Supporting Information

Pang and Maslov 10.1073/pnas.1217795110

SI Materials and Methods

Obtaining the Dependency Network and Occurrence Frequency Data for Linux Packages. The package dependency network of Linux distribution Ubuntu 11.04 Natty was obtained by first getting a complete list of packages from http://packages.ubuntu.com/, and then running the command apt-rdepends to find all of the direct and indirect requirements for each package. The resulting network contains 33,473 packages, and 57,667 direct and 2,439,011 total (direct + indirect) dependency relations.

The occurrence frequency data for 192,392 packages on 2,047,796 computers was downloaded from the package popularity contest (popcon) project (http://popcon.ubuntu.com/by_inst) (1). Participants of this project installed on their Linux computers tracking software that automatically reports the installation and subsequent use of different packages to the popcon server. We used the first column reporting the number of computers where this package was installed. A total of 189,711 packages were installed on at least one computer. Other columns not used in this study report the number of computers where this package was or was not used in the past month and the number of computers where it was recently updated.

The popcon project obtained the package data from Ubuntu Linux of a wide range of versions and CPU architectures, whose package repertoire and dependencies are a little bit different from each other. In the analysis we assumed that all participants are using Ubuntu 11.04 with x86 architecture, and based on this version of the Ubuntu Linux we calculated the direct and total dependency degree (k_{dep} and K_{dep}) of every package, and plotted the f vs. K_{dep} in Fig. 2. The packages not included in the official repositories of Ubuntu 11.04 or having zero installation frequency were ignored. Packages that are not required by any other packages (i.e., those with $K_{dep} = 0$) were also ignored.

Construction of Dependency Matrices for the Metabolic Network. The Kyoto Encyclopedia of Genes and Genomes Database (KEGG) (2) contains the data of metabolic reactions present in different organisms, and the universal metabolic network used in this study is the union of all of the reactions in KEGG consisting of 5,759 reactions and 4,785 metabolites. The group of five common metabolites present in the majority of organisms was selected as the core: H_2O, ATP, NAD+, oxygen, and CoA. The final version of the dependency network used in our study contains 1,832 reactions (or associated enzymes) connected to each other by 3,118 direct and 49,168 direct + indirect dependencies.

The goal of the metabolic network is to either convert nutrients taken up from the environment into core metabolites (catabolism), or to convert core metabolites into the constituents of the biomass and other essential ingredients (anabolism). The direction of the dependency network connecting metabolic reactions would be opposite in these two cases. For simplicity we will concentrate on the case of anabolic pathways below. For catabolic pathways we simply inverted the direction of reactions and then applied the procedure used for anabolic pathways.

To determine the set of other enzymes an enzyme i in an anabolic pathway depends on for its operation, we performed the following computational analysis. We selected all metabolic substrates of the enzyme i one by one, and for each of them we constructed the minimal pathway necessary to synthesize this metabolite from our predetermined set of 40 core metabolites. The union of all enzymes in these pathways constructed for each of the substrates of the enzyme i is a good approximation to the minimal set of enzymes necessary to enable the reaction catalyzed by the enzyme i; as such, we can plausibly assume that the enzymes in this union form the total downstream dependency set for the enzyme i.

Furthermore, by analogy to software dependency networks, the direct dependency neighbors of the enzyme i are made by the set of enzymes added at the last layer of our breadth-first search algorithm. Based on this definition, the direct dependency degree of an anabolic enzyme is closely related to the number of metabolic reactions using at least one of its products, which is one of the standard topological definitions of degree in metabolic networks (3). Therefore, the power-law distribution with the exponent −2 we measured for direct dependency degrees is closely related to previously reported scale-free topology on metabolic networks (3).

The rules by which this minimal pathway was constructed were previously described in ref. 4. For the sake of completeness, we included them in the text below.

By repeating the above procedure for all anabolic (catabolic) enzymes located downstream (upstream) from our core metabolites and thus reachable from the core by the scope expansion algorithm (5), we constructed our best approximation to the total dependency network of metabolic enzymes in the KEGG database.

