Dissecting Metabolic Networks into Functional Subnets

Wynand S. Verwoerd
WF & Molecular Bioscience Department
Lincoln University, New Zealand

1 Introduction

The processes of life in a living cell are weaved together in an intricate web of interactions between molecules. These range from the transcription and translation of information contained in the DNA macromolecules of the genome into protein molecules, regulation of those processes through interactions between the proteins, transportation of molecules to carry signals or components to specific locations, assembly of molecules into polymers to build structures, and the breakdown of molecules to recycle or dispose of their parts. Much of this is accomplished by linking together simple chemical reactions, each of which only involves a few specific small molecules.

Inherent in the listing of the distinct macro-scale molecular processes above is the recognition that the dissection of this very complex system into parts will make it more comprehensible. However, this only works well if there are not too many individual parts. Therefore, it may be necessary to apply the splitting into components at more than one level. This is the underlying philosophy of the subnet approach.

One high-level component of the molecular web is the biochemical network of small molecule reactions referred to above that supplies the ingredients for the larger scale processes. It is relatively easy to understand individual reactions in the network; however, the complexity lies in how they are interconnected. The conventional approach to interpret the role of a reaction is to elucidate one or more pathways involved in an established biological function, such as respiration, in which the reaction participates. The problem is that with the expanding knowledge facilitated by the genomic revolution, there are so many known pathways that one is once more confronted with the complexity of how the pathways interconnect.

Constructing metabolic subnetworks is an approach aimed at filling in the complexity gap between individual pathways, typically consisting of 10–20 reactions, and the complete network of thousands of reactions. The idea is to divide the complete network into parts that are sufficiently small to allow insight into how they function but large enough to put reactions and pathways into the same context that they have in the complete network. Ideally, each subnet should also represent an identifiable biological function. If only a particular functional area is of interest, the aim might be to extract a single relevant subnet. In the former case of a general subdivision, reactions would usually be uniquely assigned to a particular subnet; however, in the latter case, subnets associated with different particular functional areas might overlap.

Two distinct sets of considerations contribute to constructing subnets. The first is determined by the inherent structure of the network. By recognizing structural features such as subsets of nodes that are particularly closely connected, bottlenecks, or hierarchical structures, one might identify subnets that optimize preservation of these features. The other set of considerations goes beyond structure and is based on the
interpretation of what the network means – for example to ascribe priorities to preserving different stoichiometric constraints, or to include pathways that contribute to some biological function even though they may not be closely connected structurally, or to prefer a moderate number of medium-sized subnets to an unequal size distribution. Structural considerations are likely to be accommodated by an automated algorithm; however, the second set is more subjective and may depend on domain knowledge not explicit in the network specification, and on the purpose for which subnets are required.

Many algorithms for partitioning graphs or networks according to structure have been developed and are, in general, used in bioinformatics (see [Nayak and De, 2007] for a review). However, these tend to provide the user with a very limited scope for influencing the outcome beyond setting a few parameter values that influence overall granularity.

The approach taken here is to produce a computational framework that reflects the structural features, presenting this to the user in such a manner that intelligent decisions based on the subjective considerations can be made within the limits set by the framework. Somewhat surprisingly, the structural partitioning is generally compatible with recognizable biological functions.

It should be clear from this that the partitioning is by no means unique. This is the price of flexibility; however, this is actually inherent in the underlying concept. Even in defining the metabolic network itself, there are subjective questions. For example, should the transport of a molecule between cellular compartments be considered a chemical reaction? Strictly, it is not; however, because it can easily be placed in the same mathematical framework, the answer is usually taken as yes. Still, for considering processes inside a single compartment, such as the mitochondria, it might be appropriate to answer no. What about polymerization reactions? They are clearly legitimate reactions; however, they are usually excluded because their recursive nature makes them harder to include, or their inclusion would lead to huge increases in the number of metabolite molecules. Thus, while the synthesis of nucleotides or amino acids is included, that of DNA or proteins is not. Sensibly choosing the dividing line is crucial to reaching the goal of a manageable subunit that improves understanding of the bigger system.

There are also practical uses for metabolic subnets. The computational effort needed to solve and interpret dynamical models of the kinetics of coupled reaction equations makes it very desirable to limit these models to as few as possible reactions. For steady-state modeling, there are sophisticated methods for analyzing large reaction networks, such as elementary modes and extreme pathways [Palsson, 2006], as well as flux balance analysis (FBA) [Klamt et al., 2007]. However, size matters even here. The so-called combinatorial explosion can lead to millions of modes for a complex network as a result of joining modest mode numbers in appropriately chosen subnets. In FBA, the problem is that knowledge of some measured fluxes is needed to calculate others by applying stoichiometric and other constraints. Thus, in a study that focuses on a particular aspect of metabolism, it would be helpful to find a method that limits the FBA calculation to a “relevant” section of the network and avoids needing boundary conditions that only affect other metabolic aspects.

2 The Splitting Procedure

The approach described here has been implemented in a program called NETSPLITTER, available for download as a Mathematica [Wolfram Research, 2008] notebook [Verwoerd, 2010a]. It builds on ideas from two previous studies: the Markov clustering algorithm (MCL) [Enright et al., 2002] used for partitioning of general simple graphs, and the connectivity-based subnetwork decomposition of metabolic networks as proposed by [Schuster et al., 2002]. A detailed description of the algorithm has been published elsewhere [Verwoerd, 2010b]; therefore, only an overview to introduce the concepts is given here.
The MCL is a powerful yet mathematically simple approach based on representing network structure as a matrix of probabilities that reflect random walks on a simple graph. A simple graph consists of nodes that are connected by lines or “edges” and where every edge connects only two nodes. A metabolic network is more complicated, and will be discussed later. Imagine that at time = 0, there is a single “jumper” on each node. In the next time step, it jumps to a neighboring node, choosing randomly between the edges leading from the node it currently occupies. Each row and column of the probability matrix corresponds to a node in the network. At time = 0, each numerical value in a particular row is the probability that the jumper on the row node will reach the node corresponding to its column in a single jump. The mathematical operation of raising the matrix to the $N$-th power gives the probabilities for the jumpers to progress randomly from the row node to the column node in $N$ jumps.

