International Journal on Artificial Intelligence Tools

INTEGRATING MICROARRAY DATA BY CONSENSUS CLUSTERING

VLADIMIR FILKOV
Computer Science Dept., University of California, One Shields Ave.
Davis, CA 95616, USA
filkov@cs.ucdavis.edu

STEVEN SKIENA
Computer Science Dept., SUNY Stony Brook
Stony Brook, NY 11794, USA
skiena@cs.sunysb.edu

With the exploding volume of microarray experiments comes increasing interest in mining repositories of such data. Meaningfully combining results from varied experiments on an equal basis is a challenging task. Here we propose a general method for integrating heterogeneous data sets based on the *consensus clustering* formalism. Our method analyzes source-specific clusterings and identifies a consensus set-partition which is as close as possible to all of them. We develop a general criterion to assess the potential benefit of integrating multiple heterogeneous data sets, i.e. whether the integrated data is more informative than the individual data sets. We apply our methods on two popular sets of microarray data yielding gene classifications of potentially greater interest than could be derived from the analysis of each individual data set.

Keywords: microarray data integration; consensus clustering; median partition heuristics

1. Introduction

Microarray experiments provide measures of gene expression levels under a given set of experimental conditions. Varying the experimental conditions even slightly can potentially reveal differentially expressed genes, or groups of genes with similar expression patterns. As the volume of microarray data grows, there is increasing interest in mining large data repositories. However, meaningfully combining experimental data from diverse sources is a difficult problem.

There have been several previous attempts toward general integration of biological data sets in the computational biology community. Marcotte et al.,[1] for example, give a combined algorithm for protein function prediction based on microarray and phylogeny data, by classifying the genes of the two different data sets separately, and then combining the gene pairwise information into a single data set. Pavlidis et al.[2] use a Support Vector Machine algorithm on similar data sets to predict gene functional classification. Both methods need hand tuning with any particular type of data both prior and during the integration for best results. Ad hoc approaches

to data integration are difficult to formulate and justify, and do not readily scale to large numbers of diverse sources, since different experimental technologies have different types and levels of systematic error. In such an environment, it is not always clear that the integrated product will be more informative than any independent source.

Among general methods for integration and analysis of biological data, clustering is arguably among the most popular and useful. Clustering methods have been particularly successful in organizing microarray data and identifying genes of similar or differential expression.[3,4] We propose a methodology of *consensus clustering* as an approach to integrating diverse sources of similarly clustered microarray data. We seek to exploit the popularity of cluster analysis of biological data by integrating clusterings from existing data sets into a single representative clustering based on pairwise similarities of the clusterings. Under reasonable conditions, the consensus cluster should provide additional information to that of the union of individual data analyses. The goals of consensus clustering are to (1) integrate multiple data sets for ease of inspection, and (2) eliminate the likely noise and incongruencies from the original classifications.

In this paper:

- We develop heuristics for performing consensus clustering, specifically for that of solving the median partition problem on set partitions, that work well in practice on problems with hundreds of data sets (e.g. distinct microarrays) of thousands of entities (e.g. all genes in yeast).
- We investigate the use of consensus clustering for increasing the reliability of microarray gene expression data. We show that the consensus cluster is robust, even when derived from small numbers of independent observations each with up to 25% false classifications.
- There is no guarantee that integrating two different noisy data sets yields a more meaningful result than either data set on its own. We develop a general data-independent criterion for assessment of the efficacy of a consensus clustering.
- To prove the utility of our methods in practice, we integrate collections of various data sets, of both real and simulated data. We show how similar clusterings can be identified by experiment clustering (as opposed to gene clustering), and show that biologically similar experiments indeed cluster together.
- Statistical p-values are essential to evaluate the meaningfulness of a proposed consensus clustering. We propose two classes of p-values which enable us to measure the significance of the amount of agreement between a "gold standard" cluster and a given consensus set partition.

Our paper is organized as follows. In Section 2 we show how to compare clusterings for similarity. The consensus clustering problem is formally described in Section 3, together with an overview of prior work. Heuristics and the theory be-

hind them is given in Section 4. We discuss the implementation and testing of our heuristics in Section 5. In Section 6 we describe the data sets and their clustering. The results of integrating the data sets are given in Section 7. In Section 8 we outline some results on p-values related to our work. We conclude this paper in Section 9 with a discussion and plans for future directions.

