Supporting Information

Wuchty 10.1073/pnas.1311231111

SI Methods

Formulation of the Binary Integer Programming Problem. In general, binary integer programming revolves around the problem of finding a binary vector x that optimizes a linear function $f^T x$ subject to linear constraints. For example, a binary integer programming problem may be $min\ f^T x$ such that $Ax \geq b$, where x only can assume binary values. In particular, A is a matrix that holds the coefficients of the constraining variables, and b holds the boundaries of the constraints.

In Fig. S10*A* we consider a simple network whose minimum dominating set (MDSet) we want to determine. Each node v_i can either take value $x_i = 1$ if v_i is part of the MDSet or 0 otherwise. In Fig. S10*B*, we formulate the underlying binary integer programming problem for our toy network. In particular, we want to minimize the number of nodes in the MDSet by min $(x_1 + x_2 + x_3 + x_4)$. Because we defined the MDSet as the minimum subset of nodes that allow us to reach each remaining non-MDSet node by one step (i.e., non-MDSet nodes are in the vicinity of an MDSet node), we need to make sure that the sum of x's of a given node v_i and its neighbors is at least 1. Therefore, our constraints can be formulated for each node v_i $(i = 1,2,3,4)$ as $x_i + \sum_{j \in \Gamma(i)} x_j \geq 1$, where $\Gamma(i)$ are the neighbors of v_i in the toy network. As a consequence, the matrix notation of the constrains is $Ax \geq 1$ where A is the adjacency matrix of the underlying toy network, where $A_{ii} = 1$ as well as $A_{ij} = 1$ if there exists an edge between nodes v_i and v_j and 0 otherwise.

Solving a Binary Integer Programming Problem with a Branch-and-Bound Algorithm. lpSolve of the R package uses a linear programming (LP)-based branch-and-bound algorithm (1) to solve such a binary integer programming problem. The algorithm searches for an optimal solution to the binary integer programming problem by solving a series of LP-relaxation problems, in which the binary integer requirement on the variables is replaced by the weaker constraint $0 \leq x_i \leq 1$. Briefly, the algorithm (*i*) searches for a binary integer feasible solution, (*ii*) updates the best binary integer feasible solution so far as the search tree grows, and (*iii*) verifies that no better integer feasible solution is possible by solving a series of linear programming problems.

In more detail, the algorithm creates a search tree by repeatedly adding constraints of the problem (Fig. S10*C*). This step is called "branching" where the algorithm chooses a variable x_j that has not been set to an integer value yet. Specifically, the algorithm adds the constraint $x_j = 0$ to form one branch and the constraint $x_j = 1$ to form the other branch. In our example the algorithm starts out to add the constraint revolving around node v_1 to the tree by adding a node where x_1 is set to 0 and another node, where $x_1 = 1$. As a consequence such branching steps generate a binary tree. In general, however, the order of the variables going down the levels in the tree is not necessarily the usual order of their subscripts.

At each node, the algorithm solves an LP-relaxation problem based on the constraints that were used up to this node. In our example, let us consider the node where $x_1 = 0$. Here, x_1 is set to 0, but the remaining variables x_2, x_3, and x_4 are still free to take on either value. In other words, for each variable x_2, x_3, and x_4 we consider a relaxed constraint in the interval [0,1]. In the node where $x_1 = 0$, we, for instance, assume that $x_2 = 0.5$, $x_3 = 0.9$, and $x_4 = 0.8$, resulting in $Z = 2.2$. Such a feasible solution (i.e., no violation of the constraints) to the LP-relaxation problem provides a lower bound for the binary integer programming problem. Inevitably, we will end up with a feasible solution where all variables are set to an integer value (i.e., an integer solution). In this case such a binary integer vector would provide an upper bound or the best current integer solution Z^* for the binary integer programming problem. As a start, when no actual solution is yet available we choose a default of $Z^* = \infty$.

