Chapter 3

Exact frequent itemset model

The methods to predict protein complexes described in the last chapter use a protein-protein interaction graph as an intermediate representation. Using such an intermediate, non-probabilistic representation necessarily eliminates some information contained in the original observation. This is particularly troublesome as research indicates that observations from purification experiments differ substantially from those obtained from two-hybrid experiments [21]. It is thus likely that purification experiments are subject to substantial observational error. To derive good models despite this error, we should use all available information. We were thus inspired to explore techniques that derive the desired model of protein complexes directly from the observation.

Furthermore, it is known that protein complex assembly involves cooperative binding, leading to the same proteins appearing as member of multiple different clusters. More specifically, protein complexes may have shared subunits and consist of core components with additional more dynamic factors [14, 15]. Gavin and Superti-Furga [16] suggest that protein complexes consist of several versatile modules, each acting as "molecular machines" with different functional modules that contribute to one superimposed cellular task. The recent experiment of Gavin et al. [14] reveals modularity of protein complexes organized into core modules and shared components. These more complex relationships between proteins cannot be described by pairwise protein interactions represented in protein-protein interaction graphs.

A simple technique to determine overlapping protein clusters, which makes no assumptions about the nature of protein interactions, is counting the occurrence of protein clusters in the purification data [21]. Finding sets of items in the data that frequently appear together is known as the

frequent itemset problem. We will first ignore experimental error and find exact frequent itemsets given a database of protein purifications. We will describe the standard algorithm to discover such itemsets. We can show that by ignoring random errors, statistical significance can be computed. In Chapter 4, we will extend our solution to handle random experimental error.

3.1 Problem setting

Let $\mathbb{P} = \{i_1, i_2, i_3, \ldots, i_N\}$ be a set of N distinct items. In our setting, items are proteins and consequently an itemset represents a set of proteins. A *transaction* T is a set of items in $\mathbb{P}$, in our case the result of a single purification experiment. A database D of size M is a set of M transactions. A set, $I \subseteq \mathbb{P}$, of items is called an *itemset.* The items in an itemset I are assumed to have some domain dependent total order. In this case, we map N proteins to a set of integers $\{1, 2, 3, \ldots, N\}$. Then the items are stored in a numerical order. The number of items in an itemset is called the *length* of an itemset. Itemsets of some length k are referred to as k-itemsets.

A transaction T is said to *support* an itemset $X \subseteq \mathbb{P}$ iff it contains all items of X, i.e., $X \subseteq T$. The fraction of the transactions in D that support X is called the *support* of X, denoted as $\mathrm{support}(X)$. A user-defined *minimum support threshold* is a real number in $[0, 1]$. An itemset is *frequent* iff its fraction of support is above the minimum support. Otherwise, it is *infrequent*. The goal of frequent itemsets mining is to find all *frequent* itemsets given a database D and a minimum support threshold σ. A general introduction to frequent itemsets mining from a large database can be found in [35].

Definition 1 *Let $\mathbb{P} = \{i_1, i_2, i_3, \ldots, i_N\}$ be a set of N distinct items. A transaction T is a set of items in $\mathbb{P}$.*

Definition 2 *A database D of size M is a set of M transactions.*

Definition 3 *A set, $I \subseteq \mathbb{P}$, of items is called an itemset.*

Definition 4 *The number of items in an itemset X is the cardinality of X, denoted by $|X|$. Itemsets of some cardinality k are referred to as k-itemsets.*

Id	transaction
1	$\{1, 2, 3, 4\}$
2	$\{1, 2, 3, 5\}$
3	$\{2, 3\}$
4	$\{1, 2, 3\}$

Table 3.1: An example database with five items.