Moreover, even for networks with thousands of nodes, the $N$-th power converges for a relatively small $N$ of the order of low tens to a matrix with many empty columns. This means that the distribution of jumpers stabilizes through jumpers congregating on a subset of nodes called “sinks” while the rest are the “source” nodes (the empty columns). In effect, the network is replaced by a new graph containing the same nodes as the original network, but in which all edges directly connect sources to sinks in a star-like configuration. This is formally described as a directed acyclic graph (DAG). The probability matrix obtained by potentiating (describing this new network) is referred to as the DAG matrix. A notable feature of this matrix is that it encapsulates long-range connections between nodes because, for example, a zero value at position $(i, j)$ in the matrix implies that Node $j$ cannot be reached starting from Node $i$ by a random walk of any length.

In the MCL method, a second matrix operation is combined with the described power-raising, which progressively eliminates weaker connections so that the DAG matrix separates into subnets consisting of the most strongly connected clusters of nodes. The NETSPLITTER method does not use this method and instead performs the separation in a way tailored to the biochemical nature of the metabolic network.

However, before going into a description of the method, we need to address the fact that, unlike simple graphs, the metabolic network is a bipartite graph that contains two distinct types of nodes: metabolites and reactions. Each reaction node has incoming edges from metabolite nodes that represent its reactants, and outgoing edges to its product nodes.

A metabolic network is commonly specified by its stoichiometry matrix. Each metabolite is associated with a row of this S-matrix, and each column with a reaction. The numerical entries in each column are the stoichiometry coefficients that specify how many molecules of each metabolite participate in the reaction, with negative values for reactants and positive for products. In effect, each row specifies the mass balance constraint for its metabolite.

As shown in [Verwoerd, 2010b] a series of matrix operations can be applied to the S-matrix to transform it into a probability matrix for a reduced network that only contains metabolite nodes. The reduced network still contains all the metabolite nodes of the metabolic network. However, the transformation includes a matrix product that sums over all the reactions so that reaction nodes do not explicitly appear in the reduced network. Hence, there is an edge leading from the metabolite Node $A$ in the reduced network, to Node $B$, if any reaction in the metabolic network transforms $A$ into $B$. In this way, it retains all the connectivity information although the detailed stoichiometry constraints are lost.

Nevertheless, one can still identify where stoichiometry constraints apply by keeping track of the internal/external status of metabolites. Metabolites that are received from or delivered to the environment are termed external and are buffered by the environment. By contrast, metabolites that are produced by one reaction and consumed by another are referred to as internal and their concentrations do not change in a steady metabolic state. This can only occur by balancing fluxes through the reactions in accordance with the stoichiometry of the reactions. Thus, a stoichiometric mass balance constraint applies to each internal
metabolite, but not to an external metabolite. The distinction is conveniently expressed in a network diagram by placing external nodes on the network periphery.

This is relevant for partitioning a network because, in isolating a subnetwork, a new periphery is created. By severing the connection between the subnet and the rest of the network, some metabolites are received from or/and delivered to the rest of the network. Their mass balance can no longer be guaranteed by the subnet alone. Consequently the status of these metabolites is changed from internal to external. In clustering methods such as MCL, each node is allocated to a particular cluster. However, that would not make sense here as a metabolite that is made external by partitioning belongs to both subnets – as a product of one, and substrate of the other subnet. Clearly, the appropriate way to represent partitioning is to split the metabolite node into two, each becoming an external node in either subnet.

In terms of the random walk representation, a jumper cannot move from metabolite Nodes A to B via an external metabolite Node C because all its links point either away from or towards C. In principle, the network can also exchange a metabolite with the environment; however, because there is no mass balance requirement, this can be considered as two separate external nodes: one buffer for input and the other for output. Hence, external metabolites do not contribute to random walks, and only internal metabolite nodes are included in the probability matrix for the reduced network of the NETSPLITTER. Similarly, when an internal node is made external or split to produce partitioning, that action is represented in the matrix by deleting the corresponding row.

To recapitulate, from the S-matrix that specifies a metabolic network, a random walk probability matrix that only contains the internal metabolite nodes is constructed. This is raised to consecutive powers up to convergence to determine the DAG matrix. Suitable nodes are then selected such that when they are deleted from the matrix, the corresponding network separates into subnets.

This goal requires the capacity to recognize the appearance of disjoint subnets in the probability matrix. The first step is to rearrange metabolites so that the sink nodes resulting from the random walks are listed first and the source nodes last. Since by definition there is no probability of a jumper ending up on a source node, their columns are empty and can be omitted to produce a truncated DAG matrix. Then, in the bottom submatrix formed by source rows, any non-overlapping blocks of non-zero values that appear identify disjoint subnets.

This is illustrated by Figure 1. Matrices are displayed with colored cells representing numerical values according to a color scale.

The top square of the truncated DAG matrix in Figure 1A is relatively uninteresting; the only non-zero elements are those on the diagonal, representing the non-zero probability that a jumper that starts on a sink node will remain there. The lower sinks × sources submatrix, however, contains non-overlapping blocks, dividing each of the sources and sinks sets into two groups. Sources in Group A have a finite probability to end up on a subset of sinks from Group A, but zero probability (white cells) of ending up on sinks belonging to Group B. Conversely, sinks in Group A have zero probability of being fed from sources belonging to B. This situation represents the complete partitioning of the network into Subnets A and B. Note that it is the white blocks at the right and bottom of Block A that define the separation. Cells inside Blocks A and B may be colored or white, merely reflecting the connectivity structure inside the block.

Unless the network started out as disjoint, its DAG representation is more likely to resemble partially overlapping blocks, as in Figure 1b. In this case, the overlap region identifies sink and source nodes common to both of the blocks shown as frames, which define the interface between two potential subnets. Identifying and deleting such rows and columns from the matrix is equivalent to cutting their nodes in the network. This is a key part of the NETSPLITTER algorithm.

