Network Science **5** (4): 490–528, 2017. © Cambridge University Press 2017
doi:10.1017/nws.2017.22

The hourglass effect in hierarchical dependency networks

KAESER M. SABRIN and CONSTANTINE DOVROLIS
School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA
(e-mail: `kmsabrin, constantine@gatech.edu`*)*

Abstract

Many hierarchically modular systems are structured in a way that resembles an hourglass. This "hourglass effect" means that the system generates many outputs from many inputs through a relatively small number of intermediate modules that are critical for the operation of the entire system, referred to as the waist of the hourglass. We investigate the hourglass effect in general, not necessarily layered, hierarchical dependency networks. Our analysis focuses on the number of source-to-target dependency paths that traverse each vertex, and it identifies the core of a dependency network as the smallest set of vertices that collectively cover almost all dependency paths. We then examine if a given network exhibits the hourglass property or not, comparing its core size with a "flat" (i.e., non-hierarchical) network that preserves the source dependencies of each target in the original network. As a possible explanation for the hourglass effect, we propose the *Reuse Preference* model that captures the bias of new modules to reuse intermediate modules of similar complexity instead of connecting directly to sources or low complexity modules. We have applied the proposed framework in a diverse set of dependency networks from technological, natural, and information systems, showing that all these networks exhibit the general hourglass property but to a varying degree and with different waist characteristics.

Keywords: *modularity, hierarchy, evolvability, robustness, complexity, centrality, core-periphery networks, hourglass networks, bow-tie networks, dependency networks*

1 Introduction

Complex systems in the natural, technological, and information worlds are often hierarchically modular (Meunier et al., 2010; Parnas et al., 1984; Ravasz et al., 2002; Schilling, 2000). A modular system consists of smaller sub-systems (modules) that, at least in the ideal case, can function independently of whether or how they are connected to other modules: each module receives inputs from the environment or from other modules to perform a certain function (Baldwin & Clark, 2000; Callebaut & Rasskin-Gutman, 2005; Wagner et al., 2007). Modular systems are often also hierarchical, meaning that simpler modules are embedded in, or reused by, modules of higher complexity (Ravasz & Barabási, 2003; Sales-Pardo et al., 2007; Simon, 1991; Yu & Gerstein, 2006). In the technological world, modularity and hierarchy are often viewed as essential principles that provide benefits in terms of design effort (compared to "flat" or "monolithic" designs in which the entire system is a single module), development cost (design a module once, reuse it many times), and agility (upgrade, modify, or replace modules without affecting the entire system)

(Huang & Kusiak, 1998; Fortuna et al., 2011; Myers, 2003). In the natural world, the benefits of modularity and hierarchy are often viewed in terms of evolvability (the ability to adapt and develop novel features can be accomplished with minor modifications in how existing modules are interconnected) (Kashtan & Alon, 2005; Kashtan et al., 2007; Lorenz et al., 2011) and robustness (the ability to maintain a certain function even when there are internal or external perturbations can be accomplished using available modules in different ways) (Kirsten & Hogeweg, 2011; Kitano, 2004; Stelling et al., 2004). In information sciences, hierarchical modularity can improve the stability, quality, and speed of organizational search tasks (such as product or strategy development) (Mihm et al., 2010; Valverde & Solé, 2007). Additionally, it has been shown that both modularity and hierarchy can emerge naturally as long as there is an underlying cost for the connections between different system units (Clune et al., 2013; Mengistu et al., 2016).

It has been observed across several disciplines that hierarchically modular systems are often structured in a way that resembles a bow–tie or hourglass (depending on whether that structure is viewed horizontally or vertically). Informally, this means that the system generates many outputs from many inputs through a relatively small number of intermediate modules, referred to as the "knot" of the bow–tie or the "waist" of the hourglass.[1] This "hourglass effect" has been observed in embryogenesis (Casci, 2011; Quint et al., 2012), in metabolism (Ma & Zeng, 2003; Tanaka et al., 2005; Zhao et al., 2006), in immunology (Beutler, 2004; Oda & Kitano, 2006), in signaling networks (Supper et al., 2009), in vision and cognition (Quiroga et al., 2005; Riesenhuber & Poggio, 1999), in deep neural networks (Hinton & Salakhutdinov, 2006), in computer networking (Akhshabi & Dovrolis, 2011), in manufacturing (Swaminathan et al., 1998), as well as in the context of general core-periphery complex networks (Csermely et al., 2013; Holme, 2005).

The few intermediate modules in the hourglass waist are critical for the operation of the entire system, and so they are also more conserved during the evolution of the system compared to modules that are closer to inputs or outputs (Akhshabi & Dovrolis, 2011; Csete & Doyle, 2004; Domazet-Lošo & Tautz, 2010). These observations have emerged in a wide range of natural, technological, and information disciplines, and so it is interesting to investigate whether the so-called *hourglass effect* has deeper and more general roots that are largely domain-independent.

In this paper, we present a quantitative framework for the investigation of the hourglass effect based on network analysis. First, the organization of a hierarchically modular system is transformed into a *dependency network*, i.e., a Directed Acyclic Graph (DAG) in which vertices represent either individual modules or Strongly Connected Components (SCCs) of interdependent modules. An edge from vertex u to vertex v in a dependency network means that module v depends, in a domain-specific manner, on module u. The input vertices of the dependency network are referred to as *Sources* and the outputs as *Targets*. For example, four dependency networks with different qualitative structures are shown in Figure 1.

[1] The two terms, bow–tie and hourglass, have not been always viewed as synonymous in the network science literature. In particular, the term bow–tie has been applied even to networks for which the knot includes a large fraction of the network's vertices. We discuss the differences between the two terms in Section 7.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

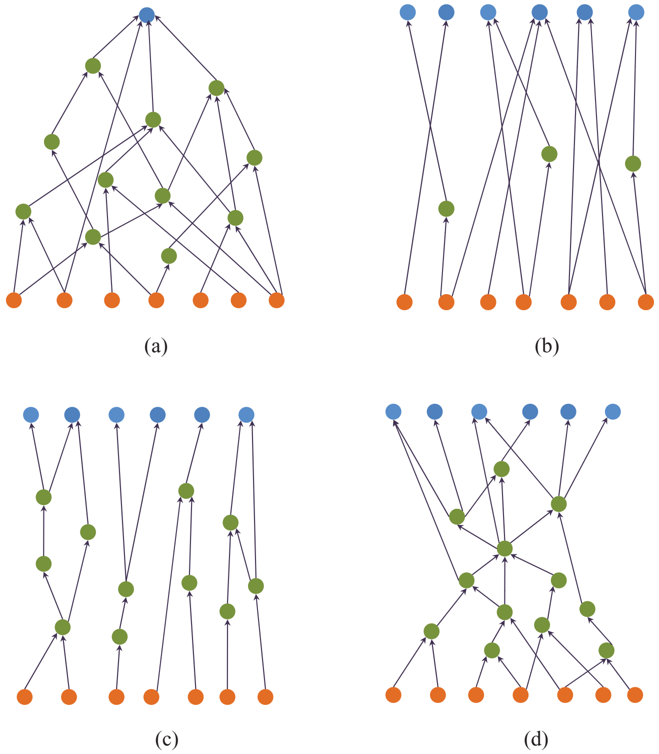


Fig. 1. Four toy examples of dependency networks with qualitatively different structure. The blue vertices are targets, the green are intermediates, and the orange are sources. (a) Pyramid: few targets depend on many sources (or the opposite) (b) Direct: targets often depend directly on sources, few intermediate vertices (c) Decoupled: little reuse of common intermediate vertices across targets (d) Hourglass: almost all source–target dependencies traverse a small number of intermediate vertices.

The importance of each vertex is quantified with a *path centrality* metric, defined as the number of source-to-target dependency paths that traverse that vertex. Based on that metric, we propose an algorithm to identify the *core* of the dependency network, i.e., the smallest set of vertices that collectively cover almost all (say 80–90%) of all source-to-target dependency paths. After computing the core, we can then evaluate if the given network exhibits the hourglass property or not by comparing its core size with a "flat" (i.e., non-hierarchical) network that preserves the source dependencies of each target. We also present a *Reuse Preference (RP)* model for the formation of a dependency network, capturing the bias of new modules to reuse intermediate modules of similar complexity instead of connecting directly to sources or low complexity modules.

We have applied this analysis framework in a diverse set of dependency networks from technological, natural, and information systems: the call-graphs of two software systems, the metabolic networks of two species, and the citation networks of US Supreme Court cases for two legal matters (legality of abortion, and pension disputes). We show that these networks exhibit the hourglass property but to a varying degree. Further we quantify the location of the waist, relative to sources and targets, and the fraction of vertices in "tendril" paths that bypass the waist. The identified vertices at the waist of each network correspond to well-known important modules in the corresponding systems.

Finally, we discuss the connections between the hourglass effect and related concepts, such as the core-periphery structure of many complex networks, the presence of network bottlenecks, and the evolvability and robustness of systems that are hierarchically modular. Together with its theoretical significance, the hourglass effect may also have important practical value, especially in the design of technological systems that operate in uncertain or evolving environments.

In summary, the main contributions of this study are as follows:

1. To show how to transform a directed hierarchical network into a dependency network, and to introduce path centrality as an appropriate metric for the analysis of dependency networks.
2. To formulate the core identification problem as finding the smallest set of vertices that are traversed by a given fraction τ of all source–target paths.
3. To show how to quantify whether a dependency network exhibits the hourglass effect.
4. To propose a probabilistic ‘RP” model of dependency network formation, which illustrates the conditions under which a dependency network exhibits the hourglass effect.
5. To apply this analysis and modeling framework on several dependency networks from different disciplines, showing that they all exhibit the hourglass effect but to a varying extent and with different waist characteristics.
6. To discuss the significance of the hourglass effect in both technology and nature in terms of network bottlenecks, cost, evolvability, and robustness.

2 Dependency networks

Suppose that we are given a directed network $\mathbf{G_0}$ that represents a hierarchically modular system. Each vertex of $\mathbf{G_0}$ corresponds to a system module. An edge from vertex u to vertex v means that module v *depends on* module u. The precise meaning of this dependency relation is domain-specific. In a software system, for instance, modules may represent C functions and edges function calls (function v calls u). In a citation network, the modules may represent research papers or patents and edges some form of information transfer (v cites u). In a mechanical or chemical process, the modules may represent different devices or materials and the edges may represent that the construction (or composition) of a device (material) v requires u as input. Such hierarchical networks are ubiquitous across biology (e.g., food webs), technology (e.g., communication protocol stacks), organizations (e.g., reporting hierarchies), and information systems or social networks (e.g., meme propagation).

In general, the network $\mathbf{G_0}$ may include cyclic relations (referred to as “feedback loops,” “recursive calls,” etc, depending on the context) between two or more vertices. Each set of such interdependent modules can be identified as an SCC (an SCC is a set of vertices, so that every vertex of that set can reach any other vertex of that set). In other words, the modules of an SCC do not have any hierarchical ordering between them; they are all mutually interdependent. To construct an acyclic hierarchical

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

network, we first compute all SCCs of $\mathbf{G_0}$; this can be done in linear time using Tarjan's algorithm (Tarjan, 1972). Then, we replace every SCC of $\mathbf{G_0}$ with a single *super-vertex* that corresponds to the set of vertices in that SCC. Any incoming edge to a vertex of an SCC from a vertex outside that SCC becomes an incoming edge to the corresponding super-vertex; similarly, we construct the outgoing edges of each super-vertex from the outgoing edges of the corresponding SCC. The replacement of SCCs with super-vertices transforms the original network $\mathbf{G_0}$ into a Directed and Acyclic Graph $\mathbf{G}$. We refer to $\mathbf{G}$ as the *dependency network* that corresponds to the original network $\mathbf{G_0}$.

In the rest of the paper, the analysis will be focusing on dependency networks, and the notation will be as follows (Table A.1 lists all our notation). The *dependency network* $\mathbf{G}$ has a set $\mathbf{V}$ of vertices and a set $\mathbf{E}$ of directed edges. The number of vertices and edges is denoted by V and E, respectively.[2] The in-degree of v is denoted by $d_{in}(v)$ and the set of vertices that point to v is denoted by $I(v)$ (*inputs* of v). Similarly, the out-degree of v is denoted by $d_{out}(v)$ and the set of vertices that v points to is denoted by $O(v)$ (*outputs* of v). The *ancestors* of v is the set of vertices that can reach v, while the *descendants* of v is the set of vertices that v can reach.

The set $\mathbf{S}$ of vertices with zero in-degree are referred to as *Sources*, while the set $\mathbf{T}$ vertices with zero out-degree are referred to as *Targets*. The set $\mathbf{M}$ of remaining vertices represent *Intermediate* modules. We have that $\mathbf{V} = \mathbf{S} \cup \mathbf{T} \cup \mathbf{M}$. When plotting dependency networks, we follow the convention that sources appear at the bottom and targets at the top, and so edges have an upward direction.

A path $p(s,t)$ from a source s to a target t is referred to as a *source–target path*, or simply *ST-path*. Focusing on a target t, the set of all ST-paths that terminate at t represent the different "dependency chains" of sources and intermediates that are involved in the formation of t. We focus on all ST-paths that terminate at t instead of all source and intermediate vertices that t depends on. This distinction is important because a source or intermediate vertex v that participates in several ST-paths that terminate at t is more important for t than a vertex u that participates in fewer such ST-paths. For instance, in the context of a citation network the set of ST-paths that terminate at t represents all distinct ways in which the information contained in those source references has been transformed and propagated by intermediate references to finally produce t.

To quantify the topological importance of a vertex in a dependency network we rely on the following metric:

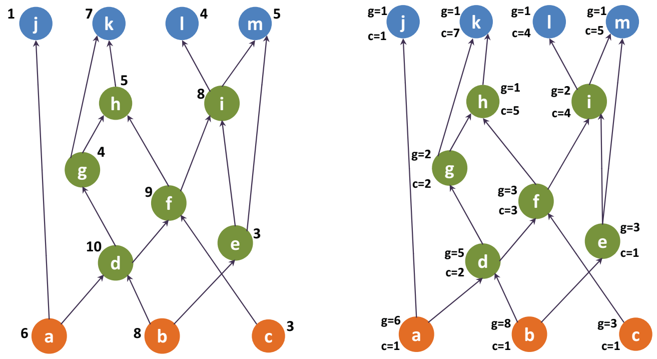
Definition 1 (*Path Centrality*)
The path centrality $P(v)$ of a vertex v is the number of ST-paths that traverse v.