2. Comparing Clusterings

Our consensus clustering idea depends on quantifying the pairwise similarity between different clusterings. A clustering is a grouping of objects by similarity, where an object can belong to only one cluster. If we disregard the distances between the clusters, a clustering is equivalent to a *set-partition*.

Formally, a set-partition, π, of $I = \{1, 2, \ldots, n\}$, is a collection of disjunct, non-empty subsets (blocks, clusters) that cover I. We denote the number of blocks in π by $|\pi|$, and indicate them as $B_1, B_2, ..., B_{|\pi|}$. If a pair of different elements belong to the same block of π then they are co-clustered, otherwise they are not.

Many measures of set-partition similarity have been developed and implemented toward solving different classification problems successfully. Among these measures, the simplest and widely used is the Rand index.[5] Others include the Adjusted Rand (correction of Rand for chance) by Hubert and Arabie,[6] the Fawlkes and Mallows measure,[7] and the Jaccard index.[8] All of these measures are based on enumeration of pairs of co-clustered elements in the partitions. Table 1 shows the contingency table for two set-partitions π_1 and π_2 based on the number of matches/mismatches in the co-clustered pairs in the partitions.

Table 1. Contingency Table for Two Partitions.

π_1 / π_2	co-clustered	not co-clustered
co-clustered	a	b
not co-clustered	c	d

Thus, a equals the number of pairs of elements co-clustered in both partitions, b equals the number of pairs of elements co-clustered in π_1, but not in π_2, etc. Hence, a and d are counts of pairs that have been consistently clustered in both partitions (i.e. number of matches), and b and c indicate inconsistencies between the partitions (i.e. number of mismatches). Notice that $a+b+c+d = \binom{n}{2}$, the total number of different pairs of n elements.

The *Rand index* is defined as $R(\pi_1, \pi_2) = (a + d)/\binom{n}{2}$.[5] This is the number of pairwise agreements between the partitions, normalized so that it lies between 0 and 1. The complementary measure, Rand distance, is defined as $1 - R(\pi_1, \pi_2) = (b+c)/\binom{n}{2}$ and it gives the frequency of pairwise disagreements between π_1 and π_2.

For our measure of choice we accept the un-normalized form of the Rand distance

$$d(\pi_1, \pi_2) = b + c\,. \tag{1}$$

This distance measure satisfies the triangle inequality and is equivalent[9] to the symmetric difference between the sets of all co-clustered pairs in π_1 and π_2 (i.e. the symmetric difference of the equivalence relations induced by the partitions).[a]

We note that the symmetric difference distance has a nice property that it is computable in linear time. Namely, $a + b$ and $a + c$ are computable in $O(|\pi_1|)$, $O(|\pi_2|)$ resp., (as the sums of the number of co-clustered pairs in each partition, i.e. $\sum_{i=1}^{|\pi_1|} \binom{B_i}{2}$, and similarly for π_2). Also, it is possible in $O(n)$ time to compute a.[12] Thus, we will consider computing $d(\pi_1, \pi_2)$ an $O(n)$ operation for any two partitions of n elements.

3. Consensus Clustering

A consensus set-partition should be representative of the given set partitions. In terms of similarity it should be close to all given ones, or in terms of distance, it must not be too far from any of them. One way to do this is to find a partition that minimizes the distance to all the other partitions.

CONSENSUS CLUSTERING (CC): *Given k partitions, π_1, π_2, ..., π_k, and $d(.,.)$, the symmetric difference distance on any two partitions, find a consensus partition π that minimizes*

$$S = \sum_{i=1}^{k} d(\pi_i, \pi)\,. \tag{2}$$

This problem is known in the literature as the *median partition problem*. Implicitly, it appears as early as the late 18th century.[13] The first mathematical treatment goes back to Régnier[10] in 1965, where the problem is transformed into an integer program, and a branch and bound solution is proposed. Mirkin[11] offers an interesting axiomatic treatment, using properties of equivalence relations to prove some bounds on the solution. Krivanek and Moravek[14] and Wakabayashi,[15] using different approaches, proved in 1986 that the median partition problem is NP-complete. A nice axiomatic treatise of the problem is given by Barthélemy and Leclerc.[16] Wakabayashi[13] gives an in-depth graph-theory analysis of the median partition problem, from a perspective of relations, and concludes that approximating relations with a transitive one is NP-complete in general.