Rules of Addition of Anabolic Pathways in Dependency Network Calculation.

i) At the beginning of the simulation, the model organism starts with a "seed" metabolic network consisting of 5 metabolites including H_2O, ATP, NAD+, oxygen, and CoA. It is assumed that our organism is able to generate all of these metabolites by some unspecified catabolic pathways.

ii) At each step, a new metabolite that cannot yet be synthesized by the organism is randomly selected from the scope (5) of our seed metabolites. This scope consists of all metabolites that in principle could be synthesized from the seed metabolites using all reactions listed in the KEGG database (5).

iii) To search for the minimal pathway that converts core metabolites to this target we first perform the scope expansion (5) of the core until it first reaches the target. In the course of this expansion, reactions and metabolites are added step by step (or layer by layer). Each layer consists of all KEGG reactions that have all their substrates among the metabolites in the current metabolic core of the organism (light blue area in figure 4 of ref. 5) and those generated by reactions in all of the previous layers (see figure 4 of ref. 5 for an illustration).

Mathematical Derivation of the Total Dependency Degree Distribution in the Random Model with $D = 2$. To mathematically derive the distribution of dependency degree K_{dep} in the simple model proposed in this study, we study its dependence on the time, t, a package was added to the growing dependency network. Here, time t is defined as the size of the network when a package was added and may have a nonlinear but monotonic relation to the actual time of addition (e.g., in exponentially expanding systems). $K_{dep}(t)$ can be calculated self-consistently from the following equation:

$$K_{dep}(t) = 1 + \int_{t+1}^{N} K_{dep}(t')D/t'. \qquad \textbf{[S1]}$$

Indeed, the total dependency degree of a package added at time t is given by the sum of total dependency degrees of packages added at later times, t', that directly depend on it. K_{dep} counts both direct and indirect dependencies, and thus indirect

dependencies of upstream packages are transferred to their downstream neighbors. In a random model, the likelihood of a package added at time t' to send a direct dependency link to a package added at time t is simply D/t'. It is easy to check that

$$K_{dep}(t) = (t/N)^{-D} \quad \text{[S2]}$$

is a solution of this equation. Indeed,

$1 + \int_{t+1}^{N} D(t'/N)^{-D} dt'/t' \simeq 1 + (t/N)^{-D} - (N/N)^{-D} = (t/N)^{-D} = K_{dep}(t)$. Eq. **S1** simply adds up the dependency degrees of multiple upstream neighbors of a node and thus ignores the inevitable overlap between these sets of nodes; this is a good approximation as long as the resulting $K_{dep}(t) \ll N$, and thus the overlap is small. It is clear, however, that if $D > 1$, Eq. **S2** cannot hold forever because it predicts $K_{dep}(1) = (1/N)^{-D} = N^D \gg N$. The total dependency degree cannot be larger than N, and this value is approximately reached at $t = N_c$ determined by

$$(N_c/N)^{-D} = N \quad \text{or} \quad N_c = N^{(D-1)/D}. \quad \text{[S3]}$$

N_c is the number of nearly universal "core" components in the system with total dependency degree $K_{dep} \simeq N$.

Eq. **S2** fully determines the power-law tail of the distribution of dependency degrees. Indeed, $P(K_{dep} \geq K) = P((t/N)^{-D} \geq K) = P((t \leq NK^{-1/D}) = NK^{-1/D}/N = K^{-1/D}$. Hence, $P(K_{dep} = K) = -dP(K_{dep} \geq K)/dK$ is given by

$$P(K_{dep}) \sim K_{dep}^{-(1+1/D)}. \quad \text{[S4]}$$

For $D = 2$, which is close to its empirical value in real-life biological and technological systems used in this study, one recovers familiar scaling laws:

$$P(K_{dep}) \sim K_{dep}^{-1.5} \quad \text{[S5]}$$

and

$$N_c = \sqrt{N}. \quad \text{[S6]}$$

Mathematical Derivation of the Total Dependency Degree Distribution in a Tree Generated by a Galton–Watson Branching Process. A Galton–Watson branching process is a Markov process in which every node in generation l produces some random number of "child nodes" in generation $l + 1$, according to a fixed probability distribution that does not vary from node to node. We denote as p_0 the probability for the process to terminate at each node, and p_d is the probability for a node to have a branch with d child nodes. The first node of the tree generated by a Galton–Watson branching process is denoted as the root. In biological and technological systems considered in this study, the root node represents the set of core metabolites, or the basic Linux packages that serve many high-level user applications. The scaling properties of the Galton–Watson process are fully determined by a single parameter, $\overline{d} = \sum_{d=0}^{d_{\max}} d \times p_d$, which is the average number of child nodes of any given node has. For $\overline{d} < 1$, referred to as an undercritical branching process, the cascade will terminate very quickly and is irrelevant to this study. Conversely, for $\overline{d} > 1$, referred to as a supercritical branching process, the cascades will likely never terminate. Moreover, for a given number of nodes N in a tree generated by an overcritical branching process, the total number of layers L is logarithmically small: $L \sim \log N / \log \overline{d}$. Real-life complex multicomponent systems such as metabolic networks and large software projects are characterized by a large number of hierarchical levels (4) incompatible by that in an overcritical branching process. Thus, overcritical branching processes will be also ignored in this study. In what follows, we limit our calculations to the third case in which $\overline{d} = 1$ is denoted as a critical branching process; this was previously demonstrated to be a good approximation to universal metabolic network (4), and the present study presents convincing evidence that it describes large software projects as well.