This metric has been also referred to as the *stress* of a vertex (Ishakian et al., 2012). Figure 2(a) illustrates the path centrality of each vertex in a small dependency network.

$P(v)$ can be computed in $O(E)$ time, due to the acyclic nature of dependency networks. Suppose that $P_S(v)$ is the number of paths from any source to v, while $P_T(v)$ is the number of paths from v to any target. $P_S(v)$ can be computed in a bottom-up manner: $P_S(v) = 1$ for all sources and $P_S(v) = \sum_{u \in I(v)} P_S(u)$ for any

[2] We denote the cardinality of a set $\mathbf{X}$ with X.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press



(a) (b)

Fig. 2. The path centrality of each vertex (shown at the left) and the generality (top number) and complexity (bottom number) of each vertex (shown at the right). (a) Path centrality. (b) Complexity & generality. (Color online)

v that is not a source. Similarly, $P_T(v)$ can be computed in a top–down manner: $P_T(v) = 1$ for all targets and $P_T(v) = \sum_{u \in O(v)} P_T(u)$ for any v that is not a target. It is easy to see that the path centrality of v is simply the product of $P_S(v)$ and $P_T(v)$,

$$P(v) = P_S(v) \times P_T(v) \tag{1}$$

The path centrality metric can be also interpreted as follows. The number of paths $P_S(v)$ from sources to v can be thought of as a proxy for v's *complexity*: The more ST-paths terminate at v, the more complex is the formation of v from all its ancestors. Sources have minimal complexity (set to one) because they do not depend on anything else. On the other hand, the number of paths $P_T(v)$ from v to targets can be thought of as a proxy for v's *generality*: The more ST-paths exist from v to the set of targets, the more general or common is the function provided by v in the formation of distinct targets. Targets have minimal generality (set to one) because they are not used to form any other module.

Based on these two definitions, *the path centrality of a vertex v is the product of v's complexity and generality*. This implies that path centrality is a metric that evaluates the topological importance of a vertex in both the upward and downward directions of a dependency network. If the complexity and generality of a vertex are both high, relative to other vertices, that vertex will also have high path centrality. Figure 2(b) illustrates the complexity and generality of each vertex in a small dependency network.

The path centrality metric is more appropriate for identifying important vertices in a dependency network than other centrality metrics. The betweenness or closeness centrality metrics, for instance, only consider the shortest paths between two vertices, and so they would not assign high centrality to a vertex that participates in many (but relatively long) ST-paths. Also, the in-degree or out-degree of a vertex is a local metric and it does not capture the positioning of that vertex in the entire dependency network. The Katz centrality metric, on the other hand, does not distinguish between intermediate vertices and terminal (source or target) vertices, and it penalizes longer dependency paths. Some other centrality metrics, such as pagerank or eigenvector centrality (Newman, 2010), are not appropriate for DAGs.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

3 The core of a dependency network

Intuitively, the *core* of a dependency network can be defined as a subset of vertices that represent the most central modules in the underlying system. One approach would be to rank vertices in terms of path centrality. This approach does not consider, however, that two or more vertices may be traversed by almost the same set of ST-paths. So, even though they may both have high path centrality, including one of them in the core would be sufficient to "cover" those source–target dependencies.

Instead, we define the core of a dependency network based on the solution of an optimization problem: identify the smallest set of vertices that are traversed by almost all ST-paths—namely, a large fraction τ of all ST-paths. We approach this problem in two steps. First, we consider the problem of computing the most central set of k vertices, when k is given, which has already been studied by Ishakian et al. (2012). Then, we use an algorithm for the previous problem to identify the minimum-size core for a given fraction τ of ST-paths.

Definition 2 (*Coverage of ST-paths*)
Let **P** be the set of all ST-paths, and **R** be a set of vertices. $\mathbf{P_R}$ is the subset of **P** that traverses at least one vertex in **R**. The corresponding *path coverage* of **R** is defined as

$$\delta_{\mathbf{R}} = \frac{P_R}{P} \tag{2}$$

Problem 1 (*Cardinality-Constrained Core with Maximum Coverage*—**C**^{3}**MC**)
Given a cardinality k, identify a set $\hat{\mathbf{R}}_k$ of k vertices with maximum path coverage.

$$\hat{\mathbf{R}}_k = \arg\max_{\mathbf{R}\subset\mathbf{V}:|\mathbf{R}|=k} \{\delta_{\mathbf{R}}\} \tag{3}$$

The set $\hat{\mathbf{R}}_k$ may not be unique but $\delta_{\mathbf{R}_k}$, denoted as $\hat{\delta}_k$ in the following, is the same for all optimal solutions.

The **C**^{3}**MC** problem is NP-Complete; a proof is given by Ishakian et al. (2012). However, the objective function of the **C**^{3}**MC** problem is monotonically increasing (obvious) and submodular (proven in the Appendix), and so the following greedy algorithm is guaranteed to produce an $(1-\frac{1}{e})$-approximation of the optimal solution (Nemhauser et al., 1978)—the same algorithm was also used in the work of Ishakian et al.

- Initially, the set $\hat{\mathbf{R}}_k$ is empty.
- In each iteration
  1. Compute the path centrality of all vertices.
  2. Include the vertex with maximum path centrality in the set $\hat{\mathbf{R}}_k$, and remove it from the network (the case of ties is discussed in Section 3.1).
- The algorithm terminates when the set $\hat{\mathbf{R}}_k$ includes k vertices.

The run-time complexity of the path centrality computation is $O(E)$ and, in the worst case, we need to recompute the path centrality of all vertices in every iteration of the algorithm. So, the run-time complexity of the previous greedy algorithm is $O(k\,E)$. In Section 6.3, we show experimentally that the run-time of the core identification algorithm increases quadratically with the number of vertices N (if the average in-degree of non-source vertices remains constant).

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

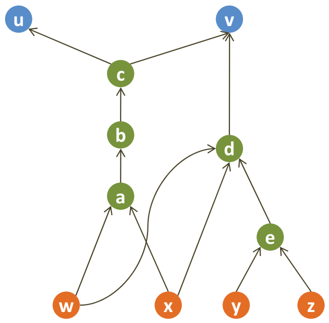


Fig. 3. Vertices a, b, c have equal path centrality and they are traversed by the same set of ST-paths. Vertex d has equal path centrality, but it is traversed by a different set of ST-paths. If there is a tie between the vertices a, b, c, d in an iteration of the core identification algorithm, either the first three vertices will be added in the core as a single Path-Equivalent Set (PES) representing the set $\{a, b, c\}$, or only vertex d will be added in the core. (Color online)

3.1 Path centrality ties

We now describe how the previous greedy algorithm breaks ties among vertices that have the same maximum path centrality. Figure 3 illustrates that there are two different types of ties. First, it may happen that the tied vertices are traversed by exactly the same set of ST-paths. This will be the case, for instance, when those vertices are connected in a linear chain (vertices a, b, and c in Figure 3). Whenever there is a maximum path centrality tie among a set of vertices that are traversed by the same set of ST-paths, we group these vertices as a single *Path-Equivalent Set* (PES). The elements of a PES are equivalent in the sense that they all capture the same set of ST-paths; it is sufficient to include any one of them in the core.

To identify a PES from a set of vertices that have equal path centrality, we pick a vertex u from that set and remove it from the network. Then, we recompute the path centrality of the remaining tied vertices and find those that now have zero path centrality. These vertices, together with u, form a PES. We repeat this process for any remaining tied vertices to identify additional PESs.

Second, there may be a maximum path centrality tie between two or more vertices that are traversed by different sets of ST-paths (for instance, vertices a and d in Figure 3). Ties of this type can be randomly broken, as long as it is sufficient to identify a single core instead of enumerating all possible cores. If it is necessary to identify all possible cores, we can consider separately every possible tie-breaker. This creates a tree of possible execution paths in which each leaf corresponds to a candidate core with k elements.

3.2 The path coverage threshold τ

In practice, the cardinality of the core is not known a priori. Instead, we can set the cardinality of the core heuristically, as follows.

If it was required that the core is traversed by *all* ST-paths, the identification of the core would be equivalent to the well-known minimum-cut problem that can be solved efficiently with a max-flow algorithm (Ravindra et al., 1993). However, requiring that the core covers every single ST-path is a very stringent condition;

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

we have observed that in real dependency networks there are often some direct ST-paths that do not traverse any intermediate vertices or that do not share common intermediate vertices with most other ST-paths.

So, a more pragmatic definition is that the core of a dependency network should cover at least a fraction τ of all ST-paths, where τ is a given *path coverage threshold* that will typically be close to one. To compute the core, we solve the $\mathbf{C^3MC}$ problem iteratively, starting with k=1. The set $\hat{\mathbf{R}}_k$ is computed incrementally by adding one more vertex in $\hat{\mathbf{R}}_{k-1}$, which requires only $O(E)$ additional operations. The algorithm terminates when the path coverage $\hat{\delta}_k$ first exceeds τ.

We use the following notation to represent the core of a dependency network for a given τ: the set of vertices in the core is $\mathbf{C}(\tau)$, the size of the core is $C(\tau)$, and the path coverage of the core is $\delta_{\mathbf{C}(\tau)} \geqslant \tau$. Note that $\mathbf{C}(\tau)$ and $\delta_{\mathbf{C}(\tau)}$ may not be unique if there were ties during the computation of the core. The core size $C(\tau)$, however, is unique.

The incremental increase of the path coverage of $\mathbf{C}$ when v is first included in that core is denoted by $\delta_{\mathbf{C}(v)}$. This metric also represents the *weight* of v in the core.

4 Hourglass dependency networks

4.1 Network flattening and H-score

Informally, the hourglass property of a dependency network can be defined as having a small core, even when the path coverage threshold τ is close to one. To make the previous definition more precise, we can compare the core size $C(\tau)$ of the given dependency network $\mathbf{G}$ with the core size of a derived dependency network that maintains the same set source–target dependencies of $\mathbf{G}$ but that is not an hourglass by construction.

To do so, we create a *flat dependency network* $\mathbf{G_f}$ from $\mathbf{G}$ as follows:

1. $\mathbf{G_f}$ has the same set of source and target vertices as $\mathbf{G}$ but it does not have any intermediate vertices.
2. For every ST-path from a source s to a target t in $\mathbf{G}$, we add a direct edge from s to t in $\mathbf{G_f}$. If there are w edges from s to t in $\mathbf{G_f}$, they can be replaced with a single edge of weight w.

Note that $\mathbf{G_f}$ preserves the source–target dependencies of $\mathbf{G}$: each target in $\mathbf{G_f}$ is constructed based on the same set of "source ingredients" as in $\mathbf{G}$. Additionally, the number of ST-paths in the original dependency network is equal to the number of paths in the weighted flat network (an edge of weight w counts as w paths). However, the dependency paths in $\mathbf{G_f}$ are direct, without forming any intermediate modules that could be reused across different targets. So, by construction, the flat network $\mathbf{G_f}$ cannot have the hourglass property.

Suppose that $C_f(\tau)$ represents the core size of the flat network $\mathbf{G_f}$. The core of $\mathbf{G_f}$ can include a combination of sources and targets, and it cannot be larger than either the set of sources or targets. Additionally, the core of the flat network is larger or equal than the core of the original network (because the core of the flat network also covers at least a fraction τ of the ST-paths of the original network—but the core of the original network may be smaller, because it can also include intermediate

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

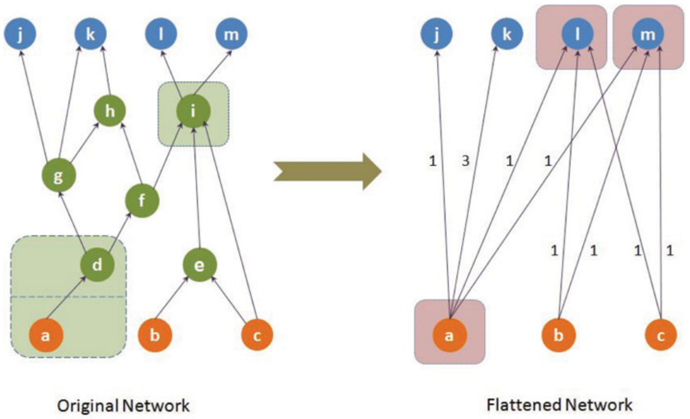


Fig. 4. The weight of an edge in the flattened network represents the number of ST-paths between the corresponding source–target pair in the original dependency network. When the path coverage threshold is τ=90%, the core of the original network (left) is the set $\{\{a,d\},i\}$ ($\{a,d\}$ form a *Path-Equivalent Set* and only one of them should be included in the core). The core of the flattened network (right) for the same τ is $\{a,l,m\}$. The H-score of the original network is $1-\frac{2}{3}=0.33$. (Color online)

vertices together with sources or targets). So,

$$C(\tau) \leqslant C_f(\tau) \leqslant \min\{S,T\} \tag{4}$$

To quantify the extent at which **G** exhibits the hourglass effect, we define the *Hourglass Score*, or *H-score*, as follows:

$$H(\tau) = 1 - \frac{C(\tau)}{C_f(\tau)} \tag{5}$$

Clearly, $0 \leqslant H(\tau) < 1$. The H-score of G is approximately one if the core size of the original network is negligible compared to the the core size of the corresponding flat network. Figure 4 illustrates the definition of this metric.

An ideal hourglass-like network would have a single intermediate vertex that is traversed by every single ST-path (i.e., $C(1)=1$), and a large number of sources and targets none of which originates or terminates, respectively, a large fraction of ST-paths (i.e., a large value of $C_f(1)$). The H-score of this network would be approximately equal to one.

4.2 Coverage and location of a vertex

Another property of an ideal hourglass network is that all vertices that participate in ST-paths should be reachable from the waist, either in the upstream or in the downstream direction. To quantify this property, we define the *core vertex coverage* metric $U_{\mathbf{C}}$, where **C** is the core of the given dependency network:

$$U_{\mathbf{C}} = \frac{\sum_{v \in \mathbf{V_{ST}}} \phi_{\mathbf{C}}(v)}{V_{ST}} \tag{6}$$

where $\mathbf{V_{ST}}$ is the set of vertices that are present in one or more ST-paths, and $\phi_{\mathbf{C}}(v)$ is equal to one when v is a vertex that can reach, or that can be reached from, at

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

least one vertex in the core $\mathbf{C}$; $\phi_{\mathbf{C}}(v)$ is zero otherwise. The metric $1 - U_{\mathbf{C}}$ can be thought of as the fraction of vertices in “tendril” paths that bypass the waist.