CC is NP-complete in general, yet it is not known whether it is NP-complete for any particular k. The case of $k = 1$ is trivial, since the partition itself gives an optimal solution. The case of $k = 2$ is also simple: given two set-partitions, any of them solves the problem optimally. Namely, here we want to minimize $d(\pi_1, \pi) + d(\pi, \pi_2)$. The triangle inequality yields: $d(\pi_1, \pi) + d(\pi, \pi_2) \geq d(\pi_1, \pi_2)$, and we obviously achieve this minimum by choosing either π_1 or π_2. Nothing is known for $k \geq 3$.

[a]Régnier[10] and Mirkin[11] were among the first to propose the use of the symmetric difference distance as a dissimilarity measure between two relations, and make attempts at the problem of finding a median relation.

In light of its hardness, exactly solving large instances of **CC** is intractable. Even so, exact methods have been implemented. Tüshaus[17] offers a branch and bound approach, which works only for small n. Wakabayashi[13] and Grötschel and Wakabayashi[18] give a Cutting planes solution, which works well for n in the hundreds.

A variety of approximations like linear programming relaxations of the integer program,[10] Lagrangian Relaxation,[19] hill-climbing heuristics,[10,20] and metaheuristics like taboo search and simulated annealing,[20] have also been applied to this problem. But none of these approximations have yielded any performance guarantees.

4. Heuristics

We present three different heuristics for the median partition problem. In Section 4.1 we present a simple factor-2 approximation algorithm. Then we give two fast, local search heuristics, a greedy and a Simulated Annealing, based on one element moves from one block to another in the median partition. We discuss the implementation in Sec. 5

4.1. *A Factor-2 Approximation*

Algorithm 1. Best-of-k (BOK)
Given an instance of **CC**, select a partition $\pi \in \{\pi_1, \pi_2, \ldots, \pi_k\}$ that minimizes $S = \sum_{i=1}^{k} d(\pi_i, \pi)$.

In other words, this algorithm yields one of the given partitions which minimizes the distance from it to all the others.

Claim 1. Algorithm 1 is factor-2 approximation to problem **CC**.

The proof follows from the following two lemmas.

Lemma 1. *For any instance of **CC**, Algorithm 1 gives a solution at most 2 times larger than the optimal.*

Proof. Let π_{opt} be a partition that minimizes the sum $\sum_{i=1}^{k} d(\pi_i, \pi_{opt})$ in **CC**. For any fixed partition π_j, $1 \leq j \leq k$, let $S_j = \sum_{i=1}^{k} d(\pi_i, \pi_j)$. Further, let $S_{opt} = \sum_{i=1}^{k} d(\pi_i, \pi_{opt})$. Then, because for any pair $\pi_i, \pi_j \in \{\pi_1, \ldots, \pi_k\}$, $d(\cdot, \cdot)$ obeys the triangle inequality $d(\pi_i, \pi_j) \leq d(\pi_i, \pi_{opt}) + d(\pi_{opt}, \pi_j)$, the sums S_j satisfy the following:

$$\begin{aligned} S_j &= \textstyle\sum_{i=1}^{k} d(\pi_i, \pi_j) \leq \sum_{i=1}^{k} [d(\pi_i, \pi_{opt}) + d(\pi_{opt}, \pi_j)] \\ &= S_{opt} + k \cdot d(\pi_{opt}, \pi_j)\,. \end{aligned} \tag{3}$$

If we sum up the above inequalities for all S_j's, we get

$$\sum_{i=1}^{k} S_j \leq k \cdot S_{opt} + k \cdot \sum_{j=1}^{k} d(\pi_{opt}, \pi_j) = k \cdot S_{opt} + k \cdot S_{opt} = k \cdot 2S_{opt}\,, \tag{4}$$

where, on the left side, we have k numbers that add up to at most k times $2S_{opt}$, on the right. Then, from the pigeon-hole principle, it follows that at least one of the k sums S_i must be smaller than $2S_{opt}$. Therefore, the smallest of the sums S_i is never greater than $2S_{opt}$. □

Lemma 2. *For some instances of **CC**, Algorithm 1 produces solutions arbitrarily close to 2 times the optimal solution.*