Definition 5 $\mathrm{support}(X)$ *is the fraction of the transactions in* D *that support the itemset* X*. An itemset* X *is frequent iff* $\mathrm{support}(X)$ *is above the minimum user-defined support* σ*.*

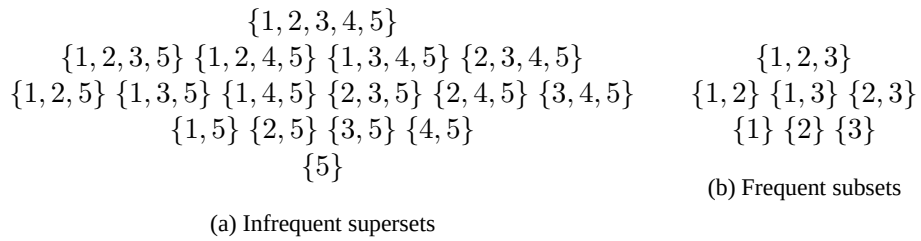
Example See Table 3.1. There are five distinct items, $\{1, 2, 3, 4, 5\}$ in the database. There are four transactions. If the minimum support is set to 0.5, then the frequent itemsets are $\{1\}$, $\{2\}$, $\{3, \}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$ and $\{1, 2, 3\}$, because they occur in at least half of the database.

3.1.1 Maximal frequent itemsets (MFI)

Among all the frequent itemsets, some will satisfy the property that they have no proper superset that are themselves frequent. Such itemsets are *maximal frequent* itemsets. In other words, an itemset is a maximal frequent itemset if and only if it is frequent and no proper superset of it is frequent. An itemset is frequent iff it is a subset of a maximal frequent itemset. The maximum frequent sets (or MFS) is the set of all the maximal frequent itemsets.

Example See Table 3.1. When the minimum support is set to 0.5, the itemset $\{1, 2, 3\}$ is a maximal frequent itemset, because it is frequent and no proper superset of it is frequent. The MFS is $\{\{1, 2, 3\}\}$. In general, there will be more than one maximal frequent itemset in the MFS.

Because an itemset is frequent iff it is a subset of a maximal frequent itemset, an algorithm for discovering frequent itemsets can find the MFS first and then generate the subsets of the MFS and count their supports by reading the database once. Therefore, the problem of discovering the exact frequent sets can be transformed into the problem of discovering the maximum frequent sets. The maximum frequent sets form a border between frequent and infrequent sets. Once the maximum frequent sets are known, the frequent and infrequent sets are known.



(a) Infrequent supersets

(b) Frequent subsets

Figure 3.1: Illustration of monotonicity property. Given the database in Table 3.1 and a support threshold of 0.5, an itemset $\{5\}$ is infrequent and so are its supersets. An itemset $\{1, 2, 3\}$ is (maximal) frequent and so are its subsets.

3.2 Monotonicity property

A typical algorithm to discover frequent sets involves classifying candidate itemsets into three types of sets:

1. *frequent:* this is the set of those itemsets that have been found as frequent,
2. *infrequent:* this is the set of those itemsets that have been found as infrequent,
3. *unclassified:* this is the set of all other itemsets.

Initially, the frequent and the infrequent sets are empty. Throughout the execution, a collection of frequent sets or infrequent sets grow monotonically, while the number of the unclassified sets shrinks. The algorithm terminates when all maximal frequent itemsets are found.

Definition 6 *(monotonicity property) If* $A \subseteq B$, $\text{support}(A) \geq \text{support}(B)$.

From the monotonicity property (Definition 6), two closure properties are true at all time:

- If an itemset is infrequent, all of its supersets must be infrequent, and
- If an itemset is frequent, all of its subsets must be frequent.

Example See Figure 3.1. Consider the database as shown in Table 3.1. The itemset $\{5\}$ is infrequent and therefore $\{2, 5\}$ must also be infrequent, because of the monotonicity of support, $\text{support}(\{2, 5\}) \leq \text{support}(\{5\})$. In other words, there will be an equal or smaller number of transactions containing both items 2 and 5 than those containing 5. Conversely, if the itemset $\{1, 2, 3\}$ is frequent, then itemset $\{1, 2\}$ must be frequent.

3.3 The Apriori algorithm

The most common approach for discovering frequent itemsets is the Apriori algorithm [1]. It consists of iteratively finding frequent itemsets of length $(k+1)$ from frequent itemsets of length k. The algorithm is described in Algorithm 3.

During the candidate generation, candidates of length $(k+1)$ are generated by joining frequent itemsets of length k with frequent singletons. Due to the monotonicity property (Definition 6), exhaustive enumeration of all frequent sets is not needed. A candidate itemset is immediately rejected if its support is less than the support threshold. Otherwise, a candidate is accepted. This is valid because from the monotonicity property we know that if a set is infrequent, all of its supersets must be infrequent and no further examination is needed.