However, extensive processing of the raw DAG matrix calculated from the network is necessary before it
Figure 1: Schematic view of: A: A truncated DAG matrix for disjoint subnets A and B; B: Sinks-and-sources submatrix for partially separated subnets

can display the discussed features. This first involves reordering rows and columns in the sinks and sources submatrix to group similar ones together. No network information is lost in this process as the ordering inherited from the S-matrix is arbitrary. The reordering is conducted based on a hierarchical clustering analysis separately applied to the rows and columns of the DAG matrix. The clustering information is further used to construct and apply a “blocking transformation” to the matrix that also assesses the extent to which each particular element can be uniquely assigned to a block as a grey scale value. The rearrangement and blocking computations are quite involved, and are detailed in the original article by [Verwoerd, 2010b].

The result is a display that simultaneously provides a graphic display of how partitioning progresses (even for very large networks) and allows NETSPLITTER to nominate metabolites nodes that it has identified as optimal for cutting in the next round. The user can veto any candidate. Once a node has been cut, the connection structure of the network changes and everything is recalculated. A new round of choices for separation nodes is presented until the user is satisfied that sufficient partitioning has been achieved.

One goal of the procedure outlined is to keep the number of nodes that are cut to the minimum compatible with the aims pursued by the user. This is because for each node that is cut, a metabolite is reclassified from internal to external, with the accompanying loss of information contained in its associated mass balance constraint.

NETSPLITTER also applies a second strategy to minimize this loss. Once the interactive stage is terminated and a number of non-overlapping blocks is determined, the program inspects every external metabolite and all those that are only connected to metabolites in a single block are reincorporated into that block. Many, even most, of the nodes that were cut at intermediate stages can be restored in this process, for example because their job has been taken over by a subsequently identified separation node. In typical metabolic networks, only approximately 2% of all nodes need to be cut to produce the final partitioning.

The connection between loss of stoichiometry constraints and making metabolites external also forms the basis of the connectivity-based network splitting proposed by Schuster et al. [Schuster et al., 2002] and implemented, for example, in the Yanasquare network analysis program [Schwarz et al., 2007]. Here, metabolite nodes that have more neighbor connections in the metabolic network than a predetermined threshold value are made external. The rationale is that since they participate in many reactions, the information loss is not so serious because the imbalance in one subnet can be compensated by contributions from an-
other. Other authors [Gagneur et al., 2003; Holme et al., 2003; Ma et al., 2004] have remarked that this only reflects the localized network structure, while partitioning should also take long-range structure into account.

In the NETSPLITTER approach, the two aspects play a complementary role. Connectivity is used as a preliminary filter to identify the most common commodity and currency metabolites such as water, ATP, ADP, and NAD(P)H, among others, that are reclassified as external even before the start of the random walk analysis. This is an essential step; without it, the DAG matrix does not exhibit sufficient internal structure and thus thwarts any attempt to identify partially separated blocks. This is because these highly-connected nodes provide paths through the network between practically every pair of nodes. This is a familiar phenomenon in percolation theory [Grimmett, 1999]. To avoid the problem, the connectivity threshold can be set at a relatively high value, such as 10 or higher. As with earlier, only a few percent of metabolites are made external initially for connectivity reasons while some are reincorporated at the end.

3 Reassembling Smaller Subnet Fragments – the Metanetwork

A second important tool that can be used to construct subnets that are functionally and structurally meaningful is the merging of subnets.

The key to merging is the observation is that once the DAG matrix has been manipulated to separate into blocks, any two blocks can be combined into a bigger block. This is obvious for neighboring blocks such as those in Figure 1A. By reordering, any subset of blocks can be brought together; however, they would still be disjoint. To rejoin them, the reincorporation step, as described for post-processing of the interactive splitting procedure, is performed to restore all external metabolites that only connect to internal metabolites of the two blocks being combined to the status of an internal metabolite in the combined block. Subnet merging conducted in this way is the complete inverse of the splitting process.

By nature, the merging of subnets is driven mostly by the subjective interpretation of the meaning of subnets or the goals set for partitioning. As a result, software support for the process consists mainly of providing information for decision making. The user communicates merge decisions by point-and-click selection of blocks or giving a list of block numbers. The merge is then executed through NETSPLITTER.

One reason to perform merging is that subnets belonging to the same functional area, such as respiration, can sometimes become separated on structural grounds. When this is undesirable, they are specifically selected and merged again. The first problem that will be encountered is the identification of the appropriate blocks out of perhaps 50–100 blocks in the partitioned DAG matrix. This can usually be accomplished by searching in a blockwise list of internal compounds for particular metabolites associated with the function. However, even this step can be tedious and error-prone when the list contains hundreds of names. To assist with the process, NETSPLITTER can read a previously prepared text file with a list of compounds of specific interest to the user, highlighting these in the block listings and graphically in the matrix display. Quite often, all such functional target metabolites end up in the same structural block; but if not, the subset of blocks containing them is visible at a glance and can be merged.

Prime examples of functionally associated subnets are those associated with the same cellular compartment. Allocation of metabolites and reactions to specific compartments such as mitochondria or the cytosol is often encoded (e.g., as a subscript in the name). However this is optional, and even if it is present, allocation plays no role in the partitioning algorithm. Nevertheless, metabolites belonging to a specific compartment tend to group together in one or a few blocks. This type of block, for example one composed mainly of mitochondrial metabolites, will usually also contain a few cytosol compounds as these obviously may be transported into or out of the compartment. Thus, the compartment boundary in terms of the metabolic net-
work structure may not coincide exactly with the physical boundary; however, the fact that they generally do provides added confidence that computational subnet partitioning reflects biological functions.

Another reason for merging is to improve the distribution of block sizes. Conceptually, it would be ideal if a network of N nodes would separate into $\sqrt{N}$ subnets, each containing $\sqrt{N}$ nodes. In that way, the complexity would be evenly spread. In practice, however, the block sizes tend to follow a Pareto distribution with a few very large blocks and many very small blocks. For the pure connectivity-based partitioning of large networks, there is usually only one very large block containing three quarters or more of the metabolites, and the rest is fragmented in small blocks. The NETSPLITTER algorithm performs better in this regard and can usually produce roughly $\sqrt{N}$ medium-to-small blocks. However, there is still a remainder of perhaps twice as many very small fragments, including orphans, containing just two reactions sharing a single internal metabolite. Incorporating these small fragments into their linked medium-sized subnets hardly changes the complexity of the subnet while tidying up the fragmentation.