We can also associate a *location* with each vertex to capture its relative position in the dependency network between sources and targets. Computing the location of a vertex based on the *topological sorting* of the depending network would not be an appropriate approach in this context because that ordering is determined from the maximum distance of a vertex from the set of sources. Another way to place intermediate vertices between sources and targets is to consider the complexity $P_S(v)$ and generality $P_T(v)$ metrics that were defined in Section 2. Recall that sources have the lowest complexity value (equal to 1), while targets have the lowest generality value (equal to 1). The following equation defines a location metric based on $P_S(v)$ and $P_T(v)$,

$$L(v) = \frac{P_S(v) - 1}{(P_S(v) - 1) + (P_T(v) - 1)} \tag{7}$$

$L(v)$ varies between 0 (for sources) and 1 (for targets). If there is a small number of paths from sources to a vertex v (low complexity) but a large number of paths from v to targets (high complexity), v’s role in the network is more similar to sources than targets, and so its location should be closer to 0 than 1. The opposite is true for vertices that have high complexity but low generality—their location should be closer to 1 than 0.

We can also calculate an *average location for the entire core*. The weight of a core vertex v is proportional to the incremental increase $\delta_{\mathbf{C}(v)}$ of the path coverage of $\mathbf{C}$ when v was first included in that core. So, the average location of the core $\mathbf{C}$ can be defined as the following weighted average of the location of the core vertices,

$$L_{\mathbf{C}} = \frac{\sum_{v \in \mathbf{C}} [\delta_{\mathbf{C}(v)} \, L(v)]}{\sum_{v \in \mathbf{C}} \delta_{\mathbf{C}(v)}} \tag{8}$$

5 Case studies

In this section, we apply the previous analysis framework in six dependency networks from three different disciplines: two call-graphs (software engineering), two metabolic networks (biology, biochemistry), and two citation networks (information science). First, we present the corresponding datasets and the process to convert them into dependency networks. Table 1 shows the basic characteristics of the six dependency networks. Note that the networks vary considerably in terms of density, fraction of source or target vertices, and average ST-path length.

5.1 Datasets and dependency network construction

5.1.1 Call-graphs

Any non-trivial software system is written in a modular and hierarchical manner: “functions” (or “methods”) are defined for distinct processing of tasks, and a function performs its task by calling other, simpler functions. The resulting hierarchy of function calls is referred to as the *call-graph* of that system. The sources of a software system are elementary functions that do not call any other function, functions provided by linked libraries, or functions that communicate directly with

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Table 1. Basic characteristics of analyzed dependency networks. All entries after the first row correspond to the Largest Weakly Connected Component (L-WCC).

	Networks					
	Software Call-graphs		Metabolic Networks		SCotUS citation Networks	
Properties	*OpenSSH v-5.2*	*Apache Math v-3.4*	*Rat*	*Monkey*	*Abortion cases*	*Pension cases*
Vertices	1,300	6,685	843	845	1,502	1,290
Largest component (L-WCC)	99%	95%	64%	61%	100%	95%
Edges	4,583	14,823	612	588	3,266	1,555
Average degree	3.5	2.3	1.2	1.1	2.2	1.3
Targets	22%	35%	24%	25%	20%	24%
Intermediates	45%	32%	56%	55%	17%	11%
Sources	33%	33%	20%	20%	63%	65%
Average ST-path length	10.4	8.8	8.3	8.1	14.1	5.1
Number of super-vertices	3	24	10	9	0	0
Super-vertex size	2.5 ± 0.5	3.2 ± 4.1	9.4 ± 7.4	9.3 ± 7.2	–	–

the primitives provided by the underlying hardware (e.g., device drivers) or the operating system. The targets are various applications or utilities that are called by external entities (the human user, other applications, libraries, and systems, etc).

In the following, we analyze the call-graph of two complex and popular software systems: OpenSSH (version 5.2, written in C) and the Apache Math library (version 3.4, written in Java). The source code for OpenSSH was retrieved in a curated form from an earlier study (Bhattacharya et al., 2012), and the call-graph was constructed using CodeViz Gorman (2015). For the Apache Math library, we use the Java dependency graph extraction tool (Gousios, 2015). We follow the earlier convention that when a function v calls a function u, there is an edge from u to v.

In the case of OpenSSH, we first remove from the call-graph all functions that include the following keywords in their name: *main* (included in many C files for testing different parts of the system independently), *log* and *debug* (used during software development for debugging), *fail*, *fatal*, *error* (generic functions called in case of unexpected errors), and *exit* (program termination). The previous functions have high path centrality mostly because they are called by many other functions but they do not provide any information about the system architecture. Similarly, for the Apache Math library, we remove all *exception handlers* (methods associated with unexpected errors) and the methods of the *Object* class, which is the generic parent class for all Java programs.

The use of *recursive programming* (i.e., one or more functions forming a loop in the call-graph) creates cycles. As discussed in Section 2, each call-graph is transformed into a dependency network by first partitioning the call-graph in a set of SCCs, and then replacing each SCC with a single *super-vertex*. The number and size of the super-vertices in each call-graph are shown in Table 1.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

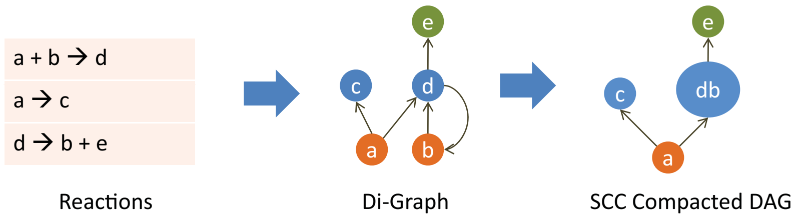


Fig. 5. Construction of a dependency network from a given set of metabolic reactions. (Color online)

5.1.2 *Metabolic networks*

Metabolic networks show how individual chemical reactions in the cell are combined to form the complex pathways associated with functions, such as glycolysis or the biosynthesis of pyrimidine or purine (Palsson, 2015). There are large databases that provide reasonably accurate and complete metabolic networks for many species (Kanehisa & Goto, 2000). The KEGG database, in particular, has been curated for more than a decade to include all known metabolic reactions that conform with the available sequenced genome information (Kanehisa et al., 2014).

In a metabolic network, the products of one chemical reaction can be used as substrates for another chemical reaction. This flow of matter and energy can be represented as a directed network where vertices correspond to metabolites, and an edge from u to v means that there is at least one reaction in which u is a substrate (input) and v is a product (output). Although most chemical reactions are reversible, most metabolic pathways are typically considered to flow in one direction. In the KEGG database, each reaction is associated with the most common direction in a given pathway.

A metabolic network often includes cycles. If two or more metabolites are present in the same cycle, it means that there is no hierarchical ordering between them—they are mutually interdependent. So, as in the case of call-graphs, after constructing the initial metabolic network we replace each SCC with a single *super-vertex* that represents the corresponding set of metabolites in that SCC. Figure 5 shows a small example of how a given set of chemical reactions can be first transformed to a directed network, and then to a dependency network.

In the following, we present results for the metabolic networks of two organisms: *Rattus norvegius* (rat) and *Macaca mulatta* (monkey). Both datsets were retrieved from the 2014 KEGG (Kanehisa et al., 2014) database. For each metabolic network, we only analyze the Largest Weakly Connected Component (L-WCC). The smaller connected components correspond to distinct pathways that do not have any common metabolites with the L-WCC.

5.1.3 *SCotUS citation network*

Dependency networks can also capture the flow of information, knowledge or legal precedent in research publications, patents, court cases, and so on. Here, we focus on the citation network of court judgments made by the Supreme Court of the United States (SCotUS). We rely on a dataset collected by Fowler et al. (2007), Fowler & Jeon (2008) that includes all SCotUS cases between 1754 and 2002.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Judicial decisions often leverage the precedent of earlier judgments to support their arguments, forming a directed citation network. Following our earlier convention, if a court case v refers to a previously settled case u, there is an edge from u to v. In the case of citation networks, the hierarchy of the dependency network implies a temporal ordering between connected vertices: if there is a path from u to v, u appeared before v.

In this paper, we focus on two legal matters that have been the subject of many SCotUS cases: the *legality of abortion* and various *pension (or benefits) disputes*. First, we use the Legal Information Institute (Legal Information Institute, 2015) of Cornell University's online legal library to find the set of SCotUS cases that focus on each of these two matters. Suppose that $\mathbf{X}$ is the set of SCotUS cases that are related to one of these two matters. We construct the corresponding citation network by including all cases in $\mathbf{X}$ as well as any other SCotUS case that directly cites, or is directly cited by, a case in $\mathbf{X}$. This expansion of the citation network with cases that do not belong in $\mathbf{X}$ is important because the SCotUS decisions about a certain matter may depend on, or they may have influenced, decisions regarding other legal matters.

The selection of sources and targets in a citation network may appear as somewhat arbitrary. This is an important issue that deserves further discussion. The sources and targets of a dependency network should be selected based on the scope, or boundaries, of the underlying system we aim to understand. Considering only parts of that system, or merging it with other systems, can mislead the analysis. For instance, if we want to identify the most significant publications associated with a specific problem in network science, say community detection, it would be incorrect to only consider the citation network of publications that focus on spectral graph partitioning, and it would also be incorrect to consider every publication that relates broadly to graphs or networks. We admit, however, that in some cases, it may be challenging to uniquely identify the scope, or boundaries, of a given dependency network; this is a problem that deserves further study.

The two citation networks are acyclic, and so we do not create any super-vertices.

5.2 Analysis of dependency networks

Figure 6 shows the maximum path coverage $\hat{\delta}_k$ that results from solving the $\mathbf{C^3MC}$ problem iteratively, for increasing values of k, until $\hat{\delta}_k$ approaches 100%. Note that all six curves are strongly concave and that almost all ST-paths are covered with a very small number of vertices relative to the size of each network.

Figure 7 examines the effect of the path coverage threshold τ on the resulting H-score of each network. As expected, if we require that the core covers a higher fraction of ST-paths the core will need to be larger. The two citation networks strongly exhibit the hourglass effect, as their H-score remains close to 0.9 even when the core covers 90–95% of all ST-paths. The two metabolic networks can be also viewed as hourglass networks, with an H-score of about 0.85, but only as long as the core covers less than 80–85% of all ST-paths. Their core would need to be significantly larger to cover the remaining paths. The two call-graphs are structured differently and they exhibit a weaker hourglass effect: OpenSSH's H-score varies erratically between 0.6 and 0.8 depending on τ, while the Apache Math library's

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

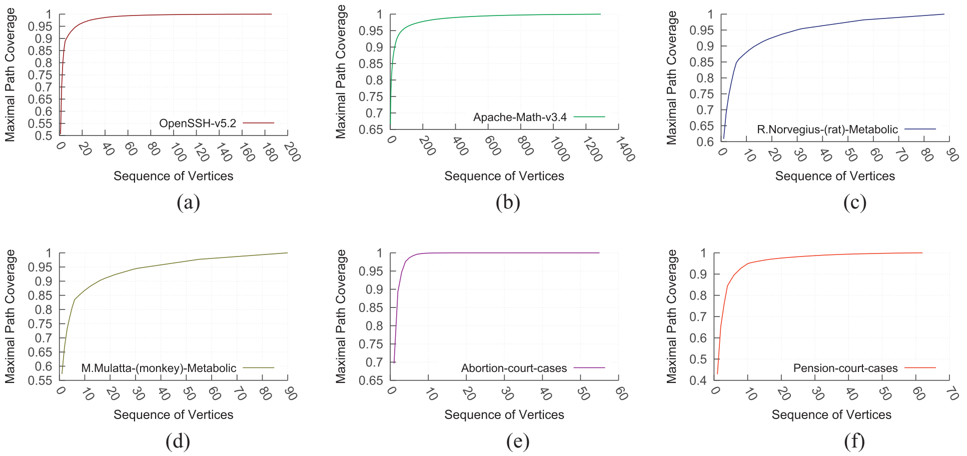


Fig. 6. The maximum path coverage $\hat{\delta}_k$ as a function of k for the six dependency networks. (a) OpenSSH-v5.2. (b) Apache-Math-v3.4. (c) Rat metabolic. (d) Monkey metabolic. (e) Abortion cases. (f) Pension cases. (Color online)

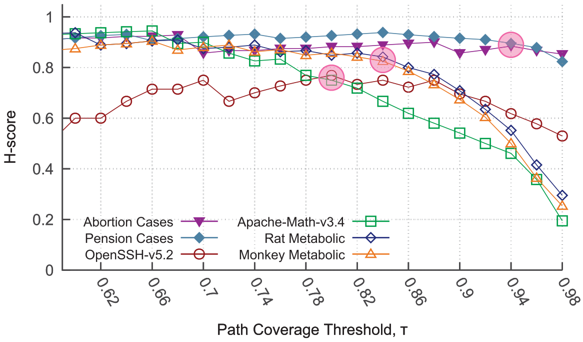


Fig. 7. Effect of τ on H-score. The value of τ that we use in the rest of the analysis for each network is shown with a magnified symbol.

H-score quickly drops below 0.8 when the core needs to cover more than 80% of all ST-paths.

Based on Figure 7, in the rest of the analysis we set τ at the largest value before the H-score shows a significant drop. After selecting the same value for each network type, we set τ as follows: call-graphs $\tau = 80\%$, metabolic networks $\tau = 85\%$, and citation networks $\tau = 95\%$.

Table 2 summarizes the key properties of the core of each dependency network. The size of the core $\mathbf{C}$ varies from 0.1% to 1% of the network size V. In all six networks, we identified only one core (no ties); some vertices in the core of the metabolic networks and of the Apache Math network are super-vertices. For the selected values of τ, the H-score is higher than 0.75 in all networks.

Even though the core of each network is quite small, relative to the total number of vertices, none of these networks can be described as an "ideal hourglass." This is shown both in terms of the H-score in Figure 7 and by the core vertex coverage: there is a significant fraction of vertices (about 20–80%, depending on the network) in ST-paths that bypass the core ("tendril paths"). The fraction $1 - \tau$ of ST-paths

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Table 2. Properties of the identified core for each dependency network.