Although the worst case running time is still exponential, because there could be the exponential number of frequent itemsets when the minimum support is low and frequent itemsets can be very long, on average, the Apriori algorithm is known to be fast and easy to implement [1].

Algorithm 3: Apriori algorithm

Input : A database D and a minimum support σ
Output : All frequent itemsets with $\text{support} \geq \sigma$

$C_1 \leftarrow \{\{i\} : i \in \mathbb{P}, \text{support}(i) \geq \sigma\}$
$L_1 \leftarrow C_1$
$FS \leftarrow L_1$
$k = 1$
while $L_k \neq \emptyset$ **do**
  // generate new candidates
  $C_{k+1} = \emptyset$;
  foreach $u \in L_k$ **do**
    foreach $s \in L_1$ **do**
      $v = u \cup s$; // join with singletons
      if $|v| == (k+1)$ **then**
        $C_{k+1} \leftarrow C_{k+1} \cup v$;
  // prune candidates
  $L_{k+1} = \emptyset$;
  foreach $u \in C_{k+1}$ **do**
    if $\text{support}(u) \geq \sigma$ **then**
      $L_{k+1} \leftarrow L_{k+1} \cup u$;
  $FS \leftarrow FS \cup L_{k+1}$;
  $k = k + 1$;
return FS

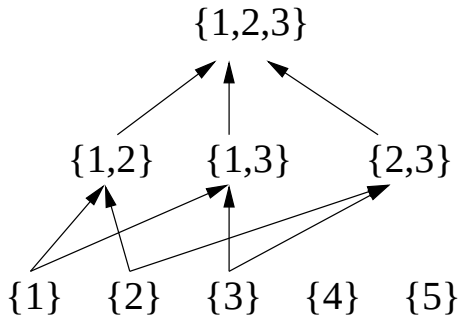


Figure 3.2: Illustration of the Apriori algorithm on Table 3.1 with minimum support 0.5. Frequent candidates are ordered by the length from bottom to top. For each length $k+1$, new candidates are generated from candidates generated at the length k. In this example, $\{4\}$ and $\{5\}$ are eliminated at the first iteration because they are infrequent.

3.4 Integer linear programming formulation

A variant of the frequent itemsets problem is to discover only the *single most frequent* itemset. This problem can be elegantly formulated as an Integer Linear Programming (ILP) problem. Wang and Wu [48] solve the approximate inverse frequent itemset using ILP. Their goal is to generate a synthetic data set that satisfies a collection of constraints on frequent itemsets. The constraints are specified as sets and their frequency. The problem of generating synthetic basket datasets from frequent itemsets is generally referred to as inverse frequent itemset mining. This problem is shown to be NP-complete [48].

Based on the inverse problem formulated by Wang and Wu [48], we can formulate the most frequent itemset as integer-linear programming. Given a database D of N items having M transactions represented as a binary matrix; o_i represents the ith row and e_j represents the jth item,

$$D_{ij} = \begin{cases} 1 & : \quad e_j \in o_i \\ 0 & : \quad otherwise. \end{cases}$$

We want to find the most frequent itemset I of a given size $|I|$ where $I_j = 1$ if $e_j \in I$. We convert this problem to ILP as follows

$$I \in \{0,1\}^N$$

$$y_i = \begin{cases} 1 & : \quad I \subseteq o_i \\ 0 & : \quad otherwise \end{cases}$$

maximize $\sum_{i=1}^{M} y_i$ subject to contraints

$$|I|y_i \leq \sum_{j=1}^{N} D_{ij} I_j \leq y_i + |I| - 1.$$

3.5 Significance of exact frequent itemsets

After we have discovered all frequent itemsets using the Apriori algorithm, we also would like to know if the discovered frequent itemsets are statistically significant.