Depending on the outcome of this evaluation step, the algorithm will decide either to continue branching (i.e., add another constraint) or to move to another node. In particular, we consider three possibilities in our toy model:

i) If the LP-relaxation problem at the current node is infeasible or Z is greater than the corresponding value of the best current integer solution Z^*, the algorithm will not search any branches below that node. The algorithm then moves to a new node according to the underlying, implemented search strategy. In our example, the algorithm moved to the other branch where $x_2 = 1$, after we found that the solution to the subproblem with $x_2 = 0$ was infeasible.

ii) If a (new) feasible integer solution with a lower Z value than that of the best current integer solution is found ($Z < Z^*$), then the algorithm keeps this solution as the new best current integer solution and moves on to the next node. In our toy example we found a best current integer solution (and the ultimately optimal solution) where $x_2 = 1$, whereas all other variables were 0, and $Z = Z^* = 1$. Our algorithm continued searching at the last branching step where we left off ($x_3 = 1$). Although the solution was feasible, the corresponding value $Z = 1.9$ was larger than $Z^* = 1$.

iii) If the solution of the LP-relaxation problem is feasible but not integer (i.e., a possible final solution) and the Z of the LP-relaxation problem is less than the best current integer solution $Z < Z^*$, we start a new branching step. Following the path to the optimal solution, we observed a series of branching steps in our toy example, illustrating this step.

Following this algorithmic outline, we end up with an optimal solution of $Z^* = 1$ and $x_1 = 1$, whereas $x_{1,3,4} = 0$, suggesting that the MDSet of our toy network consists of node v_2 only.

1. Land AH, Doig AG (1960) An automatic method of solving discrete programming-problems. *Econometrica* 28(3):497–520.

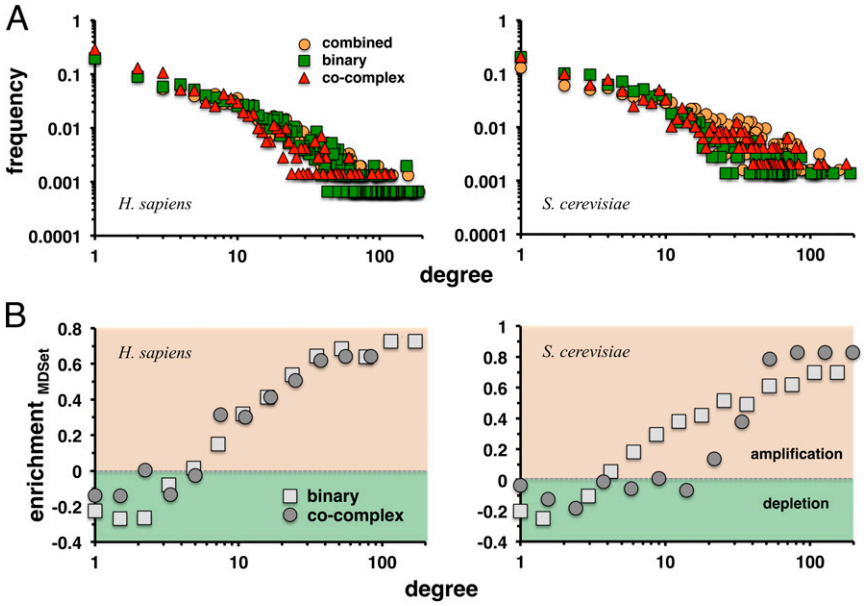

Fig. S1. Degree distributions of MDSet proteins. The degree distributions of proteins that are involved in the MDSet of the combined, binary, and cocomplex interactions in (*A*) human and yeast have fat tails. (*B*) Compared with the log-binned frequency distributions of degrees in the binary and cocomplex interaction datasets, we observed that MDSet proteins are mostly enriched among highly connected proteins in human and yeast.