Proof. We demonstrate a family of instances of **CC**, for which our algorithm can do no better than $2(k-1)/k$ times the optimal. Consider the following k partitions ($n > k$):

$$\begin{aligned}
&\pi_1 = \{\{1,2\},\{3\},\{4\},\ldots,\{n-1\},\{n\}\}\\
&\pi_2 = \{\{1,3\},\{3\},\{4\},\ldots,\{n-1\},\{n\}\}\\
&\cdots\\
&\pi_i = \{\{1,i+1\},\{3\},\{4\},\ldots,\{n-1\},\{n\}\}\\
&\cdots\\
&\pi_k = \{\{1,k+1\},\{3\},\{4\},\ldots,\{n-1\},\{n\}\}\,.
\end{aligned} \tag{5}$$

Since each of the above partitions has only one pair of co-clustered elements, and no elements are co-clustered in any pair of partitions, we have

$$d(\pi_i, \pi_j) = \begin{cases} 2 \text{ if } i \neq j \\ 0 \text{ if } i = j\,. \end{cases} \tag{6}$$

Obviously, whichever partition Algorithm 1 picks will yield a sum of distances $2(k-1)$. But, if we take $\alpha = \{\{1\},\{2\},\ldots,\{n\}\}$ to be a solution instead, the sum of distances from all the partitions to α is k, because $d(\pi_i, \alpha) = 1$, for any i. Since this solution cannot be better than the optimal one, we have that our approximation is never better than $2(k-1)/k$ on the above instance. □

This makes Algorithm 1 a factor-2 approximation. The time complexity of the algorithm is $O(k^2 n)$, since it takes $O(n)$ to compute the distance between any two partitions, and there are $O(k^2)$ pairs.

4.2. *One Element Moves*

Another approach guesses an initial solution for the median, and improves the solution by moving around the elements from one block to another. The initial guess can be random, or, for example, the output of Algorithm 1.

One element moves between blocks are a natural way to move from one to another set-partition. For example moving element 4 from block 1 into block 2 of

$\{\{1,4,5\},\{2,3\}\}$ gives $\{\{1,5\},\{2,3,4\}\}$. If moves into empty blocks, and moves from singleton blocks are allowed, it is clearly possible to get from a given set-partition to any other. Thus, one element moves can be used to explore the set of all set-partitions.

Next, we show how the sum of distances changes after a one element move. We define the characteristic vector of a set-partition, π_p, as: $r_{ijp} = 1$ iff i and j are co-clustered, and $r_{ijp} = 0$ otherwise. Then it can be shown that the symmetric difference distance becomes

$$d(\pi_1, \pi_2) = \sum_{1 \le i < j \le n} (r_{ij1} + r_{ij2} - 2r_{ij1}r_{ij2}) \,. \tag{7}$$

Then, the sum of distances in **CC** of a median partition π, with characteristic vector r_{ij}, to the given partitions $\pi_1, \pi_2, \ldots, \pi_k$ can be written as:

$$S = \sum_{1 \le p \le k} \sum_{1 \le i,j \le n} (r_{ij} + r_{ijp} - 2r_{ij}r_{ijp}) \,, \tag{8}$$

or after some algebra

$$S = \sum_{1 \le i,j \le n} r_{ij}(k - 2 \sum_{1 \le p \le k} r_{ijp}) + \sum_{1 \le i,j \le n} \sum_{1 \le p \le k} r_{ijp} \,. \tag{9}$$

Let us define some useful shorthands at this point. For given i, j the matrix $r_{ij.} = \sum_{1 \le p \le k} r_{ijp}$ counts the number of times i and j are co-clustered in all R_p set-partitions. Obviously $0 \le r_{ij.} \le k$. For convenience we let $K_{ij} = k - 2r_{ij.}$, and note that all K_{ij} are of the same parity as k and range between $-k$ and k. We also let $\sum_{1 \le i,j \le n} \sum_{1 \le p \le k} r_{ijp} = \sum_{1 \le i,j \le n} r_{ij.} = R$. With all the above, our sum becomes

$$S = \sum_{1 \le i,j \le n} r_{ij} K_{ij} + R \,. \tag{10}$$

Let π be our initial guess for a median partition, and let its blocks be $B_1, B_2, \ldots, B_{|\pi|}, B_{|\pi|+1}$, where the last block is the empty block. Then moving one-element, say x, from B_1 to B_2, amounts to transforming the set-partition $\pi = \{B_1, B_2, \ldots, B_{|\pi|+1}\}$ into $\pi' = \{B_1/\{x\}, B_2 \cup \{x\}, \ldots, B_{|\pi|+1}\}$. Since we allow moves into empty blocks too, the total number of blocks to which an element can move is always $|\pi|$.