We developed a simple background model to differentiate significant itemsets. For the background model, we assume that each item (protein) k is selected with probability p_k. p_k can be estimated from the data by setting $p_k = \frac{\text{Freq}(k)}{M}$, where $\text{Freq}(k)$ is equal to the number of occurrences of k in M transactions. Under the random model, if each item occurs independently of others in a transaction, the expectation of an itemset $I \subseteq \mathbb{P}$ is defined as

$$\text{E}[I] = \prod_{k \in I} p_k.$$

The actual number of occurrences, computed with the Apriori algorithm, is $\text{support}(I)$. Let h_0 be a hypothesis that an itemset I occurs at random according to the model and h_1 be a hypothesis that the itemset I is significantly frequent. To detect if the frequent itemset I is not likely to occur by chance, we compute the log-ratio as follows

$$\log \frac{h_1}{h_0} = \log \frac{\text{support}(I)}{\text{E}[I]}.$$

We can also compute the probability that an itemset of size r occurs k times, given a database of M transactions and N items. Assume that each protein is chosen independently of any others with probability p. Then the probability of a set of r items is equal to

$$q = p^r (1-p)^{N-r},$$

and the probability that an itemset I of size r occurs k times, given a database of M transactions and N items is equal to

$$\mathbf{P}[\text{support}(I) = \frac{k}{M}] = \binom{M}{k} q^k (1-q)^{M-k}.$$

Unsurprisingly, the larger the itemset, the fewer occurences of it make it statistically significant. The probability of it occurring randomly will become vanishingly small as r and k increase. For example, from a typical protein purification experiment, $k = 3$, $N = 1600$, $M = 500$, $p = \frac{1}{N}$ and $r = 3$, this probability is less than 10^{-13}. Thus, a threshold lower than .05 should be required for significance. When taking into account that proteins are not distributed with equal probability in the cell, it can be argued that this probability may not reflect the true significance of frequent modules. Another approach to calculate the significance is by simulating the p-value as done by Hollunder et al. [21]. In their work, a simulated database is generated with a given complex size distribution, and a protein distribution estimated from the data. For this database, frequent itemsets occuring at least twice are computed. For each frequent itemset, its frequency is computed into a p-value. Similar to our result, this p-value computation also shows that long frequent itemsets are always significant (having low p-value).

3.6 Result and discussion

We evaluated our approach on the **Gav02** data set to allow comparison with the curation provided by others [15, 24]. See Chapter 1 for an explanation of the data set and the benchmarks. The data set consists of 586 purifications (transactions). After removing all purifications that were empty, the working database comprised of 454 purifications covering 1358 proteins. Using the terminology of frequent itemsets, it is a database of 454 transactions on 1358 items.

Because the benchmark data set contains overlapping protein complexes, we cannot directly compare our frequent itemsets to the benchmark. However, assuming that all proteins in a frequent itemset interact with one another, we can evaluate the prediction capability of the pairwise interaction by computing specificity and sensitivity against the benchmark. We will refer to an (unordered) pair of proteins from the same complex as a *true* pair, and to a pair of proteins from the same cluster

as a *predicted* pair. We will call a true predicted pair *true positive* (TP), a true pair which has not been predicted *false negative* (FN), a false pair predicted to be from a complex *false positive* (FP) and a false, but not predicted pair *true negative* (TN). The following quantities summarize the performance: *Sensitivity*, $sens = \frac{\#TP}{\#TP+\#FN}$ and *Specificity*, $spec = \frac{\#TP}{\#TP+\#FP}$. A perfect clustering method would have $sens = spec = 1$, which implies that neither FP nor FN errors were made.

We show the comparison to the MIPS and Reguly benchmark in Table 3.2 and 3.3. The accuracy is comparable to the manual solution (**Cellzome**) by Gavin et al. [15] and the heuristic-based solution (**Krause**) by Krause et al. [24]. Since we did not outperform previous methods, we conclude that, despite theoretical advantages over other models, frequent itemsets do not perform very well in practice. As expected, both tables show that at high support values, prediction can be highly accurate, with few false positive errors, but many interactions cannot be identified at such high support levels because there are not enough repeated experiments done on most protein complexes, which leads to many false negative errors. At the minimum support of 9 (2%), only one-complex, Rna14 in Figure 3.4, emerges, and no complex is tested more than 13 times (support of about 3%), the highest support value possible for the Gav02 dataset before no itemsets can be recovered.

By observing the log-ratio (Figure 3.6) with respect to the background model, we find that the significance of frequent itemsets correlates with the length of frequent itemsets: the longer the frequent itemset, the more significant it is. The length of frequent itemsets also correlates with the minimum support values; at lower support values, we can observe long frequent itemsets, but only short ones at higher support values. See Figure 3.6. This implies that large complexes, more than 6 proteins, are difficult to observe. Most complexes that are found are small, less than 5 proteins. Very few discovered complexes have been shown to be larger than 9 proteins. To recover most complexes, we have to reduce the minimum support to 2 or 3 (less than 1%). A positive result is that even at this low support level, the accuracy is still comparable to the Cellzome manual solution; from this, we can conclude that for most protein complexes, the Gav02 data set contains 2 or 3 baits per complex.