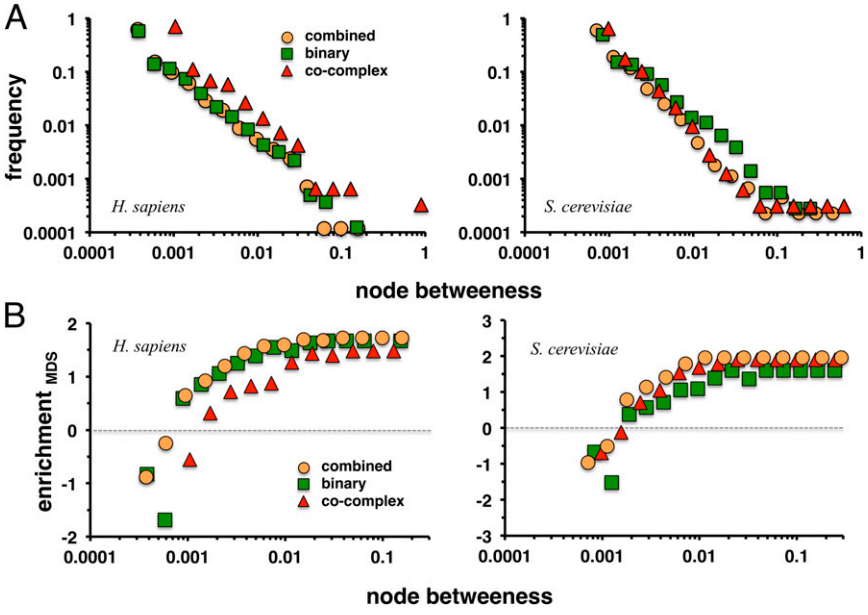

Fig. S2. Betweenness centrality distributions of MDSet proteins. (*A*) The frequency distributions of protein's betweenness centrality that are involved in the MDSet of the combined, binary, and cocomplex interactions in human and yeast have fat tails. (*B*) Determining node-specific betweenness centrality of all proteins in the binary and cocomplex interaction datasets, we observed that MDSet proteins are mostly enriched among proteins with high betweenness in both human and yeast.

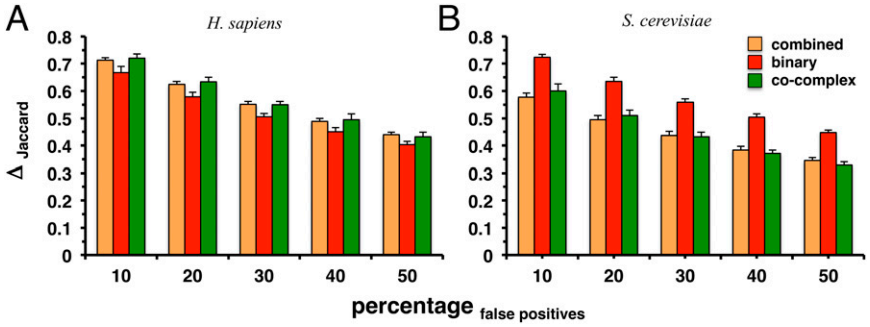

Fig. S3. Robustness of MDSets. We compared the MDSet of the actual and randomized networks by a Jaccard index, defined as $\Delta_{ij} = N_{ij}/(N_i + N_j - N_{ij})$, where N_{ij} was the number of common MDSet proteins in networks i and j, and N_i was the number of MDSet proteins in network i. To represent false-positive interactions we randomly deleted the corresponding percentage of interactions in the underlying combined, binary, and cocomplex interaction sets in (*A*) human and (*B*) yeast.

A

	H. sapiens		
	combined	binary	co-complex
cancer-related	329 (232) 9.9×10^{-43}	306 (237) 2.8×10^{-31}	232 (164) 1.2×10^{-25}
virus-targeted	280 (180) 8.5×10^{-57}	253 (186) 1.1×10^{-40}	199 (132) 3.6×10^{-24}

	S. cerevisiae		
	combined	binary	co-complex
essential	303 (190) 1.6×10^{-51}	216 (228) 5.9×10^{-6}	280 (167) 4.7 $\times 10^{-47}$