A key issue to avoid the trivial merging of unconnected subnets is to establish which subnets are structurally linked to each other. This linking is most commonly by shared externals. To provide an overview of such linkages, it is useful to introduce the concept of a metanetwork. The idea is that every subnet is contracted into a single “super reaction” and represented by a single node. Included in this node are all the subnet reactions and internal metabolites, as well as all external metabolites that are exclusive to the subnet. The remaining externals are classified as inflows and outflows shared between subnets, as well as the crossflows produced by one subnet and fed into the next.

NETSPLITTER has a facility to produce a metanet layout showing the subnets, as well as the shared externals and the reactions that connect them. An example of a metanet is shown in Figure 2. The underlying network being processed here represents flavonoid metabolism in a plant species, with 117 reactions and 137 metabolites. This is quite small compared to the full genome scale network; even so, it is difficult to represent intelligibly as a layout.

In effect, the metanet is a condensed view of the network. Greater detail can be seen by zooming in
on a subnet, such as shown in Figure 3. Both diagrams show the bipartite structure by alternating vertically between the metabolite and reaction layers. Color-coding is used to distinguish between the internal metabolites and the three kinds of external metabolites. Orphan subnets are also shown in a separate color on the metanet. For clarity, the diagrams have been simplified by suppressing the display of currency metabolites using a high-connectivity filter.

To demonstrate the use of the metanet diagram for merging, Figure 2 suggests that it would make sense to merge the three orphan subnets labeled by their internal metabolites coumaraldehyde, 4-coumaroylquinate, and CPD-3041 with Subnet 1 because of the shared externals. Similarly, the CPD-412 orphan could be merged with Subnet 1; however, the position of the shikimate-5P fragment is more ambiguous. A decision as to whether it will be merged with Subnet 2 or Subnet 3 might depend on the functional roles, or even on the fact that, with 28 internal metabolites, Subnet 3 is already quite large; thus, merging with the smaller Subnet 2 might be preferable. Merging Subnets 3 and 4 also appears to be an attractive option.

**Figure 3:** Subnetwork 1 of the metanet in Figure 2

Another kind of linkage between subnets occasionally found in networks that are more complicated is that of an internal metabolite of one subnet occasionally acting as an external metabolite of another. The origin of this phenomenon is related to the reduction of the bipartite metabolic network to a simple
metabolite network for the purposes of the splitting procedure. This is more fully discussed in [Verwoerd, 2010b]. NETSPLITTER indicates this kind of linkage on the metanet and alerts the user in the printout when it occurs. The existence of such an internal-external overlap is usually a strong indication to merge the subnets to avoid any ambiguity on the status of metabolites involved.

It is usually possible to reduce the total number of subnets substantially (by a factor of two or more) through the straightforward use of the linkage clues supplied by NETSPLITTER and some common sense. This can drastically reduce the complexity of the metanet without adding too much complexity to any one subnet. The $\sqrt{N}$ rule is a useful guideline in achieving this result. Nevertheless, there are no strict rules, and one may need to experiment with different merge choices to achieve a satisfactory result. Merging provides a versatile tool for shaping the network partitioning according to the needs of a particular study. Although the convenience of a fully automated network partitioning algorithm such as the MCL has been sacrificed, the option of applying judiciously chosen merge operations provides a great deal of control over the final result while still respecting structural relationships.

4 Extracting Network Specifications from a Database

As stated above, the main computational input needed for the network splitting procedure is a specification of the stoichiometry- or S-matrix. To render the results intelligible, this is supplemented by the listings of the names of metabolites and reactions. For reactions, the input also has to specify whether or not they are reversible; moreover, any metabolites to be taken as external may also be specified. If not, the computation determines or allocates their status as described above.

In some instances, ready-made metabolic networks, usually specified as Systems Biology Markup Language (SBML) files, may be available from model repositories such as [Biomodels Database, 2010]. In other cases however, the network information has to be extracted from a metabolic database such as those conforming to the BioCyc specification [Karp et al., 2005]. This is not quite as straightforward as it may seem, for a number of reasons. The remarks below are based on experience with BioCyc databases but may also be applicable to other metabolic databases.

A typical BioCyc database contains considerably more information than just the reaction network. The fully sequenced genome of an organism is used as the starting point; annotated genes provide information on enzymes. In turn, this provides information on the chemical reactions that constitute the metabolic network. Some of the information in this chain can be generated through automated comparison with gene and enzyme information from other organisms; however, painstaking curation is required to ensure that a putative reaction is actually present in the target organism, or to eliminate gaps in known pathways due to incomplete gene annotation.

This results in some reaction specifications being incomplete or inaccurate. Issues that could arise are whether a reaction is chemically balanced and whether it is a generic reaction.

Since the essence of the S-matrix is to specify mass balance constraints of atom species on both sides of the reaction equation, it does not make sense to include unbalanced reactions into the network specification. Such mass balance is sometimes lacking because it may be known that a reaction transforms Substance A into B, but not what other reactants also participate. Hydrogen balance also presents a particular challenge because of uncertainty of the environmental pH. Checking the mass balance is part of the curation task and is mostly certified by an explicit “BALANCED” tag in the reaction record in the database. However, since curation is an ongoing task, it cannot be assumed that a reaction that is not explicitly tagged is automatically unbalanced. The safest strategy is to check the mass balance for each equation as it is extracted. This requires the chemical formulas of all reactants usually stored in the database as a separate table, which again
Substantial numbers of reaction equations in a BioCyc database are specified in a generic, rather than an explicit, form. Various cases can occur. There are redox reactions specified as involving NAD(P)H and that need to be split into a pair of reactions with NADH and NADPH. There are also generic metabolite names specified in a format such as $\text{an aldehyde}$. Reactions involving these may or may not be balanced depending on further assumptions; however, they usually cannot be checked automatically because there is no explicit chemical formula. Finally, a recursive polymerization reaction such as $(\text{CH}_2)_N + \text{CH}_2 \rightarrow (\text{CH}_2)_{N+1}$, even though balanced, is difficult to accommodate because each row in the S-matrix designates a specific metabolite and all values of $N$ would have to be individually present.