To visualize the frequent modules, we construct a pairwise interaction graph. Assuming that all proteins in a frequent itemset interact with one another, they form a complete undirected subgraph of protein interactions. A global view of all frequent modules can be obtained by constructing a pairwise interaction graph from a union of these subgraphs. For example, the Rna14 complex is

Min. support (out of 454)	Sens.	Spec.
2	0.44	0.94
3	0.28	0.98
4	0.2	1.0
5	0.13	1.0
9	0.02	1.0

Table 3.2: **Gav02**: Comparison with the Reguly benchmark. Evaluated on 136 proteins annotated by Reguly et al. This table shows prediction accuracy on interaction pairs at different support values; the accuracy is shown as sensitivity (sens.), specificity (spec.) and number of absolute errors in pairs. There are a total of 388 true positive interaction pairs. **Cellzome**: sens = 0.57, spec = 0.92. **Krause**: sens = 0.41, spec = 0.95.

shown as an interaction graph in Figure 3.4.

For a larger data set with roughly $3,000$ items (proteins) such as the **Gav06** data set, we need to avoid the exponential growth of candidate itemsets by searching *maximal* frequent itemsets. We used an implementation called MAFIA which is an efficient implementation of maximal frequent itemsets (`http://himalaya-tools.sourceforge.net/Mafia/`). The sensitivity and specificity at different minimum support (green line) compared with the method of Krause et al. [24] are shown in Figure 3.3. When evaluating on 898 proteins drawn from the Reguly benchmark and the Gav06 data set, the result shows that the accuracy of frequent itemsets is worse than the previous method of Krause et al. [24] which assumes a partitioning model of clustering and only assigns clusters to bait proteins. Our method has the advantage that it also assigns clusters to prey proteins without creating a protein interaction graph and can predict overlapping protein complexes. However, the prediction accuracy is disappointing due to insufficient data to estimate overlapping protein complexes.

Min. support (out of 454)	Sens.	Spec.
2	0.51	0.83
3	0.37	0.91
4	0.25	0.94
5	0.16	0.90
9	0.03	1.0

Table 3.3: **Gav02**: Comparison with the MIPS benchmark. Evaluated on 176 protein. This table shows prediction accuracy on interaction pairs at different support values; the accuracy is shown as sensitivity (sens.), specificity (spec.) and number of absolute errors in pairs. There are a total of 239 true positive interaction pairs. **Cellzome**: sens = 0.78, spec = 0.78. **Krause**: sens = 0.63, spec = 0.95.

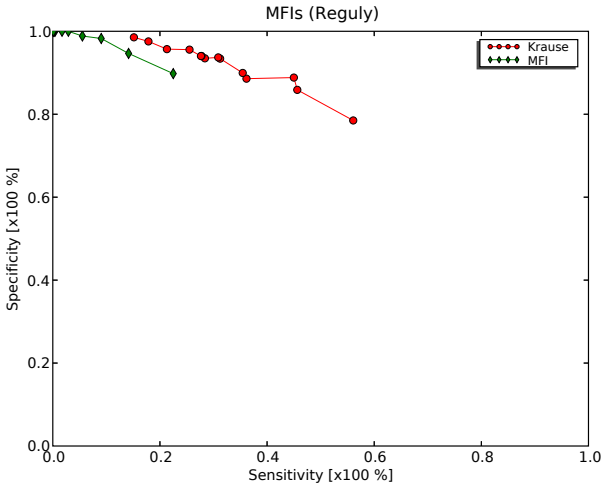


Figure 3.3: **Gav06**: Frequent itemsets compared with the Reguly benchmark. We compare the ROC curve with solution from a method used in Krause et al. [24]. Evaluated on 898 common proteins from the benchmark and the bait-proteins of Gav06 data set. MFI: Maximum Frequent Itemsets.