B

	H. sapiens		
	combined	binary	co-complex
cancer-related	306 (232) 2.1×10^{-32}	288 (237) 2.1×10^{-24}	219 (164) 6.4×10^{-20}
virus-targeted	272 (180) 9.2×10^{-52}	261 (186) 4.2×10^{-45}	201 (132) 3.0×10^{-25}

	S. cerevisiae		
	combined	binary	co-complex
essential	244 (190) 5.0×10^{-22}	224 (228) 1.3×10^{-7}	212 (167) 9.6×10^{-14}

Fig. S4. Enrichment of hubs among disease and essential genes. (*A*) We calculated how often cancer-related genes, proteins that are targeted by human viruses, and essential genes appeared in sets of most connected proteins that equaled the size of the corresponding MDSet in the given human and yeast interaction sets. Using Fisher's exact test we determined the significance of such functional proteins in these sets of most connected proteins. In parentheses we show the corresponding numbers in the matching MDSets. In *B* we present the corresponding numbers using sets of proteins with highest betweenness centrality that matched the corresponding MDSets in the underlying interaction networks.

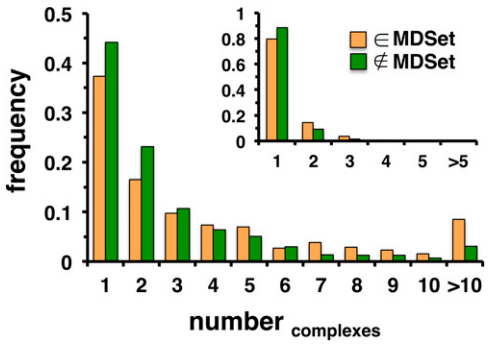

Fig. S5. Involvement of MDSet proteins in protein complexes. Counting the number of complexes proteins are involved in, we observed that human MDSet proteins in the combined human interaction network appeared in more complexes than non-MDSet proteins ($P = 2.6 \times 10^{-8}$, Wilcoxon test). (*Inset*) We observed a similar result using (non)MDSet proteins in the combined yeast interaction network ($P = 4.1 \times 10^{-5}$).

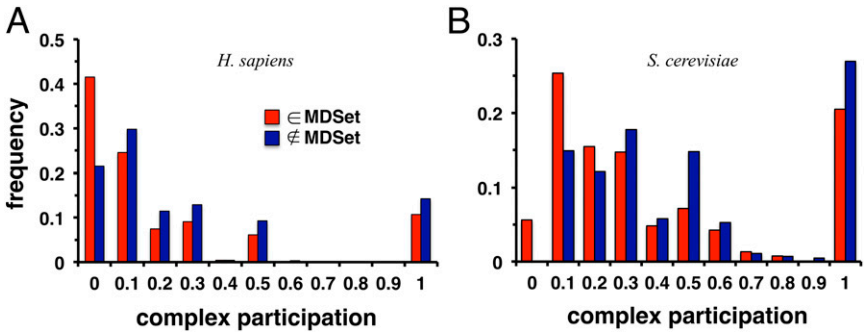

Fig. S6. Complex participation of MDSet proteins. (*A*) Focusing on the combined human network, we found that human MDSet proteins have significantly lower complex participation values than non-MDSet proteins ($P = 2.4 \times 10^{-38}$, Wilcoxon test). In (*B*) we found a similar result for (non)MDSet proteins in the combined yeast interactions set ($P = 3.3 \times 10^{-17}$).

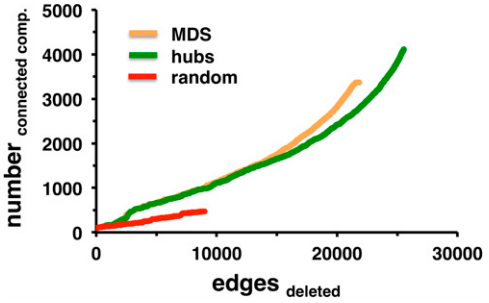

Fig. S7. Robustness of underlying human network. We sorted all MDSet proteins in the combined human interaction network according to their degree. As an equivalent set of equal size we collected and sorted the highest connected hub proteins. Furthermore, we randomly sampled a set of proteins of the same size. Starting with the most connected protein, we gradually deleted proteins and calculated the number of connected components in the altered network.