Similar problems arise when the same reactant is repeated either on different sides or on the same side of the equation. For example, $\text{ATP} + \text{AMP} \rightarrow \text{ADP} + \text{ADP}$ may be an appealing representation from the chemical point of view, but as there is only provision for a single number to specify ADP in the stoichiometry matrix, the double appearance has to be collected to give the resulting coefficient 2. This can normally be taken care of by automatic processing; however, a case such as Folatepolyglutamate-$n$ + Glutamate $\rightarrow$ $|\text{Folatepolyglutamate}-n|$ can give rise to nonsensical equations.

There are also complications in assigning reaction directions and reversibility. As listed in the database reactions table, the reactants are assigned to the left or right hand side of the reaction equation based on Enzyme Commission conventions, rather than physical evidence. Such evidence can be found in two other tables of the database: explicit pathway specifications and enzyme specifications. The latter is complicated by the fact that there may be several enzymes that catalyze a reaction, some of them possibly in opposing directions, some perhaps in both directions, and others with no direction assigned.

The following conservative strategy may be employed to reconcile all this information. First, the enzyme table is inspected for all enzymes associated with the reaction. A reaction is marked as reversible only if (i) any enzyme given as reversible is found, even if others are unidirectional or unknown, or (ii) there are enzymes for both directions. Where there is no direction information in the enzyme file for a given enzyme but there is another enzyme for the same reaction with a definite direction, the conservative assumption that the unspecified one runs in the same direction is initially made. This may have to be amended if it conflicts with a direction given in a pathway.

The next step is to inspect all pathways specified in the database and containing the particular reaction, and collect all directions assigned to the reaction. The strategy followed in the reconciliation is to give precedence to the direction as was assigned from the enzymes, as only a particular direction would usually be included in a pathway, even when the alternative is also possible. However, if there is no information in the enzyme file and a unique direction in pathways, the conservative assumption is taken that only this direction occurs. Any conflict between enzyme and pathway unique directions is resolved by assuming that since the reaction is “known” to occur in one direction in an identified pathway, an unspecified or unknown enzyme must catalyze it in that direction, and the reaction is taken as reversible. Finally, if neither source gives information (e.g., for uncatalyzed reactions), the reaction is assumed to be in the forward direction.

Performing all these checks and data collection for thousands of reactions is clearly not trivial, and the software to perform the task for BioCyc databases is provided for download as supplementary material to this chapter. The software consists of a suite of scripts for the AWK program [Robbins, 2004] routinely provided in Unix installations, and driven by a Unix script run from a command line. Under a Windows operating system, the scripts can be run under the Unix emulator [Cygwin, 2010] available for free.

Alternatively, the YANAsquare program [Schwarz et al., 2007] includes a module to extract a metabolite network from the Kyoto Encyclopedia of Genes and Genomes KEGG database [Kanehisa et al., 2006].

By implication, the strategy outlined above takes reactions as irreversible by default. The more liberal
default assumption of reversibility is often made in biochemical discussions. However, this can be problematic for network splitting based on the random walk approach because too many reversible links may sometimes hide internal network structures in a similar manner as high-connectivity metabolites.

Realistically, under given metabolic conditions, a reaction can only run in one particular direction. Hence, if the majority or a large percentage of reactions is given as reversible, it is more likely to be an expression of incomplete knowledge than of physical reality. Fortunately, from experience, the overall results of the splitting procedure are not too sensitive to directionality of individual reactions, although cases can certainly be constructed where a block will split in two if a single reaction is reversed or changed from reversible to unidirectional. In principle, this is no different from any other lacking information; adding or removing a reaction can also produce different blocks. In the end, subnet splitting is only as good as the network to which it is applied.

To alleviate concerns on incomplete directionality information in the database, NETSPLITTER is also able to read a file of actual flux values calculated in a flux balance (FBA) calculation. Since such a calculation reflects a particular metabolic state, it takes all non-zero flux values as positive information on reaction directions that override the general information in the database specification. Flux balance calculations for different metabolic states may sometimes be available, and it might be of interest in determining how the subnet partitioning differs for the different states. In this situation, the sensitivity of the partitioning algorithm to directionality would be an asset rather than a liability.

5 Running Example: Metabolism of Mycoplasma Pneumoniae

As a practical example, an application of the NETSPLITTER program to the complete metabolic network of the bacterium Mycoplasma Pneumoniae is demonstrated here step-by-step. In fact, this network is quite small and simple compared to those of eukaryotes, or even other bacteria. However, it has the benefit of being a well-studied model organism for systems biology, and the network was thoroughly curated and experimentally validated in a recent study by [Yus et al., 2009]. It was chosen for the sake of clarity in the presentation, rather than to impress with the performance of the algorithm. However, it does show most of the aspects that are found in full-scale networks a hundred times its size, and which can readily be analyzed using the same procedure.

Electronic specifications for the M. Pneum. metabolic network can be obtained from various sources, including from a Biocyc database, as described above. To facilitate detailed comparison with the published results, the work here was performed using a network specification obtained directly from the authors [Yus et al., 2009]. Rather than the preferred SBML specification, this was supplied as a spreadsheet file listing reaction and metabolite names with the applicable stoichiometric coefficients in table form.

This table was exported as a text file and converted into a suitable input format for NETSPLITTER by the AWK script MakeSmat.txt, supplied as supplementary material with this book. The resulting file was manually augmented to add the unique metabolite IDs that are listed in the article but not included in the spreadsheet. The final input file is also available in the supplementary material.

The resulting file contains 189 reactions and 229 metabolites, including some generic ones such as DNA, RNA, and “Fatty Acid”. In this case, the reactions were not checked for chemical balance to maintain the full complement of 189 reactions represented (e.g., in the metabolic map presented by [Yus et al., 2009] as their Figure 2).

To start, this specification was used as the only input to NETSPLITTER. The result of the first selection round as displayed in the user interface is shown in Figure 4A. The figure has been rotated for better print formatting, so that sinks are shown as rows and sources as columns. There are 104 metabolites recognized
by the program as internal, 17 of which are sinks. The blue background filling the entire range shows that there are no separated blocks at this stage. An aspect that is problematic for separating blocks in further stages, is that most of the interior is occupied by a single dark block. While there are some grey areas, they show very little internal structure. NETSPLITTER suggests a single metabolite, No. 17, as a candidate to be made external. However, it is clear at a glance that this will not achieve any significant block separation.