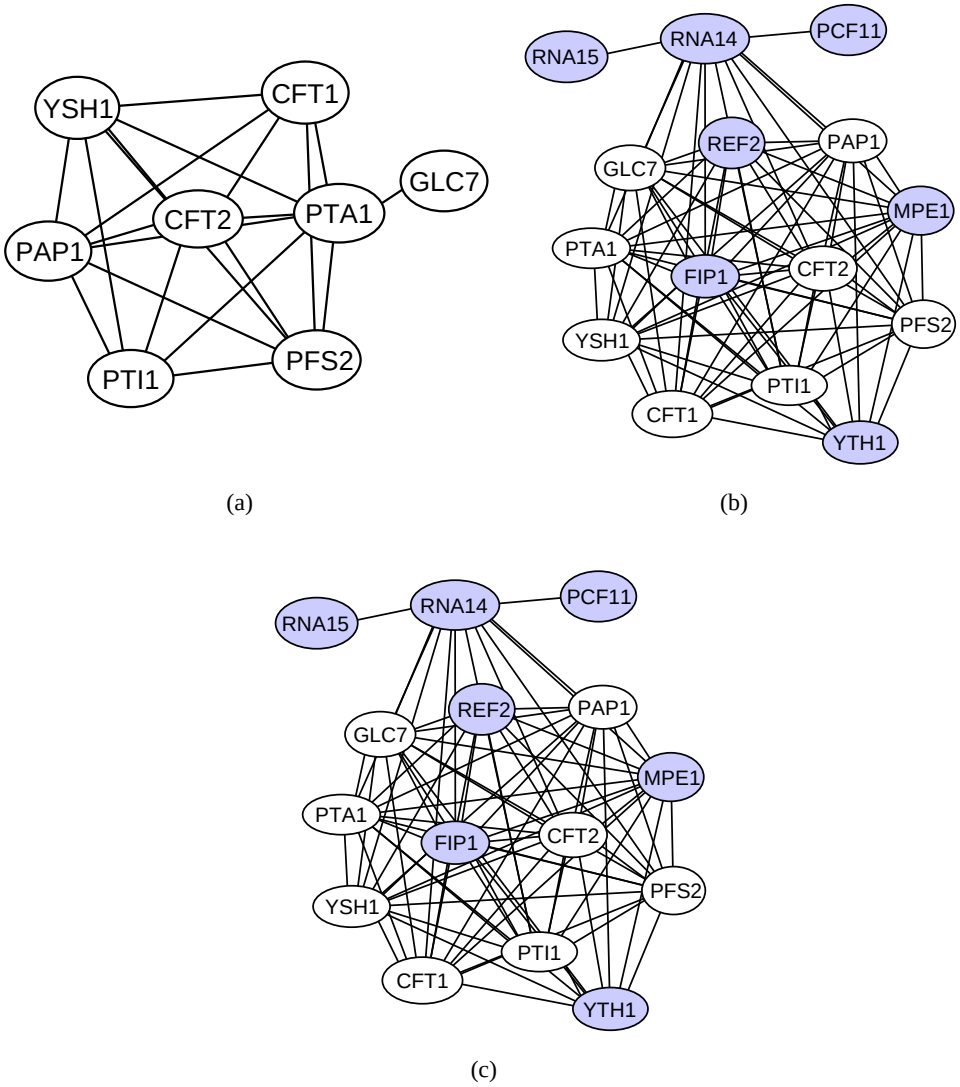
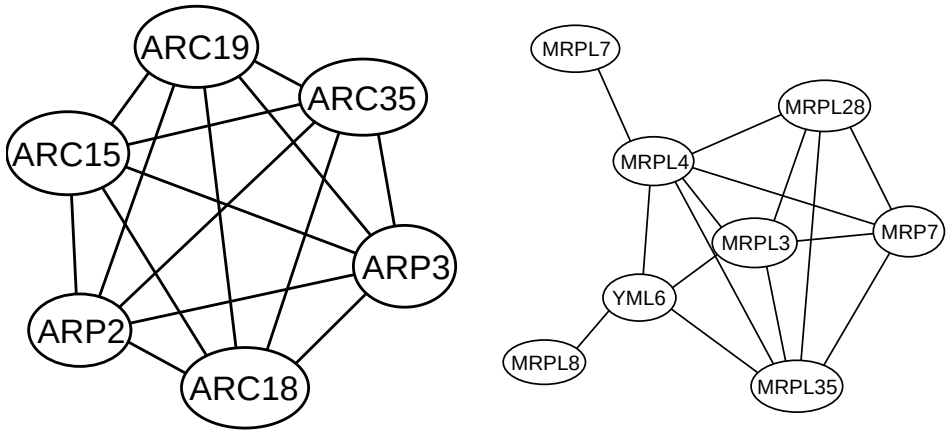
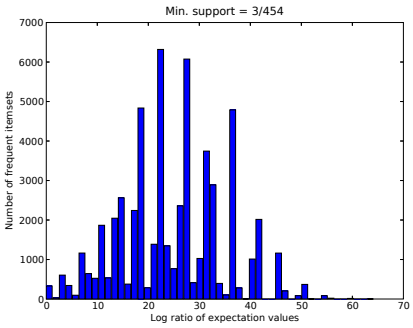