Thus, if r'_{ij} is the characteristic vector of π' the decrease in S after an element move is

$$\Delta S = \sum_{1 \le i,j \le n} (r_{ij} - r'_{ij}) K_{ij} \,. \tag{11}$$

Because π and π' are very similar it is obvious that most of r_{ij} will be equal to their counterparts r'_{ij}. The following straight forward Lemma quantifies exactly which r_{ij}, r'_{ij} are different.

Lemma 3. *Let π' be the set-partition obtained after moving element x in π to a different block. Also let r_{ij} and r'_{ij} be the characteristic vectors of π and π', respectively, as defined above. Then*

- *if $i \neq x$ and $j \neq x$ then $r_{ij} - r'_{ij} = 0$;*
- *if $i = x$ or $j = x$ and $r_{ij} = 1$ then $r'_{ij} = 0$;*
- *if $i = x$ or $j = x$ and $r'_{ij} = 1$ then $r_{ij} = 0$.*

The following is a direct consequence of the above.

Corollary 1. *Moving x in π from block B_a to B_b, yields*

$$\Delta S_{x:a\to b} = \sum_{\substack{\text{all } j \in B_a \\ j \neq x}} K_{xj} - \sum_{\substack{\text{all } j \in B_b \\ j \neq x}} K_{xj}\,. \tag{12}$$

Thus, to calculate ΔS after moving x into a different block, we need only go through the K_{ij}'s associated with the elements in the blocks where x is moving to and from. The matrix K can be calculated from the $n \times n$ matrices $r_{ij.}$, which in turn can be obtained by pre-processing $\pi_p, 1 \leq p \leq k$ in time $O(n^2k)$. This gives us a fast way, $O(|B_a| + |B_b|)$ steps, to update the sum of distances S after a move of an element.

4.3. *Simulated Annealing*

We use the fast update above to design a Simulated Annealing algorithm for the median partition problem. The target function to minimize is S, the sum of distances. A transition is a random pair of an element $x, 1 \leq x \leq n$, and a block to move it to $block_{to}, 1 \leq block_{to} \leq |\pi + 1|$.

Algorithm 2. Simulated-Annealing-One-element-Move (SAOM)
Given k set-partitions π_p, $p = 1..k$, of n elements,

(1) "Guess" an initial median π
(2) Pre-process the input set-partitions to obtain K_{ij}
(3) SA (trans.: $(x, block_{to})$, function: S)

4.4. *Greedy Algorithm*

An alternative to performing random one element moves is to perform best one element moves, at each step (greedy), toward a better median partition (while such moves exist). We pre-process π into a matrix M, where the rows are the elements of π and the columns its blocks (plus an empty block). An entry M_{ij} is indicative of the cost of moving element i to block j, calculated as follows

$$M_{ij} = \sum_{\substack{\text{all } l \in B_j \\ l \neq i}} K_{il}\,. \tag{13}$$

Here $K_{il} = k - 2r_{il.}$ and B_j is defined as above. We let $M_{i,|\pi|+1} = 0$.

In addition to M, two n element arrays, mb and mv are maintained, where mv_i is equal to the least of all entries M_{ij} in row i, and mb_i is equal to the block (i.e. second index) of the smallest of all entries M_{ij} in row i. M and the vectors mv and mb can be obtained from $r_{ij.}$ and π in $O(n^2)$, since we sum over n values for each element.

The *best one element move* in π is the one that maximizes ΔS, given that $\Delta S > 0$. It is easy to show from Cor. 1 that

$$\Delta S_{x:a->b} = M_{xa} - M_{xb}\,. \tag{14}$$

As M_{xa} is fixed, maximizing ΔS is equivalent to finding the minimal M_{xb}, which is mv_x. Thus, the best one-element move for x is to block mb_x. There are no one element moves left if $M_{i,block_i} - mv_i \leq 0$ for all elements i.

Once the best one element move has been identified it is easy to see that the updates of M, the vectors mv and mb, and finally S can be done in $O(n)$ time.