	Networks					
	Software Call-graphs		Metabolic Nets		SCotUS citation Nets	
Core Properties	*OpenSSH v-5.2*	*Apache Math v-3.4*	*Rat*	*Monkey*	*Abortion Cases*	*Pension Cases*
Path coverage threshold τ	0.8	0.8	0.85	0.85	0.95	0.95
Core size C	3	9	7	8	4	11
C/V	0.002	0.001	0.01	0.02	0.002	0.008
H-score	0.77	0.75	0.82	0.81	0.86	0.89
Number of distinct cores	1	1	1	1	1	1
SCCs in core	0	1	3	3	0	0
Number of PES in core	0	2	1	2	0	0
Core vertex coverage	0.35	0.21	0.53	0.57	0.82	0.48
Average core location	0.50	0.12	0.45	0.44	0.74	0.24

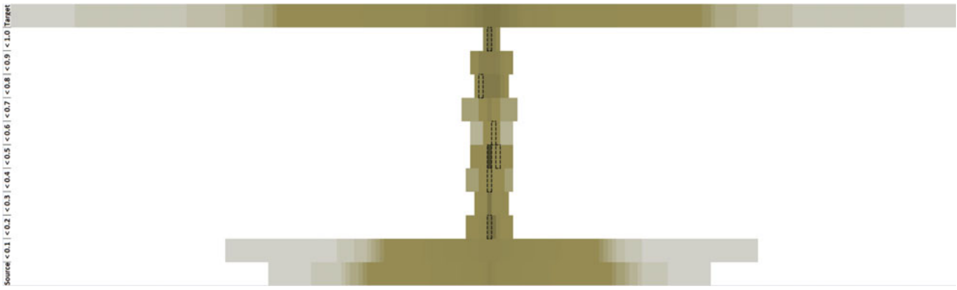


Fig. 8. A visualization of the Rat metabolic network that places vertices in the vertical direction based on their location metric. Specifically, we discretize the location metric in 12 bins (the lowest bin for sources, the highest for targets, and the 10 intermediate bins for intermediate vertices with each bin accounting for 1/10 of the 0−1 range). The path centrality of each vertex is represented by its color (darker for higher path centrality). Vertices with higher centrality are placed closer to the vertical midline. The core nodes are represented by dotted rectangles. (Color online)

that bypass the core traverse at least two vertices each (a source and a target). When these tendrils traverse several intermediate vertices, however, the core vertex coverage can be significantly lower than $2\times(1-\tau)$. As shown in the modeling results of the next section (see Figure 11(a)), such low values of the core vertex coverage can be expected when each vertex has a bias to depend on vertices of similar complexity with itself (rather than to depend directly on sources or low complexity vertices) but where that bias is not strong enough to generate an "ideal hourglass" in which a small set of intermediate vertices is traversed by all ST-paths.

Figure 8 is a visualization of the Rat metabolic dependency network that places vertices in the vertical direction based on their location metric (see caption for more details about this visualization). Note that the highest path centrality vertices tend to be at intermediate locations—but some of the sources and targets in this network also have high path centrality. Also, about half of the core vertices are located close to the center of the network (location=0.5), while the rest are closer to

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

sources or targets. The path centrality and the weight of each core vertex are shown in Table A.5. The location of the core vertices varies significantly across different networks. Similar visualizations for the other five networks are given in Figure A.1.

5.3 Which are the vertices at the waist of the hourglass?

The complete list of core vertices for each dependency network, together with a short description, the path centrality and the weight of core vertex, are given in the *Appendix*. Here, we comment on the qualitative properties of the waist vertices for each network.

The three vertices at the core of the OpenSSH call-graph are shown in Table A.2. They are functions to send and receive network packets, and to execute Unix shell commands. This is not surprising given that OpenSSH is a communication-oriented utility that can be used as a secure remote terminal, among other applications.

The Apache Math library has a core with nine vertices, listed in Table A.3. These methods cover floating point arithmetic operations, matrix decomposition, vector computations, and the "constructors" of some classes related to mathematical and geometric objects.

The vertices at the waist of the two metabolic networks are shown in Tables A.5 and A.7. In biochemistry, the following 12 *precursors* are often considered as the most important metabolites, providing an interface between the different catabolic pathways with the various biosynthesis pathways: Glucose-6-Phosphate, Fructose-6-Phosphate, Glycerone Phosphate, Glyceraldehyde 3-Phosphate, Phosphenol Pyruvate, Pyruvate, Ribose-5-Phosphate, Erythrose-4-Phosphate, Acetyl-CoA, a-ketogluterate, Oxalocetate, and Succinyl-CoA (Smolke, 2009; Alberts et al., 2002; Tanaka et al., 2005); it is not clear, however, if these precursors are equally important for every species or if the previous list should include additional metabolites. In the case of Rat metabolic network, the identified waist includes eight of the previous precursors, plus few more key compounds for the synthesis of enzymes, lipids, fatty acids, etc. In the case of the Monkey metabolic network, the waist includes seven precursors. Several waist vertices are the same with those in the Rat (or similar, in the case of SCCs or PES).

The vertices at the waist of the two citation networks are shown in Tables A.9 and A.10. The Cornell Legal Information Institute (CLII) lists several *landmark* SCotUS cases for every major legal matter in the United States (Legal Information Institute, 2015). This classification of cases as landmarks is based on input from legal experts. All court cases that appear in the waist of the Abortion network are also listed as landmarks by CLII. In the Pension network, five out of the seven waist vertices are also listed as landmarks by CLII.

6 A model of dependency network formation

What determines whether a dependency network will exhibit the hourglass property or not? Let us think about this question in the context of Lego-like toys, in which a vertex v corresponds to a Lego module and its incoming edges show which simpler Lego modules are required to put v together. The sources correspond to the given elementary building blocks and the targets correspond to the final

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

objects we want to construct. One extreme approach is to create every object only from the elementary blocks, without reusing any intermediate modules that have been previously constructed. Another approach is to reuse as much as possible intermediate modules, expecting that this will require less work. In practice, of course, the design approach is always somewhere in the middle, with more complex intermediate modules constructed from simpler intermediate modules as well as elementary blocks.

To understand the implications of this “preference for reuse,” we present here a simple, probabilistic model for the gradual formation of a dependency network. The model focuses on how each new vertex selects its incoming edges among the set of vertices that have been previously constructed. Through a single *reuse parameter* α, the model generates dependency networks in which every new vertex depends on either mostly sources (leading to flat, non-hourglass networks) or on the more recently constructed intermediate vertices (resulting in hourglass networks), or anything in between.

We refer to the following model as *RP-model.* There are V vertices that consist of S sources, M intermediates and T targets. The vertices arrive in the network, or they are created, sequentially or in batches, as follows. First, all sources are created at the same time; they represent the elementary modules of the underlying system. Then, the intermediate vertices are created sequentially (the case of batch arrivals is considered in Section 6.1). Suppose that v is the m’th intermediate vertex that has arrived in the network, with $1 \leqslant m \leqslant M$. We assign vertex v to *rank-0*, and the previously created $m-1$ intermediate vertices to *rank-1* through *rank-(m-1)* (in order of arrival—the oldest intermediate vertex always has *rank-(m-1)*). The S sources are randomly given ranks m through $m+(S-1)$. Note that the ranking changes every time a new vertex is added. The T targets are created in a batch at the end of the network formation process, and they are given the same rank (*rank-0*).

Suppose that we are given the in-degree $d_{in}(v)$ of v. The origin of every incoming edge to v is determined as follows. When the m’th intermediate vertex v is created, we select the vertices it will depend on probabilistically. In the following, we use the Zipf distribution (but other statistical models could also be used). Specifically, the probability that v will have an incoming edge from a vertex u at *rank-r* is given by

$$\mathrm{Prob}[(u,v) \in \mathbf{E}] = \frac{r^{-\alpha}}{\sum_{i=1}^{S+m-1} i^{-\alpha}}, \quad 1 \leqslant r \leqslant S+m-1 \tag{9}$$

The incoming edges to the T target vertices are determined in the same way; note that a target will never by connected to another target because all targets are added in the same batch, having *rank-0.* Additionally, we artificially exclude the possibility of multi-edges.

When $\alpha = 0$ the newly created vertex v selects dependencies uniformly across all earlier vertices. As α increases above zero, v has a preference for more recently constructed vertices, increasing the level of reuse in the dependency network. On the other hand, as α decreases below zero, v has a preference for older vertices, i.e., closer to the sources, decreasing the level of reuse.

Figure 9 shows three small dependency networks constructed using the RP-model for three different values of α. When $\alpha = -1$, almost all ST-paths are directly connecting sources to targets (little reuse of intermediate vertices), and

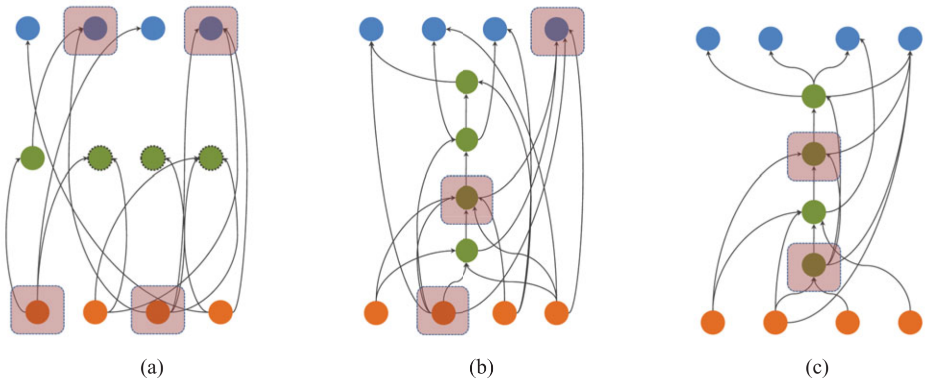


Fig. 9. Three dependency networks generated by the RP-model for different values of α ($V = 12$, $S = T = M = 4$, $d_{in} = 1 + \text{Poisson}(2)$, $\alpha = \{-1, 0, 1\}$, and $\tau = 0.90$. The sources are shown in orange, the targets in blue, and the intermediates in green. Vertices that do not belong to any ST-path are shown as dotted. The core vertices for each network are shown in boxes. (a) $\alpha = -1$, H-score = 0.0. (b) $\alpha = 0$, H-score = 0.25. (c) $\alpha = +1$, H-score = 0.5.

most intermediate vertices are not used in the construction of any target (shown as dotted). The core consists of a combination of sources and targets, and it is relatively large (in this example, equal to the number of sources or targets). On the other hand, when $\alpha = +1$, the preference to connect to higher complexity vertices leads to longer dependency paths. A small number of intermediate vertices are traversed by a large fraction of ST-paths, just based on chance, and so those vertices end up with much higher path centrality than most other vertices. The core of such dependency networks is then small, relative to the number of sources or targets, and those networks have high H-score.

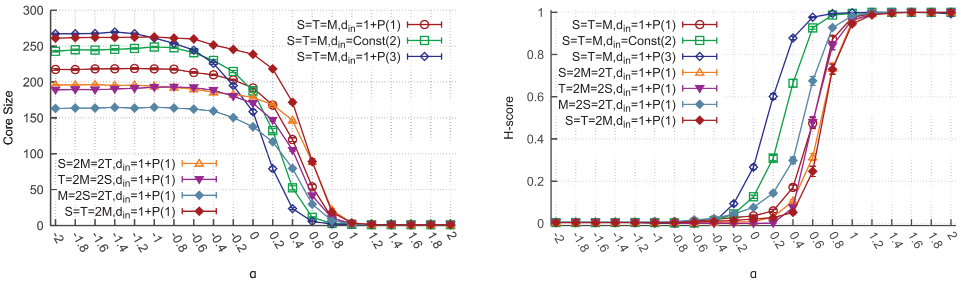
In the following, we illustrate the behavior of the RP-model with computational experiments. All networks have V=1,000 vertices but we vary the proportion of sources, targets, and intermediate vertices. The path coverage threshold τ is set to 90%, unless stated otherwise. The in-degree of each vertex is either constant (denoted as "d_{in} = Const(x)") or set to 1 + Poisson(x), where x is the mean of a Poisson distribution (denoted as "$d_{in} = 1 + \text{P(x)}$"). All results are based on 100 simulation runs, and they are reported with 95% confidence intervals.

Figures 10 and 11 show the effect of the reuse parameter α on the core size $C(\tau)$, the H-score $H(\tau)$, the core vertex coverage $U_{\mathbf{C}}$, and the average core location $L_{\mathbf{C}}$. Each graph shows results for seven sets of network parameters, varying the proportion of sources, targets, intermediates, and the in-degree values and distribution. For example, the label $S = 2M = 2T$ means that $S = 500$ and $M = T = 250$ (so that $V = 1,000$).

Let us first focus on negative values of α:

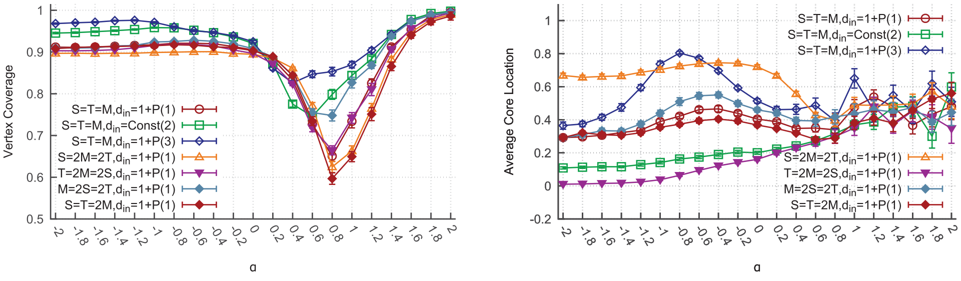
a. As α decreases below zero, it becomes more likely that targets connect directly to sources (see the "direct" network of Figure 1(b) or Figure 9(a)). Most intermediate vertices are not included in any ST-path, their path centrality is close to zero, and so they are not included in the core. Instead, the core consists of mostly a combination of sources and targets. To cover the large

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press



(a) (b)

Fig. 10. Effect of α on the core size and H-score metric. (a) Core size. (b) H-score.



(a) (b)

Fig. 11. Effect of α on the core vertex coverage and average core location metrics. (a) Core vertex coverage. (b) Average core location.

fraction τ (90%) of these direct ST-paths, however, the core needs to include many vertices. For instance, in the scenario $M = 2S = 2T$ the core has about 160 vertices, while $\min\{S, T\} = 250$. The higher the average in-degree is, the larger the core needs to be (to cover the increased number of ST-paths).