**Figure 4:** Blocking matrices at the first round of subnet separation: (A) External connectivity cutoff = 8, no external metabolites input file; (B) External connectivity cutoff = 4, no external metabolite input file; and (C) External connectivity cutoff = 8, common external metabolites listed in a separate input file.

The problem at this stage is that too few metabolites have been recognized as external – only those with connectivities of 9 or higher, as the default cutoff value = 8 for internals was used in this calculation.

There are two possible remedies for this situation. First, the cutoff value can be reduced; in fact, it has to be reduced to the value of 4 to make a difference. The result is shown in Figure 4B. The network is fragmented into 26 subnets (including 14 orphans, not shown in the figure). Four of these are of significant size and show some internal structure promising further separation in subsequent rounds.

The other possibility is to specify externals in a separate input file. In this case, the external metabolite file prepared has the content shown in Figure 5. Inline comments in this file can optionally be included preceded by a "#" sign. The first section lists metabolites that were specifically labeled as extracellular compounds in the network specification. NETSPLITTER will probably recognize these automatically, but inclusion here makes it explicit. The next section lists a number of common sugar phosphates; most of these should be recognized automatically because of their high connectivities. However, in a small network such as in this example, it is better not to rely on the connectivities. The same applies to the carrier molecules listed in the third section.

The result of specifying this list while keeping the connectivity cutoff value at 8 is shown in Figure 4C.
Comparing the two strategies, merely reducing the connectivity cutoff was quite successful in this example; however, this is largely due to the fact that, as pointed out by [Yus et al., 2009], the \textit{M. Pneum.} network is not only small, but also quite linear. This is evidenced by a small average connectivity degree and a small proportion of branching nodes compared to other model bacteria. Generally, reducing the cutoff to the extent needed to resolve very large blocks produces severe fragmentation. This happened to some extent even here; with only 73 internal metabolites remaining, the square root rule suggests 8 or 9 subnets, rather than the 26 actually obtained.

Using the more targeted externals list instead, 90 internal metabolites remained, indicating that fewer mass balance constraints were needed to be sacrificed. Only a few blocks separated at the initial stage, which allows for more user control by spreading the separation over several interactive rounds. How that proceeds will be demonstrated in Figure 7.

The use of an additional input file giving target metabolites is demonstrated by considering as an example the case that we are specifically interested in the metabolism of pyrimidine nucleotides. An appropriate input file for this is shown in Figure 6. In this case, cytosine is not at all present in the network; listing it as a target does no harm, as it will just be ignored by \textsc{Netsplitter}. If there are additional areas of interest, additional files can similarly be prepared as they can be loaded at will (one at a time) during a \textsc{Netsplitter} run. In this file, as in the external metabolites file, only the metabolite IDs need to be specified, the names are optional and are only included here for clarity.

When \textsc{Netsplitter} is run with the three input files (the stoichiometry specification, external metabolites, and targets) and a connectivity cutoff = 6, the progress of the network separation is illustrated in Figure 7. The first round is essentially as in Figure 4C, except that with the slightly smaller cutoff; an additional three metabolites were taken external leaving 87 internals. Moreover, note that the pink stubs indicate that the target metabolites are contained in the large block. At this stage, \textsc{Netsplitter} offers five candidates to be made external, as indicated by the rows highlighted in red and orange shades. Since it is not yet known whether these will eventually be reincorporated, the proposal is accepted and \textsc{Netsplitter} proceeds to the next round, as shown in Figure 7B.
Figure 7: Progressive separation of blocks by deleting selected internal metabolites over the four separator selection rounds represented by Figures (A) to (D).

Note that the total number of internal metabolites shown has reduced from 87 to 75. This is due to the five that were deleted, as well as the fact that the program recognizes that the three small blocks at the top left in Figure 7A contain only a single row or column each and cannot be split further. They are thus set aside for later and not presented to the user in the next round.

This leaves two blocks to be processed in the second round (Figure 7B); five more candidate externals are presented. The smaller block shown in the figure only shows a single black center and does not indicate good potential for further splitting. One strategy allowed by NETSPLITTER would be for the user to select only the large block for further splitting.

However, simply accepting the options as offered, the large block is split into three in the third round (Figure 7C), and four more candidates are proposed. Accepting these, we arrive at the fourth round, the results of which are shown in Figure 7D.

The target compounds have been split over two small blocks. This could have been prevented if the splitting was terminated in the third round, or if only the first three blocks were selected for further processing in the third round. However, the splitting of the target blocks is not really a problem, as the resulting small blocks can be merged back into a combined block later in the merge section of the program.

It is noticeable that the two blocks (Numbers 2 and 6) containing the target metabolites in Figure 7D have no internal structure that will allow further splitting (in fact, each consists of a single row/column). If we are only interested in the target subnetwork, it would make sense to terminate splitting at this point.

On the other hand, assuming that the splitting of the middle block in Figure 7D is still of interest, we proceed to the fourth round by accepting the single candidate offered. Implementing this, no further candidates that could satisfy the criteria built into NETSPLITTER are found, and the splitting process terminates.
Figure 8: Maximal separation into 15 subnetworks

automatically.

At this stage, NETSPLITTER applies the full set of external metabolites collected from all four rounds to the original DAG matrix while reincorporating all those that will not upset the block separation achieved during the external selection rounds.

The final result is displayed in Figure 8. With the total number of internal metabolites increased to 93, the final number of mass balance constraints sacrificed to obtain splitting is a significant improvement on the number resulting from indiscriminate connectivity-based splitting (Figure 4B). Inspection of the corresponding NETSPLITTER printout reveals that, of the 15 metabolites made external during the four externals selection rounds, only 7 survived the reincorporation and became essential separators. In addition, there are 7 metabolites out of the 24 listed in the external metabolites file and 13 high-connectivity metabolites that are retained as externals at this stage.

The figure shows that there are 5 subnetworks of significant sizes ranging from 7 to 19 metabolites, 6 smaller blocks, and 4 orphans (the metabolites 30–33, shown at the bottom of the figure). The total of 15 subnets is somewhat larger than the 9 or 10 that the square root guideline suggests for 93 internal metabolites. Furthermore, a more homogeneous distribution of subnet sizes would be desirable. Merging of some subnets is therefore indicated.