Algorithm 3. Best-One-element-Move (BOM)
Given k set-partitions π_p, $p = 1..k$, of n elements,

(1) "Guess" an initial median π
(2) Pre-process the input set-partitions and π to obtain K_{ij} and M_{ij}
(3) Until *best one element moves* exist perform them and update S, M, mv, and mb after each.

The correctness of Step 3 of this algorithm follows from Cor. 1 and the fact the the minimal sum of distances (i.e. the optimal solution) is never larger than the sum of distances for the approximation median partitions at any iteration. If a best move exists, one is always found and it brings π closer to the optimal.

The running time consists of two parts: pre-processing time and element move identification and update time. Pre-processing takes $O(n^2k) + O(n^2) = O(n^2k)$ time. Identifying the best one-element move is linear in n. The updates in Step 3 are also linear in n.

5. Implementation

We have developed a general software system, CONPAS (CONsensus PArtitioning System),[9] as a unified framework for handling set-partition data structures and operations on them, as well as data extraction interfaces. In it we implemented the three heuristics above, BOK, BOM, and SAOM. We used the algorithms of Wilf[21] to generate and enumerate set-partitions. Overall the algorithms are very close to real time. In its final version, the slowest of the three heuristics, SAOM, runs in under a minute on an instance of $n = 1000$, $k = 300$, on a 450MHz PII, 192MB RAM PC.

Next, we assess the performance of our heuristics in practice by (1) comparing the overall performance of all heuristics on five simulated and three real data sets,

and (2) analyzing in detail the best heuristic's (SAOM) performance on completely random input, and on meaningful but noisy input. In the next section we describe our studies on real data sets and point out some interesting findings.

5.1. *Comparison of the three heuristics*

In our first study we compare the overall performance of the three heuristics on 7 data sets, five of simulated and two of real data. Preliminary studies indicated that SAOM and BOM do not depend on the number of set-partitions given initially, when that number is higher than 20. Further studies are needed for smaller number of set partitions. Thus, all five simulated data sets, $R_1, \ldots, R_5$ consist of 50 random set-partitions. The number of elements in the set-partition is $n = 10, 50, 100, 200, 500$ respectively. The two real data sets, **yst** and **ccg**, are described in Section 6. The results are shown in Table 2, where the resulting consensus partitions are summarized by the average sum of distances to the given set-partitions, i.e. *Avg. SOD* $= S/(k\binom{n}{2})$.

Table 2. Tests on seven data sets show good performance of SAOM.

Data	n	k	BOK	BOM	SAOM
yst	541	173	0.172	0.165	0.139
ccg	541	73	0.186	0.176	0.139
R_1	10	50	0.173	0.147	0.138
R_2	50	50	0.094	0.086	0.061
R_3	100	50	0.059	0.054	0.036
R_4	200	50	0.313	0.256	0.231
R_5	500	50	0.418	0.399	0.351

BOM was always seeded with the result from BOK, and SAOM with a random partition. Although both SAOM and BOM start from a seed partition, and thus their results are expected to vary with the choice of it, we noticed that overall, SAOM was not dependent on the seed partition. BOM, however, was very dependent on the seed. It invariably produced better results when the seed was closer to the consensus. Its results were poor in general when seeded with a random partition. The SAOM is averaged over 20 runs.

As expected, the hill-climbing, and the simulated annealing did better than best-of-k in every test (there could be a data set, though, for which Best-one-move cannot find a one element move toward a better consensus, thus performing no better than BOK). In all of the 7 test data sets SAOM did better than BOM.

Our conclusion is that SAOM does a good job in bettering the consensus, often better than 20% than BOK, and is therefore our heuristic of choice for solving **CC** in practice.

5.2. *Efficacy of Consensus Clustering*

Next, we did a more detailed study to analyze the behavior of SAOM on random set-partitions. For various n ranging from 50 to 1000, we generated 50 random set-partitions of n elements, and ran SAOM 100 times for each n. The results of the study are reported in Table 3.