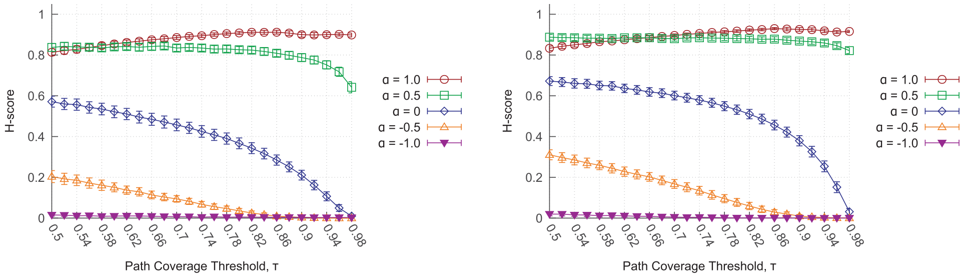
b. The corresponding flat dependency network is similar to the original network in terms of how sources and targets are directly connected, and so it has approximately the same core size; this is why the H-score is close to zero.

c. The core vertex coverage is close to one for the following reason: if all ST-paths are direct connections between sources and targets and the core covers a fraction τ of these paths, the core vertex coverage will be at least $1 - 2(1 - \tau)$ because every non-covered ST-path contributes at most two non-covered vertices.

d. The location of the core varies significantly with the network parameters because the core consists of mostly sources and targets. So, if the core consists mostly of sources (as in the $T = 2M = 2S$ scenario) the core location moves closer to zero, while if the core includes mostly targets (as in the $S = 2M = 2T$ scenario) the core location moves closer to one.

Let us now focus on positive values of α:

a. As α increases above zero, each target or intermediate vertex prefers to connect to vertices that are close to it in the given hierarchy (see Figure 9(c)). So, the ST-paths become longer and some intermediate vertices get to be traversed by



(a)

(b)

Fig. 12. Effect of path coverage threshold τ on H-score, for different values of α. Network parameters: $S = T = 200$ and $M = 600$ ($V = 1{,}000$). (a) $d_{in} = 1 + \text{Poisson}(2)$. (b) $d_{in} = 1 + \text{Poisson}(3)$. (Color online)

a larger fraction of ST-paths (just based on chance). Vertices with high path centrality tend to form the core of the dependency network, and their number gradually drops as α increases.

b. The core of the flat network, on the other hand, is much larger, as in the case of negative α, and so the corresponding H-score approaches its maximum value (one) as α increases.

The transition point, from $H(\tau) \approx 0$ to $H(\tau) \approx 1$, shifts toward lower values of α as the density of the network increases (see scenario $d_{in} = 1 + \text{P}(3)$) because the likelihood that few intermediate vertices will acquire much higher path centrality increases.

c. The core vertex coverage curves follow an interesting pattern: as α increases from negative values to positive values, $U_{\mathbf{C}}$ first decreases and then increases. During the transition from a flat network ($H(\tau) \approx 0$) to an hourglass-like network ($H(\tau) \approx 1$), it is common for ST-paths to traverse one or more intermediate vertices that are not traversed by many other ST-paths (see the "decoupled" network of Figure 1(c)). So, in that transition range, the fraction $1 - \tau$ of ST-paths that are not covered by the core account for more than $2(1 - \tau)$ non-covered vertices (because they include one or more intermediate vertices). As α further increases, the core is traversed by an increasing fraction of ST-paths, eventually covering almost all ST-paths, and so also covering almost all vertices that appear in ST-paths.

d. The location of the hourglass waist is gradually converging toward the middle of the dependency network, i.e., $L_{\mathbf{C}} \approx 0.5$. We should note that the location of a PES is, by definition, equal to the median location of the vertices in that set. So, one reason that the location of the waist converges to 0.5 as α increases is that the waist in that regime often includes a large PES with many intermediate vertices that have locations between 0 and 1.

Finally, Figure 12 shows the effect of the path coverage threshold τ on the H-score for few different values of α. When the reuse parameter α is close to one (or higher), the H-score is almost one, largely independent of τ, meaning that the hourglass property is robustly established.[3] When α is negative or even

[3] The slight increase of the H-score with τ, when $\alpha = 1$, is because the core size of the flat network increases faster than the core size of the original network, as τ increases.

close to zero, on the other hand, the H-score is typically less than 50% and so those networks clearly do not have the hourglass property, independent of the selection of τ. For intermediate values of α, the H-score depends on the selection of τ and on other network parameters, such as the average in-degree.

6.1 Fitting the RP-model to a given dependency network

We now describe how to parameterize the RP-model so that it produces random networks G that have approximately the same H-score with a given dependency network G'. We also compare these synthetic networks with G' in terms of the path centrality distribution, the out-degree distribution, and some more network metrics that are relevant to dependency networks.

Given G', we can easily identify its set of sources and targets. A synthetic network G will have the same set of sources, targets, and intermediate vertices. Since it may not be possible to identify a global ordering between the intermediate vertices, we place the vertices of G in layers based on the topological sorting of G', as follows. First, all sources of G' are placed at layer-0 of G. Then, recursively, we place at layer i of G those intermediate vertices of G' that depend on at least one vertex of layer $i - 1$ (for $i > 0$). Finally, the targets of G' are placed at the top layer of G (independent of the layer of their incoming edges). This layered representation of G gives a partial ordering relation between vertices: the vertices of layer i are supposed to arrive (or to be created) as a batch, and they do not depend on each other.

The in-degree $d_{in}(v)$ of each non-source vertex v in G is the same with G'. To generate the specific inputs of v, we identify the set of ancestors $A(v)$ of v in G' – v depends directly or indirectly on these vertices. When a vertex v is created at layer i, it can receive incoming edges only from vertices in $A(v)$. The selection of inputs of v among the vertices in $A(v)$ is performed probabilistically based on Equation (9). The only difference with the original RP-model is that vertices of $A(v)$ that belong to the same layer have the same rank, and so the same probability of being connected to v.[4]

To parameterize the RP-model, we estimate the value of the RP exponent α so that the synthetic networks G have an H-score that is approximately the same with that of G'. To do so, we generate 100 synthetic networks G for each value of α and compute the average H-score of that sample—the optimal value of α is the value that gives the minimum difference from the H-score of G'.

Table 3 shows the estimate of α for each dependency network of Section 5. The average H-score of those synthetic networks is within 10% of the H-score of G'.

The RP-model generates ST-paths with a similar average length as in the given dependency networks. The average length of the dependency paths is an important metric, as it represents the typical number of intermediate vertices between a source and a target. The synthetic networks are often similar with the given dependency networks in terms of the core vertex coverage and the average core location but there are also some significant deviations (the model overestimates the core vertex coverage of the call graphs and metabolic networks, and it does not predict correctly the location of the core of the SCOTUS abortion cases network).

[4] In the original RP-model, vertices are created sequentially and so each layer (other than the boundary layers of sources and targets) includes only one vertex.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Table 3. Fitting the RP-model to the six dependency networks of Section 5: we show the α estimate, the average ST-path length, the core size (for the same value of τ as in the analysis of the original networks—see Table 2), the core vertex coverage U_C, and the average core location L_C. The corresponding values for the original dependency networks are shown in parentheses.

	Networks					
	Software Call-graphs		Metabolic nets		SCotUS citation Nets	
	OpenSSH v-5.2	*Apache Math v-3.4*	*Rat*	*Monkey*	*Abortion cases*	*Pension cases*
α estimate	1.1	2.3	2.4	2.5	2.7	2.3
Core size	3 ± 1 (3)	4 ± 1 (9)	4 ± 4 (7)	4 ± 2 (8)	5 ± 0.5 (4)	8 ± 1 (11)
H-score	0.69 ± 0.05 (0.77)	0.78 ± 0.03 (0.75)	0.76 ± 0.07 (0.82)	0.75 ± 0.04 (0.81)	0.78 ± 0.02 (0.86)	0.87 ± 0.01 (0.89)
ST-path length	8.8 ± 0.8 (10.4)	9.5 ± 0.4 (8.8)	11.3 ± 3.5 (8.3)	12.3 ± 3.8 (8.1)	14.3 ± 0.5 (14.1)	6.4 ± 0.3 (5.1)
Core vertex coverage	0.28 ± 0.04 (0.35)	0.12 ± 0.01 (0.21)	0.32 ± 0.07 (0.53)	0.3 ± 0.07 (0.57)	0.85 ± 0.1 (0.82)	0.45 ± 0.05 (0.48)
Average core location	0.51 ± 0.3 (0.50)	0.05 ± 0.02 (0.12)	0.36 ± 0.12 (0.45)	0.3 ± 0.12 (0.44)	0.2 ± 0.15 (0.74)	0.29 ± 0.18 (0.24)

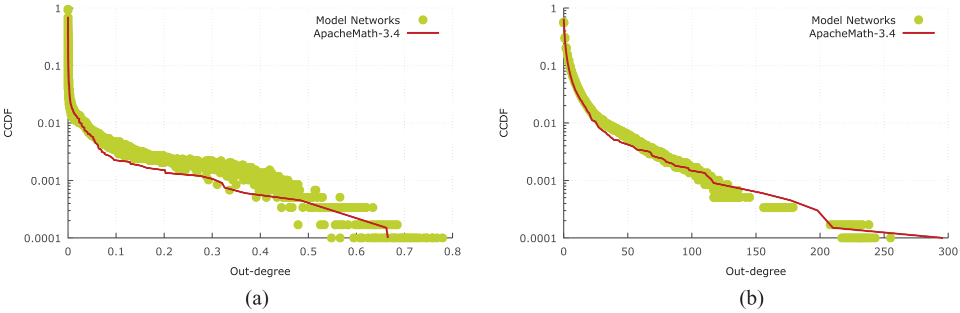


Fig. 13. Comparing path centrality and out degree distribution of a real network with model generated synthetic networks. (a) Path centrality distribution. (b) Out-degree distribution. (Color online)

Figure 13 shows the path centrality and the out-degree distributions for the Apache Math call-graph and for an ensemble of 100 synthetically generated networks by the RP-model, as described earlier. Similar results for the five other dependency networks are shown in the Supplementary Material (see Figure A.2). Even though there is significant variability between members of the ensemble (both distributions are highly skewed), the model is able to generate distributions of path centrality and out-degree that encompass the main mass of the empirical distributions of G'.

6.2 Comparison with another dependency network model

We are not aware of any other model that can generate hourglass dependency networks. However, there is a well-known class of models that can generate growing dependency networks based on variations of the "edge-copying" mechanism (Kleinberg et al., 1999; Krapivsky & Redner, 2005). The simplest instance of the edge-copying model is as follows: a new vertex v depends with probability β on a randomly chosen vertex u, and with probability $1 - \beta$ on a randomly chosen vertex w that u depends on, i.e., v copies an incoming edge of u (Kleinberg et al., 1999). If these dependencies are represented with directed edges from u (or w) to v, the out-degree distribution follows a power-law with exponent $-\frac{2-\beta}{1-\beta}$ (Kumar et al., 2000). For $\beta < 1/2$, the edge-copying model generates scale-free networks and some vertices are expected to be hubs. An important question is as follows: Can the edge-copying model generate hourglass dependency networks, at least for some values of β? And if so, is it that the hubs appear at the waist of the hourglass network?

We follow the same process as in Section 6.1 to fit the edge-copying model in a given dependency network, i.e., the number of sources, targets and intermediate vertices, the (partial) ordering with which the vertices are created, and the in-degree of each vertex are as in the given dependency network. One special case that we need to address is the following: What if a vertex v selects to copy (with probability $1 - \beta$) an incoming edge of a source u? Since sources do not have incoming edges, we assume that v should receive an incoming edge from u instead. Also, we do not allow multi-edges.

Figure 14 shows the results of fitting the edge-copying model in the OpenSSH call-graph, Rat metabolic network, and Abortion cases citation network: the y-axis shows the H-score (average and 95% confidence interval) of 100 synthetic networks generated for different values of the parameter β. Note that the H-score is close to zero throughout the range of β, meaning that the edge-copying model is *not* able to generate hourglass networks. As β approaches one, each new vertex depends on randomly chosen existing vertices—which is also what happens in the RP-model when $\alpha = 0$; we have already seen that such networks do not exhibit the hourglass effect. As β approaches zero, the edge-copying mechanism is applied more often and this causes the emergence of hubs. These hubs, however, tend to be sources because the latter are created first, and so their number of outgoing edges increases faster than other vertices (Barabási & Albert, 1999). As a result, most targets are connected directly to sources generating dependency networks with very short ST-paths, a large fraction of disconnected intermediate vertices, and a core that consists of almost all source vertices—consequently, the H-score of such networks is close to zero.

Comparing the RP-model with the edge-copying model, we note that the former is able to generate hourglass networks, when α is close to one or higher, because the preference to connect to higher complexity vertices leads to longer dependency paths, and thus to the emergence of few intermediate vertices with much higher path centrality.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

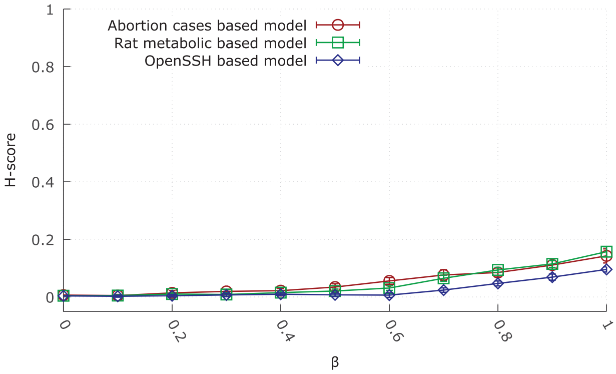


Fig. 14. H-score of synthetic networks generated by the edge-copying model. The network parameters (number of sources, targets, partial ordering of vertices, and in-degree of each vertex) are set based on the three empirical dependency networks shown in the legend. (Color online)

6.3 Run-time analysis of core identification algorithm

We can also use the RP-model to examine the scalability of the core identification algorithm. We created synthetic dependency networks of different sizes, for three different values of α (–0.5, 0, 0.5). The proportion of sources and targets remains constant (25% each), while the in-degree of each non-source vertex is $1+\text{Poisson}(2)$.

As discussed in Section 3, the core identification greedy algorithm has a run-time complexity of $O(k\,E)$, where k is the size of the core and E is the number of network edges. In the dependency networks we construct, E increases proportionally with the number of vertices N, i.e., $E = d_{in}(N - S)$, where d_{in} is the average in-degree for non-source vertices and S is the number of sources. The relation between k and N is not something we could derive analytically, and it certainly depends on α and the path coverage threshold τ.