The direct dependency degree k_{dep} of a node in the Galton–Watson process is given by its number of child nodes plus 1 (to account for the dependency of a node on itself). The distribution of k_{dep} is then determined by p_d as $P(k_{dep} = 1) = p_0$, $P(k_{dep} = 2) = p_1$, $\ldots$ $P(k_{dep} = d) = p_{d-1}$; it can have any functional form as long as its average is equal to $1 + \overline{d}$, that for a critical branching process is equal to 2. It is important to emphasize that the Galton–Watson branching process does not provide an explanation for the power-law form of the distribution of direct dependency degrees (Fig. 3). However, the total dependency degree $K_{dep}(i)$ corresponds to the size of the entire subtree initiated at the node i. The Galton–Watson branching process is a Markov process and thus each node can be thought as starting its own instance of a branching process that is independent of the branching ratios of its predecessors. Hence, one would naively expect that the total dependency degree of N nodes in a tree generated by the critical branching process will have the same power-law distribution $P(K_{dep}) \sim K_{dep}^{-1.5}$ as N independently started branching processes. However, a quick calculation convinces one otherwise. Indeed, the largest dependency degree $D_{\max}$ in this case will be determined by the equation $1/N = P(K_{dep} > D_{\max}) = \sum_{D_{\max}} K_{dep}^{-1.5} \sim D_{\max}^{-0.5}$, or $D_{\max} = N^2$. Thus, the dependency degree cannot be larger than N—the total number of nodes in the tree. The size of the universal network with N nodes imposes a strict cutoff of N on sizes of its subtrees. Thus, the following process reproduces the distribution of sizes of subtrees of a critical branching tree with N nodes. In this process one simulates the critical branching process N times and stops it when and if its size s reaches N nodes if it does not terminate on its own before that. Therefore, among N nodes of the critical branching tree, one expects to find $N \times P(s \geq N) = N \times N^{-0.5} = \sqrt{N}$ nodes with the largest total dependency degree $K_{dep} = N$. The rest of the nodes follow the power-law distribution $P(K_{dep}) \sim K_{dep}^{-1.5}$, and this is indeed what we see in our numerical simulations on the universal network of 5,000 nodes generated by the critical branching process with $p_0 = p_2 = 1/2$ (data not shown).

1. Pennarun A, Allombert B, Reinholdtsen P. Ubuntu Popularity Contest. Available at http://popcon.ubuntu.com. Accessed September 5, 2011.
2. Kanehisa M, Goto S (2000) KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res* 28(1):27–30.
3. Jeong H, Tombor B, Albert R, Oltvai ZN, Barabási A-L (2000) The large-scale organization of metabolic networks. *Nature* 407(6804):651–654.
4. Handorf T, Ebenhöh O, Heinrich R (2005) Expanding metabolic networks: Scopes of compounds, robustness, and evolution. *J Mol Evol* 61(4):498–512.
5. Pang TY, Maslov S (2011) A toolbox model of evolution of metabolic pathways on networks of arbitrary topology. *PLOS Comput Biol* 7(5):e1001137.