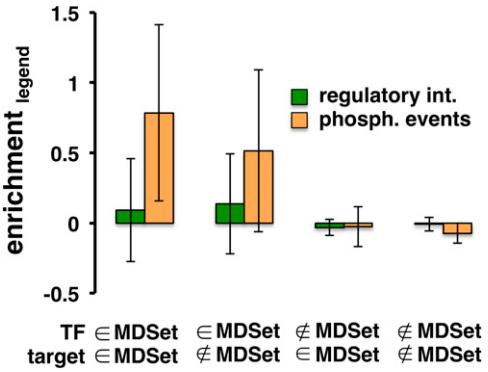

Fig. S8. Enrichment of MDSet proteins among yeast transcription factors, kinases, and their targets. We calculated the enrichment of transcription regulatory interactions and phosphorylation events between (non)MDSet proteins in the combined yeast interaction network. We observed that both types of interactions preferably appeared between MDSet proteins, whereas we found the opposite for non-MDS proteins. Specifically, we obtained best results when the corresponding transcription factor or kinase was involved in the MDSet.

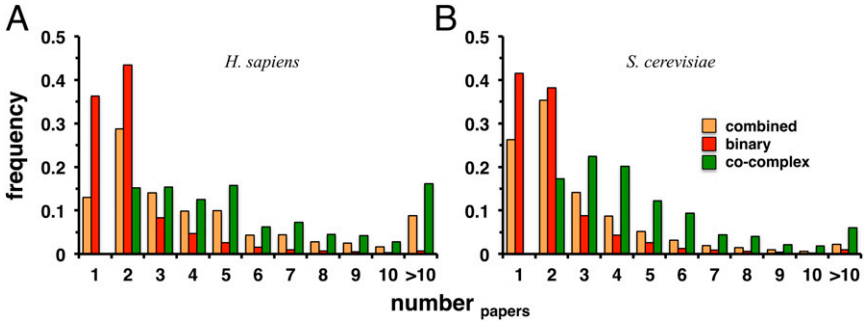

Fig. S9. Quality analysis of protein interaction sets. We counted the number of papers that reported the presence of an interaction. In (*A*) human and (*B*) yeast we observed that the majority of interactions in the binary interactions sets were confirmed in more than one paper. Cocomplex interactions were only accounted for when they appeared in at least two publications.

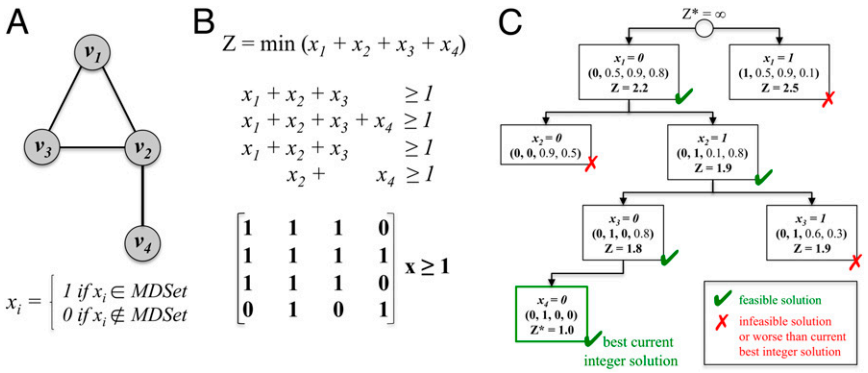

Fig. S10. Concept and solution of binary integer programming problems. (*A*) Toy network to illustrate the determination of an MDSet. (*B*) Formulation of the binary integer programing problem to determine the MDSet of the toy network. (*C*) Search tree as generated by a branch-and-bound algorithm to solve the underlying binary integer programming problem to find an MDSet.