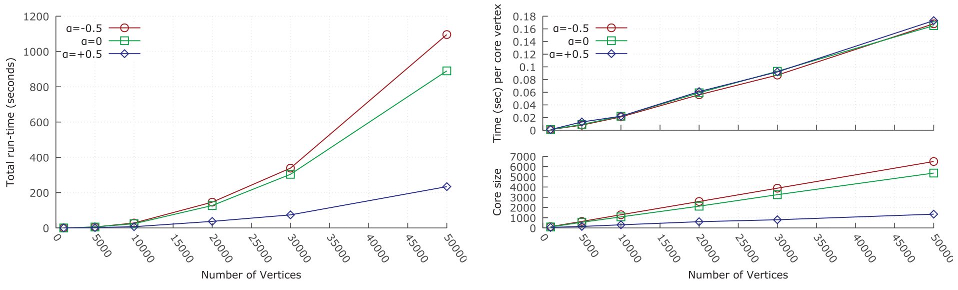
Figure 15 shows the run-time, the run-time per core vertex, and the core size k as a function of N, for τ=0.90. Note that k increases almost linearly with N for all values of α we consider. Consequently, the total run-time becomes the product of two linear functions of N, and so it *increases quadratically with the network size*. As expected, non-hourglass networks (e.g., when $\alpha = -0.5$) have a larger core, and so they require more computation than hourglass networks.

7 Related work

The terms “hourglass” and “bow–tie” are often mentioned informally in the network science literature and in other disciplines—their precise meaning and whether the two terms are synonymous is not always clear however.

The term bow–tie, in particular, always refers to directed (but not necessarily acyclic) networks. It first appeared in the context of the WWW graph, after the 2,000 study of Broder et al. (2000). The “knot” of that bow–tie was described as the largest SCC in the graph, which included about 25% of the network’s vertices. Similarly, the term bow–tie has been also used in the context of metabolic networks (Ma & Zeng, 2003; Zhao et al., 2006). Since then, several directed networks have been described as bow–ties, as long as there is a central SCC with incoming edges

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press



(a)

(b)

Fig. 15. Run-time analysis of the core identification algorithm using networks generated with the RP-model. The run-time increases quadratically with the network size N. The experiments were run on an Intel-2.5 GHz dual-core processor with 6 GB of memory. (a) Run-time. (b) Core-size and run-time per core vertex. (Color online)

from a large input component and outgoing edges to a large output component (Newman et al., 2002; Capocci et al., 2006; Saito et al., 2007; Vitali et al., 2011; Easley & Kleinberg, 2010). In other words, the term "bow–tie network" refers mostly to a visual representation of directed networks based on the previous decomposition of vertices into four sets: an input component, a core (the largest SCC), an output component, and any other vertices that are not in the previous three components (referred to as "tendrils" and disconnected components). There is no requirement that the vertices in the knot of the bow–tie account for only a small fraction of the network size. There is also no requirement that the vertices in the knot are highly central, for any definition of centrality.

Hourglass networks, on the other hand, are typically directed and acyclic graphs, and the vertices at the hourglass waist need to be a small fraction of the total number of vertices. Further, the few vertices at the waist are present in almost all source–target paths in the network, and so they can be thought of as functionally very important for the underlying system (Akhshabi & Dovrolis, 2011; Csete & Doyle, 2004).

The three most relevant studies about the hourglass effect focused on the special case of layered and acyclic directed networks in which edges can only exist between successive layers (Akhshabi & Dovrolis, 2011; Akhshabi et al., 2014; Friedlander et al., 2015). In those studies, the hourglass effect is defined in terms of the number of vertices at each layer, and a network is referred to as an hourglass if the width of the intermediate layers is much smaller relative to the width of the input and output layers. The first study (Akhshabi & Dovrolis, 2011) proposed an evolutionary model (called EvoArch) for the emergence of the hourglass effect in computer networking protocol stacks; EvoArch captures the creation and competition between modules that perform similar functions and it may be also applicable in other layered technological systems. The second study (Akhshabi et al., 2014) made the case that the topological structure of developmental regulatory networks (namely that the specificity of regulatory interactions increases during embryogenesis) is sufficient for the emergence of the hourglass effect in that context. The third study (Friedlander et al., 2015) showed that a layered and directed network can evolve to a bow–tie structure if the relation between inputs and outputs can be represented with a

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

rank-deficient matrix, and if the mutations in the intensity (weights) of module interactions (network edges) can be modeled as products by a random number (rather than sums).

The previous models and analysis frameworks, however, are not applicable in more general dependency networks. Even if we artificially place vertices in layers based on topological sorting (i.e., sources are placed at the bottom layer, and each vertex is placed at the lowest possible layer so that all its incoming edges are from vertices of lower layers), edges can traverse more than one layer, and targets can appear at different layers. Additionally, a general dependency network may include cyclic dependencies and SCCs of interdependent modules. So, those studies do not define the hourglass property in general hierarchically modular systems and they do not show how to identify their waist.

In the context of DAGs, a relevant prior study is Ishakian et al. (2012). That work had a different focus (not related to the hourglass effect or modeling hierarchical systems) but it considered the same centrality metric (referred to as #P centrality) that we also use, and it analyzed the computational complexity of the problem of identifying the k vertices that have, collectively as a group, the largest #P centrality (referred to as the C^3MC problem in our work).

Another relevant study is the *BowTieBuilder* algorithm (Supper et al., 2009). That work examined to what extent signal transduction pathways follow the bow–tie structure proposing a centrality metric ("bow–tie score") for each protein in the network, based on the number of sources and targets that are connected with paths traversing that protein. The knot of the bow–tie was defined as the set of proteins with maximal bow–tie score.

The "morphospace" of all possible hierarchical networks was investigated in Corominas-Murtra et al. (2013). The three dimensions of the considered morphospace in that study are "treeness," "feedforwardness," and "orderability." A large number of networks, mostly metabolic, neuronal, and language, are shown to fall in the part of the morphospace that corresponds to hourglass or bow–tie networks.

8 Discussion—significance of the hourglass effect

The hourglass effect is significant for several reasons. One of them is that the modules at the waist of the network create a "bottleneck" in the flow of information from sources (or inputs) to targets (or outputs). Such bottleneck network effects have been studied in the literature under different names. For instance, the term "core–periphery networks" has been broadly used in network science to refer to various static and dynamic topological properties (e.g., rich-club effect, onion-like networks) that result from a dense, cohesive core that is connected to sparsely connected peripheral vertices (but not necessarily organized in an acyclic input–output hierarchy) (Borgatti & Everett, 2000; Csermely et al., 2013; Rombach et al., 2014). Bottlenecks have been also observed in gene regulatory networks (Bhardwaj et al., 2010), in protein networks (Yu et al., 2007), in general evolutionary models (Jain & Krishna, 2002), among many other domains. The methodology, we have presented in this paper for the identification of the core and for the quantification

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

of the hourglass effect can serve as a unified approach for the study of bottleneck network phenomena in a wide range of disciplines.

Why do so many networks in nature and technology exhibit the hourglass effect? Is there a single underlying explanation or are there different mechanisms through which a hierarchical network can acquire this property? In technological networks, the reuse of existing modules has economic benefits in terms of design and implementation cost, and so it may be that the hourglass property results "by design" (Yan et al., 2010). In natural networks, on the other hand, are there similar costs that an evolutionary process gradually reduces or should we look for a completely different explanation? The model of Friedlander et al. (2015) captures how a realistic evolutionary process searches for the network that results in a desired input-output (linear) transformation. A more recent work (Siyari et al., 2016) proposes an optimization-based framework, modeling sources as characters, and targets as strings, that creates the given targets through the construction and reuse of intermediate substrings. The proposed RP-model offers a different, probably more general explanation for the hourglass effect: a dependency network with multiple sources and targets exhibits the hourglass effect when each vertex tends to depend on vertices of similar complexity (instead of connecting directly to sources or vertices of much lower complexity). This "preference for reuse" tends to create deep hierarchies in which a small set of intermediate vertices is traversed by most dependency paths. The RP-model is probabilistic, and so it is not possible to predict which specific intermediate vertices will emerge at the waist. In practice, we expect that the vertices at the waist will correspond to modules that are both highly general (meaning that their function is needed, directly or indirectly, by many targets) and highly complex (meaning that to provide that function, those modules need to utilize, directly or indirectly, the functionality of many sources).

The hourglass effect is also significant for the evolvability and robustness of hierarchically modular systems. Intuitively, the hourglass effect should allow a system to accommodate frequent changes in its sources or targets (i.e., to be able to evolve as the environment changes) because the few modules at the waist "decouple" the large number of sources from the large number of targets. If there is a change in the inputs (sources), the outputs do not need to be modified as long as the modules at the waist can still function properly. Similarly, if there is need for a new target, it may be much easier (or cheaper) to construct it reusing the modules at the waist rather than directly relying on sources. This is related to the notion of "constraints that de-constrain," introduced by Kirschner and Gerhart in the context of biological development and evolvability (Kirschner & Gerhart, 1998). At the same time, however, the presence of these critical modules at the waist (the "constraints") limit the space of all possible outputs that the system can generate ("phenotype space"), at least for a given maximum cost. The mechanisms through which the hourglass effect can improve evolvability but also limit the phenotype space is an important issue not only for natural systems but also for evolving technological systems (Rexford & Dovrolis, 2010).

Finally, understanding the implications of the hourglass effect for the cost, robustness, and evolvability of designed or technological systems can also have significant practical applications. In engineering, the primary focus is typically on optimality rather than on evolvability or robustness (e.g., design the minimum

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

cost electronic circuit that can perform a given logic function). Such system-wide cost minimizations may appear attractive at first but they typically lead to non-hierarchical (monolithic) designs that are hard to test, evolve, or operate in the presence of failures. On the other hand, hierarchical design often lacks a systematic framework and the tools that would allow the designer to automatically identify, given a set of inputs and a set of outputs, the intermediate modules that would be most reusable. This becomes an even harder problem when we consider that most technological systems need to evolve as the desired functionalities (outputs) and conditions (inputs) often change over time. One approach, which has not been pursued so far to the extent of our knowledge, is to start the design process from the waist, rather than bottom-up or top-down: first design a relatively small number of modules of intermediate complexity that will form the waist of the dependency network. Then, construct these modules based on the inputs, and in parallel construct the outputs based on these modules at the waist. Of course the key challenge in this approach is to develop algorithms and tools that can automatically identify those few central building blocks that will form the hourglass waist from the system specifications.

Acknowledgments

This research was supported by the National Science Foundation (NSF award CNS-1319549). We are also grateful to Saamer Akhshabi (Georgia Tech), Payam Siyari (Georgia Tech), Mathieu Nassif (McGill), Prof. Bistra Dilkina (Georgia Tech), and Prof. Martin Robillard (McGill) for their valuable help and input. We are also grateful to the anonymous reviewers for their thoughtful and constructive comments.

Dedicated to the memory of Saamer Akhshabi.

References

Akhshabi, S., & Dovrolis, C. (2011). The evolution of layered protocol stacks leads to an hourglass-shaped architecture. In *ACM SIGCOMM Computer Communication Review*, vol. 41, ACM, pp. 206–217.

Akhshabi, S., Sarda, S., Dovrolis, C., & Yi, S. (2014). An explanatory evo-devo model for the developmental hourglass. *F1000research*, **3**(156). doi: 10.12688/f1000research.4583.2.

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., & Walter, P. (2002). *Molecular biology of the cell* (4th ed.). New York: Garland Science.

Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity*. Vol. 1. Cambridge, MA: MIT Press.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**(5439), 509–512.

Beutler, B. (2004). Inferences, questions and possibilities in toll-like receptor signalling. *Nature*, **430**(6996), 257–263.

Bhardwaj, N., Yan, K.-K., & Gerstein, M. B. (2010). Analysis of diverse regulatory networks in a hierarchical context shows consistent tendencies for collaboration in the middle levels. *Proceedings of the National Academy of Sciences*, **107**(15), 6841–6846.

Bhattacharya, P., Iliofotou, M., Neamtiu, I., & Faloutsos, M. (2012). Graph-based analysis and prediction for software evolution. In *Proceedings of the 2012 International Conference on Software Engineering*, IEEE Press, pp. 419–429.

Borgatti, S. P., & Everett, M. G. (2000). Models of core/periphery structures. *Social Networks*, **21**(4), 375–395.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., . . . Wiener, J. (2000). Graph structure in the web. *Computer Networks*, **33**(1), 309–320.

Callebaut, W., & Rasskin-Gutman, D. (2005). *Modularity: Understanding the Development and Evolution of Natural Complex Systems.* Cambridge, MA: MIT Press.

Capocci, A., Servedio, V., Colaiori, F., Buriol, L. S., Donato, D., Leonardi, S., & Caldarelli, G. (2006). Preferential attachment in the growth of social networks: The Internet encyclopedia Wikipedia. *Physical Review E*, **74**(3), 036116.

Casci, T. (2011). Development: Hourglass theory gets molecular approval. *Nature Reviews Genetics*, **12**(2), 76–76.

Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society of London B: Biological Sciences*, **280**(1755), 20122863.

Corominas-Murtra, B., Goñi, J., Solé, R., & Rodríguez-Caso, C. (2013). On the origins of hierarchy in complex networks. *Proceedings of the National Academy of Sciences*, **110**(33), 13316–13321.

Csermely, P., London, A., Wu, L.-Y., & Uzzi, B. (2013). Structure and dynamics of core/periphery networks. *Journal of Complex Networks*, **1**(2), 93–123.

Csete, M., & Doyle, J. (2004). Bow ties, metabolism and disease. *TRENDS in Biotechnology*, **22**(9), 446–450.

Domazet-Lošo, T., & Tautz, D. (2010). A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. *Nature*, **468**(7325), 815–818.

Easley, D., & Kleinberg, J. (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* Cambridge: Cambridge University Press.

Fortuna, M. A., Bonachela, J. A., & Levin, S. A. (2011). Evolution of a modular software network. *Proceedings of the National Academy of Sciences*, **108**(50), 19985–19989.

Fowler, J. H., & Jeon, S. (2008). The authority of supreme court precedent. *Social Networks*, **30**(1), 16–30.

Fowler, J. H., Johnson, T. R., Spriggs, J. F., Jeon, S., & Wahlbeck, P. J. (2007). Network analysis and the law: Measuring the legal importance of precedents at the us supreme court. *Political Analysis*, **15**(3), 324–346.

Friedlander, T., Mayo, A. E., Tlusty, T., & Alon, U. (2015). Evolution of bow-tie architectures in biology. *PLoS Computational Biology*, **11**(3), e1004055.

Gorman, M. (2015). *Codeviz: A callgraph visualiser.* Available at: `http://www.csn.ul.ie/∼mel/projects/codeviz/`.