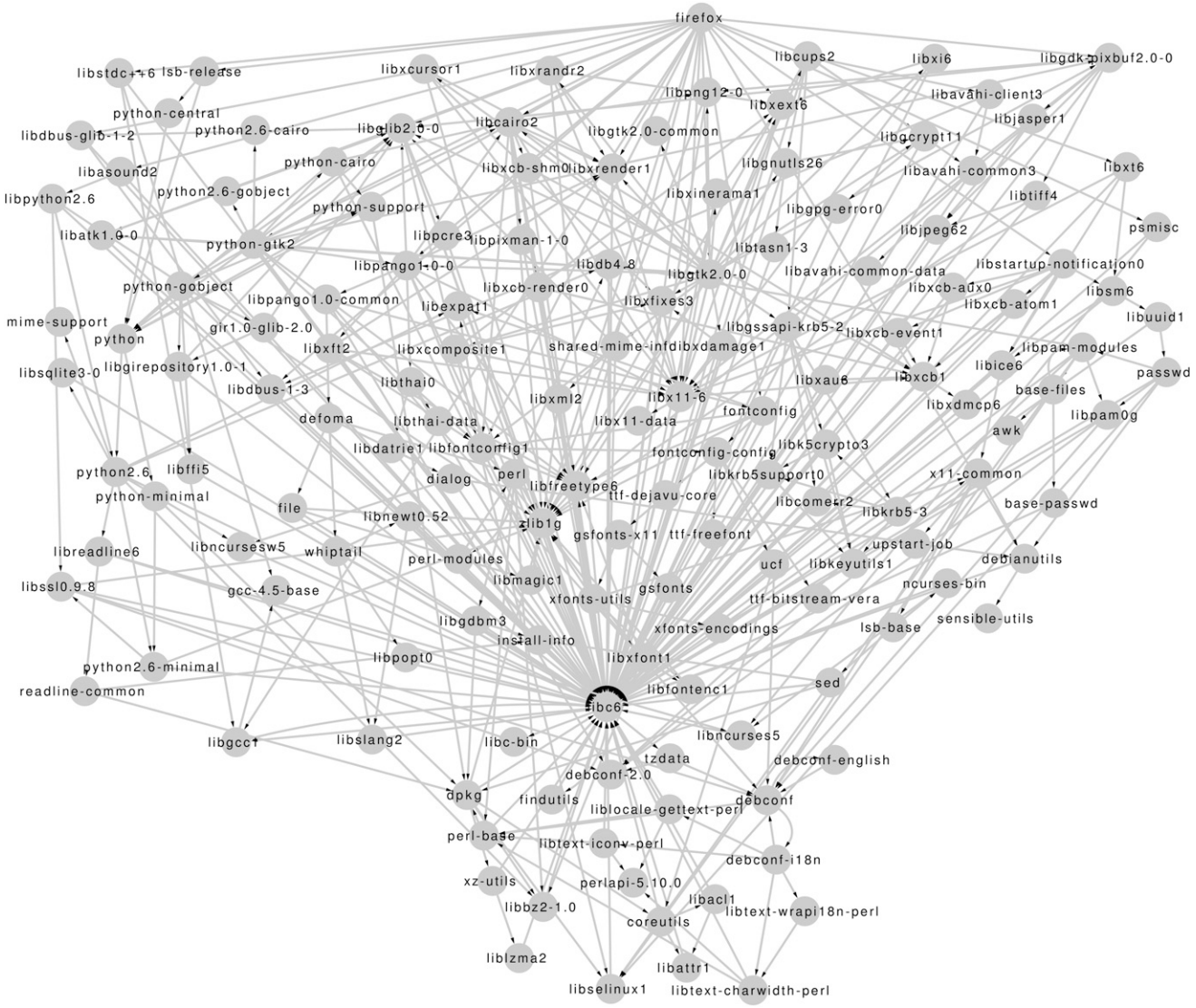

Fig. S1. Pathway diagram created with Cytoscape (1) showing 153 Linux packages directly or indirectly required for installation of the package Firefox (the top node).