Table 3. The dependence of the Consensus Partition Avg. SOD on n is non-monotonic.

n	Avg. SOD	n	Avg. SOD
20	0.1143	250	0.3816
30	0.0864	300	0.4416
40	0.0682	400	0.4327
50	0.0577	500	0.3658
75	0.0423	600	0.3112
100	0.0341	700	0.2719
120	0.2246	800	0.2407
150	0.2142	900	0.2163
200	0.2948	1000	0.1963

It is clear from the table that on random data the Avg. SOD values of the consensus are distributed around means that depend on the number of elements in the set-partitions. The dependence is complex though, most likely because the Symmetric Difference distance is not normalized or corrected for chance. Note the apparent non-monotonicity of the dependence. If the partitions are strictly bi-partitions, then the dependence is strictly monotonous (experimental observation). Thus we suspect that the expected number of blocks in a set-partition of size n is intimately connected to the random symmetric difference distance, although more studies are needed.

This table can be used to tell if the consensus partition is meaningful (i.e. how close (tight) are the data set's partitions). Namely, if the Avg. SOD of a consensus partition of a given profile of set-partitions is close to the average in the table for that n, then we can conclude that the profile of set-partitions are likely too spread out for the consensus to be informative.

5.3. *A Noise Study*

We designed a noise study to assess how well our algorithm eliminates inherent noise in the original clusterings. Given a set-partition, noise is introduced in it by performing one element moves. If n is the number of elements in partition π, and m the number of one element moves, we define the amount of noise as: m/n. An m-noise neighborhood of a partition includes all partitions at most m one element moves away from it. We show elsewhere that one element moves are a good measure of noise.[9]

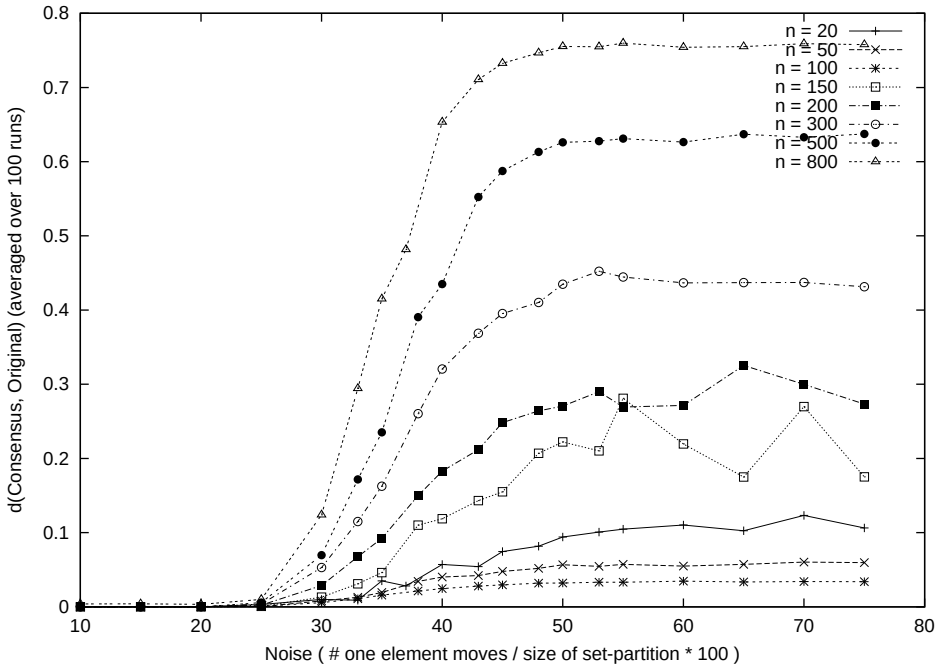

Fig. 1. Performance of the SAOM heuristic under noise. Each point represents 20 runs of the SA, for k = 50 given set-partitions. Evidently, good results are obtained even for noise levels of up to 25%.

To evaluate our algorithm on noisy input, we performed a Monte-Carlo study where for a given level of noise, we generated k random set-partitions in the m-noise neighborhood of an original one, and then calculated their consensus partition. We repeated this 100 times for each noise levels between 10 and 80%, and for different values for n (the size of the set-partitions), and k, their number. The results for the SAOM heuristic are shown in Fig. 1.

6. Data and Clusterings

We applied our methods to expression data of yeast (Saccharomyces cerevisiae). A comprehensive resource for yeast microarray data is the Saccharomyces Genome Database at Stanford[b] where more than 770 experiments on yeast are available (as of February, 2004). Here we will use the popular data sets by Cho et al.,[3] Spellman et al.,[22] and Gasch et al.,[23] all obtained from the SGD. See Filkov et al.[24] for reviews of methods used to analyze the data of the first two.