The most obvious candidates for merging are Blocks 10 and 11 (the rightmost blocks in Figure 8; blocks are numbered from the top) since they are identified by the pink highlights as the ones containing the target compounds.

A second clue is obtained from the NETSPLITTER printout. Following on the listings of internal compounds in each block, it contains a section reading:

- **Externals/internals overlap between blocks**
  
<table>
<thead>
<tr>
<th>Block</th>
<th>Overlaps with blocks</th>
<th>And with isolated sinks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>{9}</td>
<td>No overlaps</td>
</tr>
</tbody>
</table>

This shows that some internals of Block 5 appear as externals of Block 9, and vice versa. As explained in the section on merging, this is a strong indication that these subnets should be merged. In the present case, Block 5 is a small block consisting of the three internal metabolites UTP, UDP-glucose, and UDP-galactose while Block 9 is a larger block with 17 internals that, from the presence of choline, diacylglycerol, and fatty acid and their derivatives, is easily recognizable as representing lipid metabolism. It therefore makes good biochemical sense to merge these two blocks.

The next step is to inspect the layout diagram for the orphan subnets. This is shown in Figure 9. It is clear that the subnets for the orphans aspartate and glutamate (Asp and Glu) are linked by common externals and can be sensibly merged, while the other two orphans remain unlinked.
Having performed these three merge operations, one can now inspect the metanet to look for opportunities to consolidate small subnets into larger ones. The layout of the metanet at this stage is shown in Figure 10. Next to the remaining pair of orphans, the smallest are Subnets 2, 3, and 13, with only two internal metabolites each. Of these, only Subnets 2 and 3 show connections to others. In principle, part of the reason may be that the diagram was generated without showing metabolites with connectivities higher than 6. However, NETSPLITTER allows this threshold to be adjusted, and experimenting with higher values does not reveal further connections for the orphans or Subnet 13.

This particular metanet does not show a very compelling case for further merges. Subnet 3 merely shares the production of CO2 with other subnets, which does not seem likely to be significant. Subnets 1 and 2 share a link to PRP, as does the quite small Subnet 4 with five metabolites; Subnets 1 and 4 also share the metabolite R1P. To demonstrate the line of argument, Subnets 1, 2, and 4 were tentatively merged to produce a single subnet of medium size. In larger networks, however, cases often occur where a group of subnets can be identified on the metanet diagram that are closely connected to one another and that can be usefully merged to reduce the complexity of the metanet. In fact, Figure 10 does show that Subnets 7 and 14 are rather closely connected by many common external metabolites; however, as they are already quite large individually, merging would not be desirable. Performing the merge of Subnets 1, 2, and 4 produces a total of 10 subnets, a good number for the 93 final internal metabolites.

All of the manipulations so far relied just on structural aspects of the network. Using those, a separation into eight reasonably sized non-trivial subnets and two orphans are obtained. It is now instructive to inspect the subnets in greater detail to determine if they make sense in terms of biochemical functions.

Table 1 lists the internal metabolites of each subnet, and shows that these can readily be used to associate appropriate functional areas with each subnet. Not all metabolites in well-known pathways such as glycolysis or the pentose phosphate (PPP) are listed because some of them are classified as external. The
listed internals are, however, sufficient to identify the subnets, if necessary, by reference to biochemistry texts.

The naming of functional descriptions are chosen to conform to those used by [Yus et al., 2009], and comparison with their Figure 2 shows a remarkable correspondence with the partitioning of their metabolic map. They list a total of 18 named sections; 17 of which are uniquely contained in one of the constructed subnetworks, as listed in Table 1. There are two minor discrepancies: (i) the single reaction they ascribe to Vitamin B6 metabolism, is missing from the table because it only involves external metabolites and is not a proper subnet, and (ii) the Lipoamide reactions in Block 6 appear to be missing from their figure.

When comparing the results from NETSPLITTER, one should bear in mind that the metabolic map of Yus et al. is pathway-oriented; hence, metabolite nodes are duplicated when they occur in different pathways and only reactions relevant to that pathway are shown in each instance.

The above is different from the approach followed here; each internal metabolite is allocated uniquely to a particular subnet, and all reactions in which it participates are by definition included in the subnet. Only external metabolite nodes are split between subnets. Even for these, full connectivity is shown in the metanet. This provides a much more direct overview of the interconnection structure, albeit at the price of greater visual complexity.

Most subnets in Table 1 contain more than one identified function. In some cases, this is a consequence of the network structure; however, for example in the case of Subnet 15, it is a product of deliberate merging decisions. As originally split (illustrated in Figure 10), each of the three indicated functions is carried by its own smaller subnet. The question of whether lumping these together, as reflected in the table, is sensible is a matter of opinion and of what further purpose is pursued through the splitting process.

While the overall agreement between the purely computational procedure based on network structure
embodied by NETSPLITTER and biochemical wisdom expressed in pathway interpretations is quite remarkable, there are some discrepancies in the details.

For example, the occurrence of NADP+ and NADPH in Block 7 that contains purine metabolism, appears anomalous as these metabolites are not involved in the purine pathways, as described in Figure 2 of Yus et al. In fact, NADP+ and NADPH appear there in three different functional partitions: first, in a redox reaction with thioredoxin in pyrimidine metabolism; second, in reducing methylenetetrahydrofolate in folate metabolism; and finally, in phosphorylation reactions with ATP in nicotinate metabolism. Thioredoxin also plays a role in both purine and pyrimidine metabolism.

Since both thioredoxin and NADP+ and NADPH are internal metabolites, the constraints on the framework under which NETSPLITTER operates dictates that thioredoxin has to be allocated uniquely and then ends up in the purine subnet. As a consequence, NADP+ and NADPH are also incorporated into the same subnet. If this is unacceptable, a straightforward remedy would be to force NADP+ and NADPH as externals by explicitly specifying them as such in the stoichiometry input file. The fact that they were not automatically recognized as such by the program is because (1) there are reactions that both produce and consume NADPH; and (2) in this small network, their connectivities remain low. Biochemical knowledge, however, implies that these molecules are carriers of reduction potential, just as NAD+ and NADH that are automatically made external. Hence, it makes sense that explicit use of this knowledge improves the

### Table 1: Functional associations of subnets implied by their internal metabolites. Abbreviations for metabolite names are taken from [Yus et al., 2009]. Subnet numbering is as used by NETSPLITTER (e.g., Subnet 12 was produced by merging Subnets 5 and 9, so that those numbers become vacant). Orphan subnets are not numbered but identified by their single internal metabolite.