1. Shannon P, et al. (2003) Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res* 13(11):2498–2504.

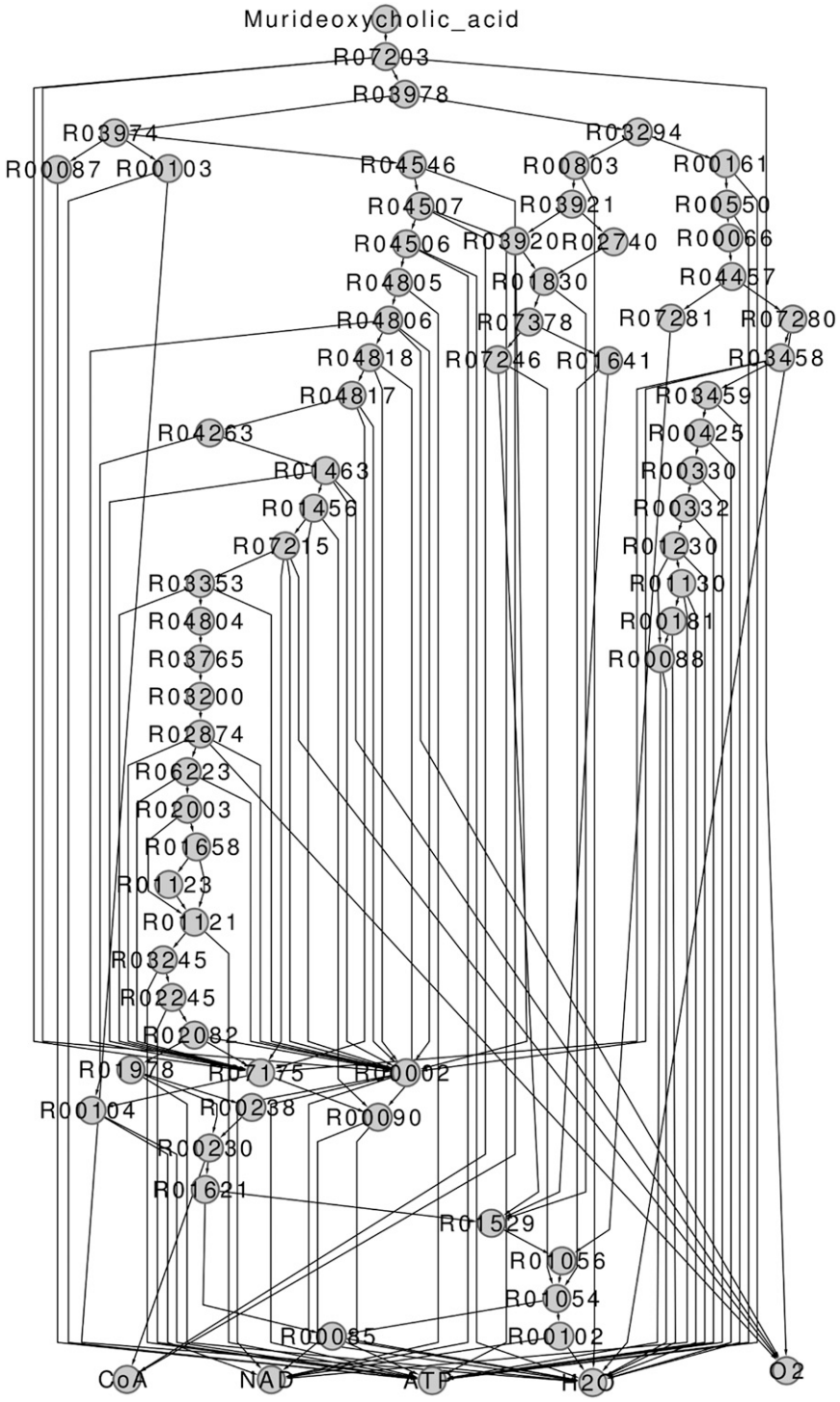

Fig. S2. Pathway diagram created with Cytoscape (1) showing 65 reactions (or equivalently, enzymes catalyzing these reactions) that the production of the metabolite murideoxycholic acid (KEGG database compound C15515; the top node) directly or indirectly depends on. Reaction numbers are given in KEGG notation.

1. Shannon P, et al. (2003) Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res* 13(11):2498–2504.

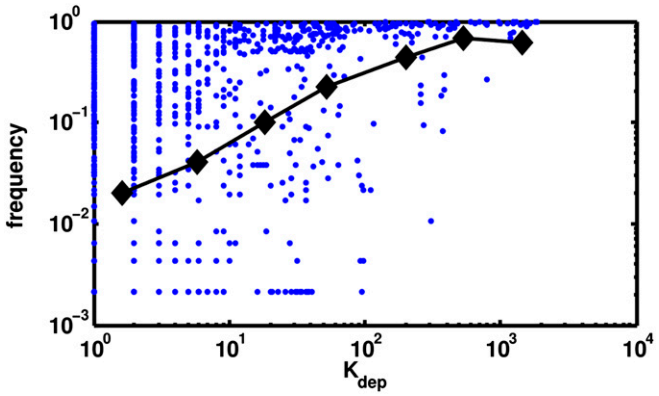

Fig. S3. The frequency of occurrence f (y axis) vs. the total (direct + indirect) dependency degree of metabolites K_{dep}. The two quantities are positively correlated (Spearman's $r_s = 0.45$). The black curve and symbols shows the average f in the logarithmic bins of K_{dep}.