Gousios, G. (2015). *java-callgraph: Java call graph utilities.* Available at: `https://github.com/gousiosg/java-callgraph`.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, **313**(5786), 504–507.

Holme, P. (2005). Core-periphery organization of complex networks. *Physical Review E*, **72**(4), 046111.

Huang, C.-C., & Kusiak, A. (1998). Modularity in design of products and systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **28**(1), 66–77.

Ishakian, V., Erdös, D., Terzi, E., & Bestavros, A. (2012). A framework for the evaluation and management of network centrality. In *Sdm*, SIAM, pp. 427–438.

Jain, S., & Krishna, S. (2002). Large extinctions in an evolutionary model: The role of innovation and keystone species. *Proceedings of the National Academy of Sciences*, **99**(4), 2055–2060.

Kanehisa, M., & Goto, S. (2000). Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, **28**(1), 27–30.

Kanehisa, M., Goto, S., Sato, Y., Kawashima, M., Furumichi, M., & Tanabe, M. (2014). Data, information, knowledge and principle: Back to metabolism in kegg. *Nucleic Acids Research*, **42**(D1), D199–D205.

Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, **102**(39), 13773–13778.

Kashtan, N., Noor, E., & Alon, U. (2007). Varying environments can speed up evolution. *Proceedings of the National Academy of Sciences*, **104**(34), 13711–13716.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Kirschner, M., & Gerhart, J. (1998). Evolvability. *Proceedings of the National Academy of Sciences*, **95**(15), 8420–8427.

Kirsten, H., & Hogeweg, P. (2011). Evolution of networks for body plan patterning; interplay of modularity, robustness and evolvability. *PLoS Computational Biology*, **7**(10), e1002208.

Kitano, H. (2004). Biological robustness. *Nature Reviews Genetics*, **5**(11), 826–837.

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., & Tomkins, A. S. (1999). The web as a graph: Measurements, models, and methods. In *International Computing and Combinatorics Conference*. Lecture Notes in Computer Science book series, vol. 1627. Springer, pp. 1–17.

Krapivsky, P. L., & Redner, S. (2005). Network growth by copying. *Physical Review E*, **71**(3), 036118.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., & Upfal, E. (2000). The web as a graph. In *Proceedings of the 19th ACM sigmod-sigact-sigart Symposium on Principles of Database Systems*, ACM.

Legal Information Institute, Cornell University Law School. (2015). *Historic Supreme Court Decisions*. Available at: `https://www.law.cornell.edu/`. Accessed: 2015-10-30.

Lorenz, D. M., Jeng, A., & Deem, M. W. (2011). The emergence of modularity in biological systems. *Physics of Life Reviews*, **8**(2), 129–160.

Ma, H.-W., & Zeng, A.-P. (2003). The connectivity structure, giant strong component and centrality of metabolic networks. *Bioinformatics*, **19**(11), 1423–1430.

Mengistu, H., Huizinga, J., Mouret, J. B., & Clune, J. (2016). The evolutionary origins of hierarchy. *PLoS Computational Biology*, **12**(6), e1004829.

Meunier, D., Lambiotte, R., & Bullmore, E. T. (2010). Modular and hierarchically modular organization of brain networks. *Frontiers in Neuroscience*, **4**, 200. doi: 10.3389/fnins.2010.00200.

Mihm, J., Loch, C. H., Wilkinson, D., & Huberman, B. A. (2010). Hierarchical structure and search in complex organizations. *Management Science*, **56**(5), 831–848.

Myers, C. R. (2003). Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical Review E*, **68**(4), 046116.

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, **14**(1), 265–294.

Newman, M. (2010). *Networks: An Introduction*. Oxford, UK: Oxford University Press.

Newman, M. E. J., Forrest, S., & Balthrop, J. (2002). Email networks and the spread of computer viruses. *Physical Review E*, **66**(3), 035101.

Oda, K., & Kitano, H. (2006). A comprehensive map of the toll-like receptor signaling network. *Molecular Systems Biology*, **2**, 2006.0015. doi: 10.1038/msb4100057.

Palsson, B. O. (2015). *Systems Biology*. Cambridge: Cambridge University Press.

Parnas, D. L., Clements, P. C., & Weiss, D. M. (1984). The modular structure of complex systems. In *Proceedings of the 7th International Conference on Software Engineering*, IEEE Press, pp. 408–417.

Quint, M., Drost, H.-G., Gabel, A., Ullrich, K. K., Bönn, M., & Grosse, I. (2012). A transcriptomic hourglass in plant embryogenesis. *Nature*, **490**(7418), 98–101.

Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., & Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, **435**(7045), 1102–1107.

Ravasz, E., & Barabási, A.-L. (2003). Hierarchical organization in complex networks. *Physical Review E*, **67**(2), 026112.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., & Barabási, A.-L. (2002). Hierarchical organization of modularity in metabolic networks. *Science*, **297**(5586), 1551–1555.

Ravindra, K. A., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: Theory, algorithms, and applications*. Englewood Cliffs, NJ: Prentice Hall.

Rexford, J., & Dovrolis, C. (2010). Future internet architecture: clean-slate versus evolutionary research. *Communications of the ACM*, **53**(9), 36–40.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, **2**(11), 1019–1025.

Rombach, M. P., Porter, M. A., Fowler, J. H., & Mucha, P. J. (2014). Core-periphery structure in networks. *SIAM Journal on Applied Mathematics*, **74**(1), 167–190.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Saito, H., Toyoda, M., Kitsuregawa, M., & Aihara, K. (2007). A large-scale study of link spam detection by graph algorithms. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, ACM, pp. 45–48.

Sales-Pardo, M., Guimera, R., Moreira, A. A., & Amaral, L. A. N. (2007). Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, **104**(39), 15224–15229.

Schilling, M. A. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review*, **25**(2), 312–334.

Simon, H. A. (1991). The architecture of complexity. In *Facets of Systems Science*. New York: Springer US, pp. 457–476.

Siyari, P., Dilkina, B., & Dovrolis, C. (2016). Lexis: An optimization framework for discovering the hierarchical structure of sequential data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16, New York, NY, USA: ACM, pp. 1185–1194.

Smolke, C. (2009). *The metabolic pathway engineering handbook: Tools and applications*, Vol. 2. Boca Raton, FL: CRC Press.

Stelling, J., Sauer, U., Szallasi, Z., Doyle, F., & Doyle, J. (2004). Robustness of cellular functions. *Cell*, **118**(6), 675–685.

Supper, J., Spangenberg, L., Planatscher, H., Dräger, A., Schröder, A., & Zell, A. (2009). Bowtiebuilder: Modeling signal transduction pathways. *BMC Systems Biology*, **3**(1), 1.

Swaminathan, J. M., Smith, S. F., & Sadeh, N. M. (1998). Modeling supply chain dynamics: A multiagent approach. *Decision Sciences*, **29**(3), 607–632.

Tanaka, R., Csete, M., & Doyle, J. (2005). Highly optimised global organisation of metabolic networks. *IEE Proceedings-Systems Biology*, **152**(4), 179–184.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, **1**(2), 146–160.

Valverde, S., & Solé, R. V. (2007). Self-organization versus hierarchy in open-source social networks. *Physical Review E*, **76**(4), 046118.

Vitali, S., Glattfelder, J. B., & Battiston, S. (2011). The network of global corporate control. *PloS One*, **6**(10), e25995.

Wagner, G. P., Pavlicev, M., & Cheverud, J. M. (2007). The road to modularity. *Nature Reviews Genetics*, **8**(12), 921–931.

Yan, K.-K., Fang, G., Bhardwaj, N., Alexander, R. P., & Gerstein, M. (2010). Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks. *Proceedings of the National Academy of Sciences*, **107**(20), 9186–9191.

Yu, H., & Gerstein, M. (2006). Genomic analysis of the hierarchical structure of regulatory networks. *Proceedings of the National Academy of Sciences*, **103**(40), 14724–14731.

Yu, H., Kim, P. M., Sprecher, E., Trifonov, V., & Gerstein, M. (2007). The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Computational Biology*, **3**(4), e59.

Zhao, J., Yu, H., Luo, Jian-Hua, Cao, Zhi-Wei, & Li, Yi-Xue. (2006). Hierarchical modularity of nested bow-ties in metabolic networks. *BMC Bioinformatics*, **7**(1), 386.

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Appendix—Supplementary material

Notation

Table A.1. List of symbols.

Symbol	Description
X	The cardinality of a set **X**
$\mathbf{G_0}$	The original directed network (may include cycles)
$\mathbf{G}$	Dependency network (directed and acyclic, by construction)
$\mathbf{V}$	Set of vertices
$\mathbf{E}$	Set of edges
$\mathbf{S}$	Set of sources
$\mathbf{T}$	Set of targets
$\mathbf{M}$	Set of intermediates
$\mathbf{I}(v)$	Set of vertices with edges to v—inputs of v
$\mathbf{O}(v)$	Set of vertices with edges from v—outputs of v
$d_{in}(v)$	In-degree of v
$d_{out}(v)$	Out-degree of v
$p(s,t)$	A path from a source s to a target t— an ST-path
$P(v)$	Path centrality of v
$P_S(v)$	Number of paths from sources to v (complexity of v)
$P_T(v)$	Number of paths from v to targets (generality of v)
$\mathbf{P}$	Set of all ST-paths
$\mathbf{P_R}$	Set of ST-paths that traverse a set **R** of vertices
$\delta_{\mathbf{R}}$	Path coverage of $\mathbf{R}(= P_R/P)$
$\hat{\mathbf{R}}_k$	Set of k vertices with maximum path coverage
$\hat{\delta}_k$	Path coverage of $\hat{\mathbf{R}}_k$
τ	Path coverage threshold
$\mathbf{C}(\tau)$	A core (there may be more than one) for a given path coverage threshold τ
$\delta_{\mathbf{C}(\tau)}$	The path coverage of $\mathbf{C}(\tau)$ (may be more than τ)
$\mathbf{G_f}$	The flat network that corresponds to **G**
$\mathbf{C_f}(\tau)$	the core of the flat network $\mathbf{G_f}$ for the threshold τ
$H(\tau)$	H-score for the threshold τ
$U_{\mathbf{C}}$	Core vertex coverage of core **C**
$\mathbf{V_{ST}}$	Set of vertices in at least one ST-path
$\phi_{\mathbf{C}}(v)$	Indicator variable that vertex $v \in \mathbf{V_{ST}}$ is reachable from, or can reach, at least one vertex In core **C**
$L(v)$	Location of vertex v
$L_{\mathbf{C}}$	Average location of core **C**
$\delta_{\mathbf{C}}(v)$	Incremental path coverage when vertex v is added in core **C** (also referred to as the “weight” of core vertex v)
α	Reuse preference exponent
d_{in}	Average in-degree
β	Parameter of edge-copying model

Submodularity of the $\mathbf{C^3MC}$ *objective function*

Lemma 1
The objective function of the $\mathbf{C^3MC}$ problem is submodular, i.e.,

$$\delta_{\mathbf{X}\cup\{v\}} - \delta_{\mathbf{X}} \geqslant \delta_{\mathbf{Y}\cup\{v\}} - \delta_{\mathbf{Y}} \tag{A.1}$$

for any $\mathbf{X}, \mathbf{Y}$ such that $\mathbf{X} \subseteq \mathbf{Y} \subseteq \mathbf{V}$ and for any vertex v in $\mathbf{V}$.

Proof
The function $\delta_{\mathbf{R}}$ is non-negative and non-decreasing.

- Case 1: Consider all ST-paths that traverse v but not any vertex in $\mathbf{Y}$. These paths do not traverse any vertex in $\mathbf{X}$ either. So, the increase in the coverage of $\mathbf{X}$ and $\mathbf{Y}$ will be the same when we add v in both sets.
- Case 2: Consider all ST-paths that traverse v as well as one or more vertices of $\mathbf{Y}$ but not any vertex of $\mathbf{X}$. Such ST-paths increase the coverage of only $\mathbf{X} \cup v$.
- Case 3: Consider all ST-paths that traverse v as well as one or more vertices of $\mathbf{X}$. These paths are already included in the coverage of both $\mathbf{X}$ and $\mathbf{Y}$, and so they will not cause any further coverage increase by including v in $\mathbf{X}$ and $\mathbf{Y}$.

These three cases account for all ST-paths traversing v, and we have shown that the submodularity condition is satisfied in all of them. □

Vertices at the waist of each dependency network

Table A.2. The waist of the OpenSSH-v5.2 call-graph network.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
Packet_send	0.50	0.50	*Wrapper function for formatting and sending TCP packet.*
Packet_read_seqnr	0.37	0.20	*Function to return type of received packet.*
Do_exec	0.42	0.10	*Function responsible for spawning a sub-shell as part of session creation*

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press

Table A.3. The waist of the Apache-Math-v3.4 call-graph network. SCCs are listed in Table A.4.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.60	0.60	*Methods from the decimal floating point library class.*
Vector3D:init	0.08	0.06	*Initializer for base class implementing vectors in a three-dimensional space.*
DerivativeStructure:init	0.10	0.04	*Initializer for base class that is the workhorse of differentiation library.*
FastMath:abs	0.04	0.03	*Faster math library's absolute value computing method.*
EigenDecomposition:init	0.10	0.02	*Initializer for the class handling eigen decomposition of a real matrix.*
BigFraction:init	0.05	0.02	*Initializer for base class representing a rational number without any overflow.*
MatrixUtils: createRealMatrix	0.09	0.01	*Method to create and initialize a real-valued matrix from given data.*
Line:init	0.04	0.01	*Initializer for the three dimensional geometric line Java class.*
IntervalsSet:iterator	0.008	0.01	*Iterator for traversing a set of one dimensional geometric intervals.*