Figure 3.4: The Rna14 complex. (a) A core module of Rna14 complex recovered by setting minimum support to 9 out of 454. (b) At minimum support of 4 out of 454, the RNA14 complex almost reconstructed. Blue nodes indicate proteins included due to applying the lower support threshold. (c) The Rna14 complex organized into frequent modules found by Hollunder et al. [21].

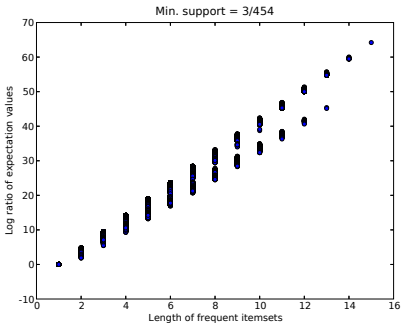


(a) (b)

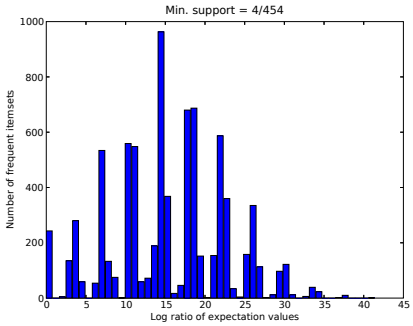
Figure 3.5: Examples of complexes recovered at minimum support of 4. (a) A frequent module found to perfectly match of The Arp2/3 complex. (b) Another module found to consist only of mitochondrial ribosomal proteins (Mrpl).



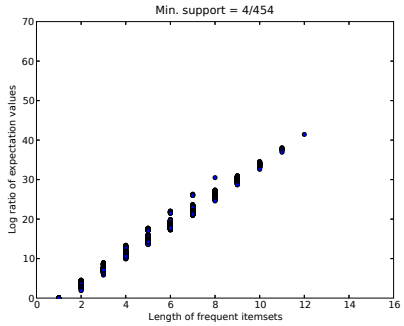
(a) Min. support = $2/454$



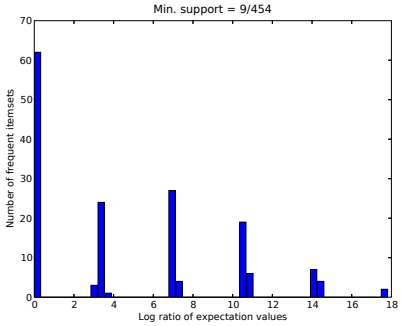
(b) Min. support = $2/454$



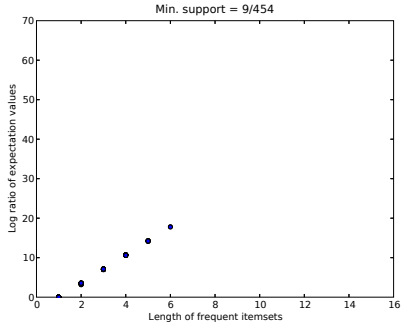
(c) Min. support = $4/454$



(d) Min. support = $4/454$



(e) Min. support = $9/454$



(f) Min. support = $9/454$

Figure 3.6: Comparison with the background model (**Gav02**): (a),(c) and (e) The log-ratio of expectation values with respect to the background model. (b),(d) and (f) Scatter plot; the x-axis is the length of frequent itemsets and the y-axis is the log-ratio.