physics integration tool. Adv Eng Softw 2013; 60-61:89-97.

Rezaei R, Chiew TK, Lee SP. An interoperability model for ultra large scale systems. Adv Eng Softw 2014; 67:22-46.

Ryzhakov PB, Rossi R, Idelsohn SR, Onate E. A monolithic Lagragian approach for fluid structure interaction problems. Comput Mech 2010; 46:883-99.

Sharir M. A Strong-Connectivity algorithm and its applications in data flow analysis. Comput Math Appl 1981; 7:67-72.

Sheth AP, Larson JA. Federated database systems for managing distributed, heterogenous, and autonomous databases. ACM Comput Surv 1990; 22(3):183-236.

Singh R, Ahmed M. Blade design and performance testing of a small wind turbine rotor for low wind speed applications. Renew Energy 2013; 50:812-819.

Sonntag SJ, Kaufmann TS, Busen MR, Laumen M, Linde T, Schmitz-Rode T, et al. Simulation of a pulsate total artificial heart: Development of a partitioned fluid-strucutre interaction model. J Fluid Struct 2013; 38:187-204.

Sternel DC, Schafer M, Heck M, Yigit S. Efficiency and accuracy of fluid-structure interaction simulations using an implicit partitioned approach. Comput Mech 2008; 43:103-13.

Tabiei A, Sockalingam S. Multiphysics coupled fluid/thermal/structure simulations for hypersonic reentry vehicles. J Aerosp Eng 2012; 25(2):273-81.

Tarjan R. Depth-first search and linear graph algorithms. SIAM J Comput 1972; 1(2):146-60.

Traverso A, Tucker D, Haynes CL. Preliminary Experimental results of integrated gasification fuel cell operation using hardware simulation. J Eng Gas Turb Power 2012; 134:071701 (10 pages).

Voinov A, Cerco C. Model integration and the role of data. Environ Model Softw 2010; 25:965-9.

Walhorn E, Kolke A, Hubner B, Dinkler D. Fluid-structure coupling within a monolithic model involving free surface flows. Comput Struct 2005; 83:2100-11.

Winkler W, Lorenz H. The design of stationary and mobile solid oxide fuel cell-gas turbine systems. J Power Sources 2002; 105:222-7.

Winkler W, Nehter P, Williams MC, Tucker D, Gemmen R. General fuel cell hybrid synergies and hybrid systems testing status. J Power Sources 2006; 159:656-66.

Xing Y, Moan T. Multi-body modeling and analysis of a planet carrier in a wind turbine gearbox. Wind Energy 2013; 16:1067-1089.

CHAPTER 3

CONCLUSIONS AND FUTURE RESEARCH

This thesis presented a novel application of Tarjan's algorithm as a mechanism for organizing and scheduling the computational workflow in a federated system composed of a large number of heterogeneous, distributed models and databases. Details of Tarjan's algorithm, and its implementation, as well as a demonstration of its application to identify the necessary workflow in two alternative engineering systems, have been presented. Results show the algorithm to be a viable mechanism for accurately identifying the necessary computational workflow in federated systems of models. Additionally the algorithm was shown to be an effective tool for quickly identifying the impact that the addition, substitution, or removal of models or databases has on the computational workflow needed to solve a federated system. This allows different model and database configurations to be considered based on the complexity of the resulting workflow that is needed to accurately perform a large-scale system simulation.

However, Tarjan's algorithm only addresses aspects related to the self-assembly of the computational workflow in a federated system of models and does not address issues related to the actual integration and coupling between the component models and databases. As a result, the construction of a federated system infrastructure or framework that supports component interoperability and information sharing requires additional methods and tools to be utilized and developed including

- Methods and standards for integrating and establishing interoperability between models and databases that are developed, validated, and revised independently from one another. These methods must resolve issues related to model inconsistencies, e.g., differing parameter names

and variable definitions, as well as software inconsistencies, e.g., differing languages, data structures, and protocols.

- A coupling infrastructure that enables the exchange of data between component models and databases that are executed or queried across a network of distributed computational resources. The infrastructure should incorporate tools that support the exchange of data between models characterized by alternative spatial discretizations with respect to structure, resolution, and dimensionality. Additionally, these tools should provide a means of converting or translating data to the appropriate structure or form that is required by the component models to be used.

REFERENCES

Bitz CM, Shell KM, Gent PR, Bailey DA, Danabasoglu G, Armour KC, et al. Climate sensitivity of the Community Climate System Model, Version 4. Journal of Climate 2012; 25:3053-3070.

David O, Ascough II JC, Lloyd W, Green TR, Rojas KW, Leavesley GH, et al. A software engineering perspective on environmental modeling framework design: The Object Modeling System. Environmental Modeling & Simulation 2013; 39:201-213.

Ford RW, Riley GD, Bane MK, Armstrong CW, Freeman TL. GCF: A General Coupling Framework. Concurrency and Computation: Practice and Experience 2003; 00:1-10.

Hill C, DeLuca C, Suarez M, Da Silva A. The architecture of the Earth System Modeling Framework. Computing in Science and Engineering 2004; 6:18-28.

Joppich W, Kurschner M. MpCCI - a tool for the simulation of coupled applications. Concurrency and Computation: Practice and Experience 2006; 18:183-192.

Klein SA, Alvarado FL. EES Engineering Equation Solver manual. F-Chart Software, Middleton, WI. 1998.

Larson J, Jacob R, Ong E. The Model Coupling Toolkit: A new Fortran90 toolkit for building multi physics parallel coupled models. International Journal of High Performance Computing 2005; 19:277-292.

Patzak B, Rypl D, Kruis J. MuPIF - A distributed multi-physics integration tool. Advances in Engineering Software 2013; 60-61:89-97.