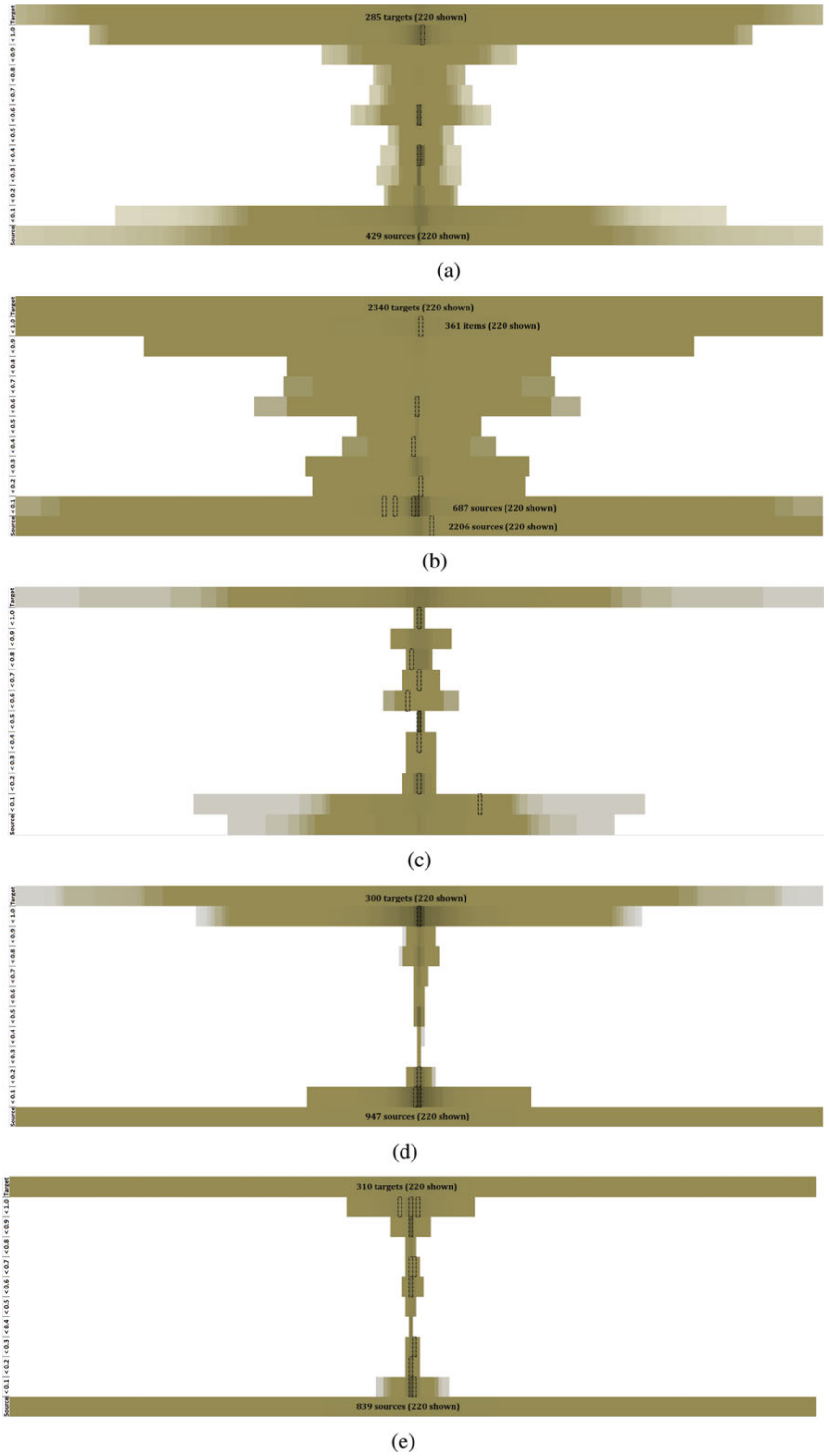
Table A.4. SCCs in the core of the Apache-Math-v3.4 call-graph network.

SCC	Components
SCC-1	*DfpMath:splitPow, Dfp:lessThan, Dfp:align, DfpMath:split, Dfp:dotrap, Dfp:multiply, Dfp:divide, DfpMath:log, Dfp:multiplyFast, DfpMath:logInternal, Dfp:negate, Dfp:add, Dfp:remainder, Dfp:init, Dfp:power10K, Dfp:trunc, Dfp:unequal, DfpMath:splitMult, Dfp:subtract, DfpMath:exp, Dfp:floor, Dfp:newInstance, DfpField:newDfp, Dfp:toDouble, Dfp:rint, Dfp:greaterThan, Dfp:round, DfpMath:pow, DfpMath:expInternal, Dfp:intValue, Dfp:copysign*

Table A.5. The waist of the Rat (*R. Norvegius*) metabolic network. SCCs are listed in Table A.6.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.60	0.60	*Contains the metabolic precursors: Pyruvate, PhosphenolPyruvate, Oxalocetate.*
Arachidonate	0.08	0.09	*Essential for enzyme synthesis.*
Acetyl-CoA	0.25	0.05	*A metabolic precursor.*
SCC-2	0.25	0.04	*Contains the metabolic precursors: Glycerone Phosphate, Ribose-5-Phosphate, Glyceraldehyde 3 Phosphate.*
Phosphatidate	0.07	0.04	*Essential for Lipid synthesis.*
SCC-3	0.02	0.02	*These compounds take part in Purine metabolism.*
{GQ1b, Glycan 9-11}	0.01	0.01	*These compounds take part in Ganglioside metabolism.*

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press



(a)

(b)

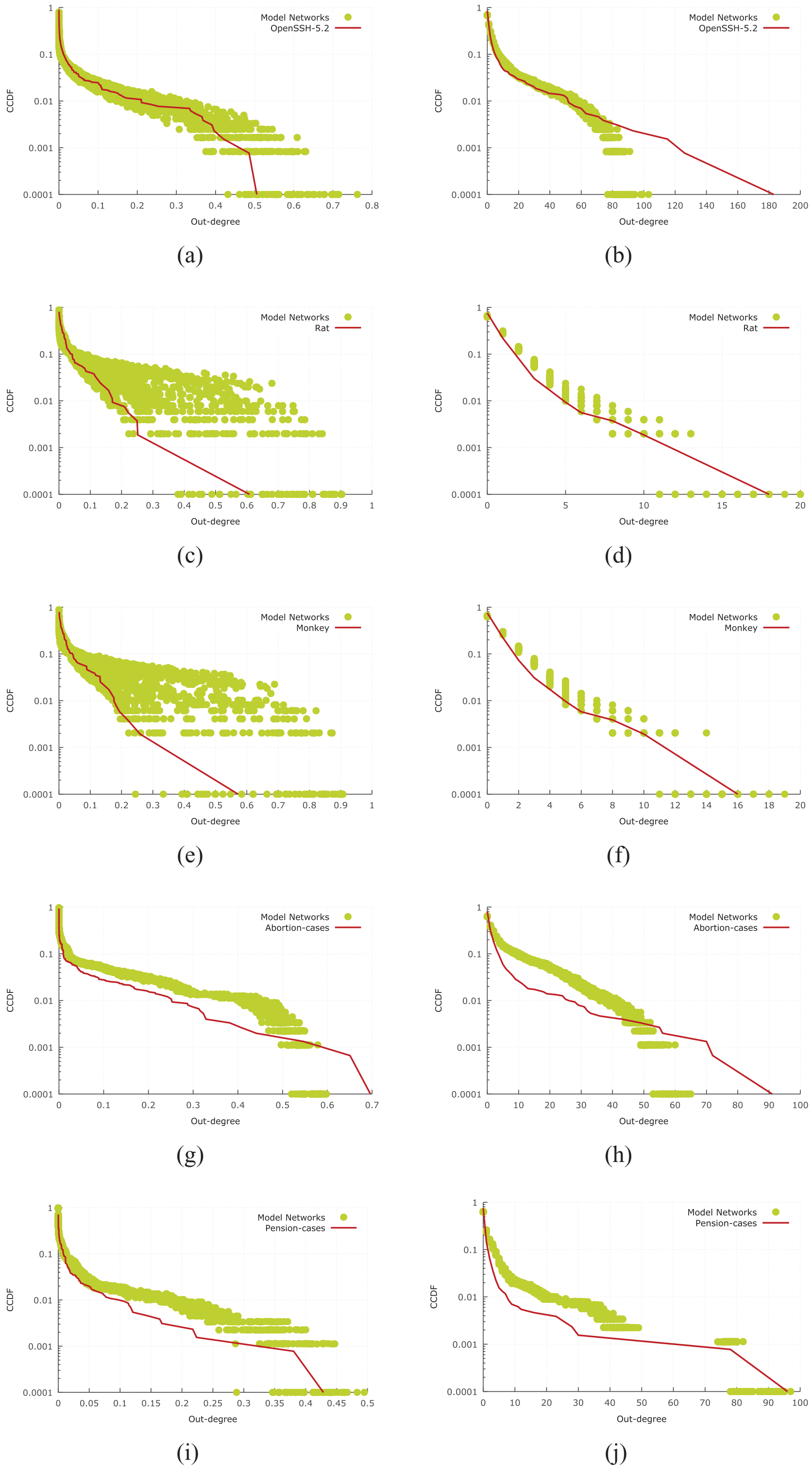
(c)

(d)

(e)

Fig. A.1. Visualizations of the location and path centrality for each network. Please refer to the caption of Figure 8 for a description of this visualization. (a) OpenSSH call-graph. (b) Apache math call-graph. (c) Monkey metabolic network. (d) Abortion case citation network. (e) Pension case citation network. (Color online)

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press



(a) (b)

(c) (d)

(e) (f)

(g) (h)

(i) (j)

Fig. A.2. Comparison of path centrality and out-degree distributions between some real dependency networks and the corresponding synthetic networks generated by the RP-model. (Color online)

Table A.6. SCCs in the core of the Rat (*R. Norvegius*) metabolic network.

SCC	Components
SCC-1	*Ammonia, Pyruvate, Oxalocetate, L-Alanine, L-Aspartate, Glutathione, Glycine, L-Arginine, L-Glutamine, L-Serine, Phosphoenolpyruvate, gamma-Glutamylcysteine, L-Argininosuccinate, Mercaptopyruvate, L-Cystathionine, Casbene, Carbomoyl Phosphate, Fumarate, Citrolline, Malate, L-Cysteine, L-Ornithine*
SCC-2	*Glycerone Phosphate, Ribose 5-Phosphate, Glyceraldehyde 3-Phosphate, PRPP, D-Xylulose 5-Phosphate, beta-D-Fructose-6-phosphate, Sedoheptulose 7-phosphate, beta-D-Fructose 1,6-bisphosphate*
SCC-3	*AMP, GDP, DNA, Guanine, Deoxyadenosine, dATP, IMP, dGTP, XMP, Xanthine, Guanosine, dADP, dAMP, dGDP, Inosine, Adenine, Hypoxanthine*

Table A.7. The waist of the Monkey (*M. Mulatta*) metabolic network. SCCs are listed in Table A.8.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
SCC-1	0.57	0.57	*Contains the metabolic precursors Pyruvate, PhosphenolPyruvate, Oxalocetate.*
Arachidonate	0.10	0.09	*Essential for enzyme synthesis.*
Acetyl-CoA	0.25	0.06	*A metabolic precursor.*
Phosphatidate	0.08	0.05	*Essential for lipid synthesis.*
SCC-2	0.21	0.03	*Contains the metabolic precursors: Glycerone Phosphate, Ribose-5-Phosphate, Glyceraldehyde 3 Phosphate.*
SCC-3	0.03	0.03	*These compounds take part in Purine metabolism.*
Lc3Cer	0.01	0.01	*Aids in biosynthesis of Glycolipids.*
Malonyl-[acp]	0.01	0.01	*A key compound for fatty acid synthesis.*

Table A.8. SCCs in the core of the Monkey (*M. Mulatta*) metabolic network.

SCC	Components
SCC-1	*Ammonia, Pyruvate, Oxalocetate, L-Alanine, L-Aspartate, Glutathione, Glycine, L-Arginine, L-Glutamine, L-Serine, Phosphoenolpyruvate, gamma-Glutamylcysteine, L-Argininosuccinate, Mercaptopyruvate, Cyc-Gly, Carbomoyl Phosphate, Fumarate, Citrolline, Malate, L-Cysteine, L-Ornithine*
SCC-2	*Glycerone Phosphate, Ribose 5-Phosphate, Glyceraldehyde 3-Phosphate, PRPP, D-Xylulose 5-Phosphate, beta-D-Fructose-6-phosphate, Sedoheptulose 7-phosphate, beta-D-Fructose 1,6-bisphosphate*
SCC-3	*AMP, GDP, DNA, Guanine, Deoxyadenosine, dATP, IMP, dGTP, XMP, Xanthine, Guanosine, dADP, dAMP, dGDP, Inosine, Adenine, Hypoxanthine, Adenosine, GMP, Adenylosuccinate, Xanthosine*

Table A.9. The waist of the SCotUS citation network on Abortion cases. Cases labeled as "landmarks" are listed as Historic by the Legal Information Institute at Cornell University.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
Planned Parenthood v. Casey (1992)	0.69	0.69	*A "landmark" decision on abortion rights.*
Roe v. Wade (1973)	0.65	0.20	*A "landmark" decision in favor of abortion rights with* certain restrictions.
Bigelow v. Virginia (1975)	0.38	0.05	*A "landmark" decision on protecting First Amendment right on advertising, where the advertisement in question was on abortion services.*
Harris v. McRae (1980)	0.55	0.03	*A "landmark" decision regarding federal funds restriction on abortion.*

Table A.10. The waist of the SCotUS citation network on Pension cases. Cases labeled as "landmarks" are listed as Historic by the Legal Information Institute at Cornell University.

Name	$\frac{P(v)}{P}$	$\delta_C(v)$	Description
Goldberg v. Kelly (1970)	0.42	0.42	*A "landmark" decision that established the full evidential hearing requirement before termination of welfare benefits.*
Allied Structural Steel Co. v. Spannaus (1978)	0.22	0.22	*A "landmark" decision that reinstated pension rights for certain Allied Steel employees.*
L.A. Dept. of Water & Power v. Manhart (1978)	0.16	0.11	*A "landmark" decision that stated discrimination in pension contribution requirement based on sex is unlawful.*
US Railroad Retirement Bd. v. Fritz (1980)	0.38	0.09	*A "landmark" decision that reinstated pension rights for certain US Railroad employees.*
Johnson v. Robison (1974)	0.21	0.03	*A decision that retained certain benefits for combat veterans.*
Hishon v. King & Spalding (1984)	0.08	0.02	*A decision regarding benefit discrimination based on sex.*
Helvering v. Davis (1937)	0.06	0.02	*A "landmark" decision defending the constitutional validity of the Social Security Act.*
Nollan v. California Coastal Com. (1987)	0.07	0.01	*A decision concerning 5th and 14th amendment for property protection.*
United States v. Kokinda (1990)	0.11	0.01	*A decision involving first amendment rights for free speech.*
Pension Benefit Guar. Corp. v. LTV Corp. (1990)	0.17	0.01	*A decision involving insurance of pension benefits.*
Plaut v. Spendthrift Farm (1995)	0.11	0.01	*A decision concerning separation of power between legislation and judiciary.*

https://doi.org/10.1017/nws.2017.22 Published online by Cambridge University Press