<table>
<thead>
<tr>
<th>Subnet</th>
<th>Internal metabolites</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>CBP, Cir</td>
<td>Arginine fermentation</td>
</tr>
<tr>
<td>6</td>
<td>AC-P, AcAld, AcCoA, dR5P, Lia, DHLA, SAcDHLiA</td>
<td>Pyruvate; Lipoamide</td>
</tr>
<tr>
<td>7</td>
<td>Guo, Gua, dATP, dGTP, 2GP, 3GP, dAMP, dADP, dGMP, dGDP, NADP+, NADPH, redThio, oxThio, CDP, DGP, GDP, GMP, GTP</td>
<td>Purine nucleotides</td>
</tr>
<tr>
<td>8</td>
<td>GLY, DHA, DHAP, Mt1P, G6P, M6P, 3dG6P, FBP, A6P, GA, F1P, proHis, pro-NpPHis</td>
<td>Glycolysis; Carbohydrates; Ascorbate</td>
</tr>
<tr>
<td>12</td>
<td>Ado3,5P, dPCoA, PC, pGLY, UDP-GAL, 1aG3P, ACP, ACP-R, apoACP, CDP-DAG, CHO, DAG, FA, G3PC, PAC, PAN4P, pG3P, snG3P, UDP-GLC, UTP</td>
<td>Lipids; Choline; CoA</td>
</tr>
<tr>
<td>13</td>
<td>Asp, Glu</td>
<td>Amino acid</td>
</tr>
<tr>
<td>14</td>
<td>THF, 10fTHF, dTPP, Ser, UMP, Ura, Ur, Cyd, dCMP, dCyd, DHF, dTDP, dTMP, dUMP, dUrd, FOR, Thd</td>
<td>Pyrimidine nucleotides; Folate</td>
</tr>
<tr>
<td>15</td>
<td>RIB, R5P, dRu5P, E4P, S7P, NACD, dNAD, Ade, Ado, SAH, SAM</td>
<td>Pentose phosphate; Nicotinate; Methionine</td>
</tr>
<tr>
<td>FMN</td>
<td>FMN</td>
<td>Riboflavin</td>
</tr>
<tr>
<td>TPS</td>
<td>TPS</td>
<td>Thiamine</td>
</tr>
</tbody>
</table>
This example illustrates that while the computational algorithm generally produces excellent overall results, some direction from the user may be useful to improve the details. It is one of the goals of the NETSPLITTER implementation to provide the means by which such fine-tuning can be done.

There are a few more options and functions available in NETSPLITTER that have not been discussed in detail in this example. For example, once the splitting is complete, all subnets can be saved in individual files, either in a simple tab-separated text format (TSV) or as independent network models in SBML format. At the same time, the current state of the separation is preserved in an automatically created external metabolites file that also stores the configuration and interactive choices made by the user. This can be used to resume a calculation, such as for adding further merge steps. There is also provision to roll back previous merge steps using this file. There is also an option to simply convert an input model between the TSV and SBML formats. This has been used to produce the *M. Pneumoniae* input file in SBML format as supplied in the supplementary material.

One available option that merits further discussion is the treatment of reversible reactions. The example presented above used the default option of ignoring reversibility (i.e. all reactions are treated as unidirectional in the direction implied by the stoichiometry matrix). The data file used specifies a large fraction, 108 of the total of 189 reactions, as reversible. Repeating the calculation described but with the option to “expand reversible reactions” chosen, results in much inferior block separation leaving a large unresolved block with 79 internal metabolites. Neither the further reduction of the maximum connectivity threshold nor expansion of the external metabolites file improves matters in this case. This is quite typical of networks where a substantial number of reactions are reversible. Experience shows that expanding reversible reactions is successful only when the fraction of reversible reactions is small, or additional direction information can be supplied through the optional additional input file listing flux values from FBA calculation.

6 Conclusion

NETSPLITTER is a generally applicable program that processes a metabolic network specification and presents it in an intelligible form as a collection of subnetworks. Each subnetwork contains a subset of reactions and pathways in the same interconnected context that they have in the full network. Moreover, the connections between subnets are also presented in an overview form as a metanetwork in which each subnet appears as a “super reaction.”

The running example presented in detail showed that the computational partitioning of a simple bacterial network closely reflects the established biochemical interpretation of the same network. It also demonstrated various ways in which the outcome can be tailored and fine-tuned to suit individual applications.

The example network for *M. Pneumoniae* was not a particular challenge to the algorithm because of its small size (189 reactions × 229 metabolites) and the fact that it has been shown elsewhere to have a relatively straightforward linear structure. However, the program has also been applied to full-scale eukaryotic networks, such as for a single cell algae (*Chlamydomonas Reinhardtii*, 1703 reactions × 1820 metabolites) and a mammal (*Mus Musculus*, 2016 reactions × 2158 metabolites). The calculation time increased from a few seconds for *Mycoplasma Pneumoniae* to approximately five minutes for the mouse. However, using the same strategies and manipulations described in the example produces a similar level of simplification into subnetworks. Typically, the number of subnets increases to 60–80, and the individual size increases accordingly to an average of approximately 30, which still achieves the goal of an intermediate complexity level that displays pathways in their interconnected context without the overwhelming complexity of the full network.
7 Online Resources

The online version of this book, including figures in full color, is available on iConcept Press Website. The NETSPLITTER program is available for free download for non-commercial use [Verwoerd, 2010a] in the form of a Mathematica notebook. Its use requires installation of Mathematica version 6 or higher [Wolfram Research, 2008] or a corresponding version of Mathematica Player Pro. Both programs are available commercially from Wolfram Research, Inc.

Copies of the input files used for the running example as well as a set of AWK script files referred to in this chapter that are useful for extracting metabolic network specification from public databases are also available for download from the publisher’s website for this book.

References