The Cho et al. and Spellman et al. data sets were produced in studies of cell cycling genes of yeast. The first one consists of 17 microarray plates (17 experiments),

[b]http://genome-www.stanford.edu/Saccharomyces/

of 6601 orfs. The second, consists of three different data sets (alpha, cdc15, and elu) of 18, 24, and 14 experiments, respectively of 6177 orfs. Both of these experiments are known as the cell cycling genes (**ccg**) experiments, consisting of 73 experiments.

In the third experiment, also known as the yeast stress (**yst**) experiment, the responses of 6152 yeast orfs were observed to 173 different stress conditions, like sharp temperature changes, exposure to different chemicals, etc.

To cluster the microarray data sets we used our own clustering tools, developed in earlier studies,[25] based on median link, hierarchical clustering, with the Euclidean metric as the measure of distance. We clustered the yeast ORFs for each experiment in the data sets. We set a threshold of 50 clusters in our clustering tool. The data sets were cleaned of missing values by discarding rows that had any, and the ORFs were matched across all data sets, leaving us with 541 genes and 246 set partitions.

7. Integrating Real Data

The idea of consensus clustering is to combine similar clusterings into one representative clustering. We judge the benefit of an integration by whether the resulting Avg. SOD is close to the Avg. SOD of a similar number of random set-partitions. In Sec. 7.1 we show interesting results of consensus clustering of integrated data sets, when we don't know how close the clusterings are. In Sec. 7.2 we aim to do a better job by first identifying similar clusterings, via clustering of the clusterings, and then finding the consensus within those similar groups of clusterings.

7.1. *All experiments*

First of all, we attempted to integrate the set-partitions from the Spellman et al. data. The combined data set totals 56 conditions (18 in alpha, 24 in cdc15, and 14 in elu).

SAOM was run on each three of the Spellman's subsets, and on the combined set. The resulting Avg. SODs are: alpha, 0.1121, cdc15, 0.1042, elu, 0.1073, and combined, 0.107. We want the Avg. SOD to be as small as possible, thus, integrating the data sets is beneficial for all but cdc15, which appears to be a "tighter" set to begin with. The others benefit from the integration, especially alpha. An interesting thing happens when the Cho et al. data is integrated with the Spellman et al. data. Namely, the 17 set-partitions of Cho et al. yielded a consensus with Avg. SOD of 0.145. When joined to the 56 partitions of Spellman et al. the Avg. SOD dropped to 0.1392. Compared to the above, Spellman's data is tighter than Cho's, and only the second one benefits overall from the integration.

Integrating across studies, we used our heuristic on all 246 set-partitions of **yst** and **ccg**. Individually, they scored Avg. SODs of 0.1771 and 0.1392, respectively, and integrated an Avg. SOD of 0.1659. This goes to say that Gasch's data was not as "tight" as the cell cycling genes data. Again, there was a certain gain from the integration.

Our consensus method is useful in all these cases in that it offers a data independent way to assess whether an integration is beneficial or not.

The resulting consensus can be used both as a strong clustering of the data and as a discovery tool. Here we report on two random examples from the consensus of Spellman's data, illustrative of its general utility in discovery. The following two genes were, as expected, co-clustered: YPR178w, a pre-mRNA splicing factor, and YOL039w, a structural ribosomal protein. Both are involved in transcription. Likewise the following two genes occur together: YHL027w, a meiosis transcription factor, and YLR097c, whose function is unknown.

Although there were some benefits to the integration of the whole data sets, they are not spectacular. This was to be expected as the 246 experiments were different, and thus caused different reactions in the yeast genes. To do a better job at the consensus we must identify similar experiments, i.e. experiments in which the genes were clustered similarly, according to the symmetric difference. We describe one way to do that in the next section.

7.2. *Clustered Experiments*

We used the symmetric difference distance to cluster the clusterings (i.e. identify similar experiments). First off, we built a matrix of distances between all 246 clusterings. Then, we fed this distance matrix into our clustering tool (average link, hierarchical clustering with Euclidean distance as the metric). Again we limited the output to 50 clusters. The elements of each cluster are experiments, named following the original experiments. Thus, **ccg** comprise of *alpha*∗, *cdc*∗, and *elu*∗, and **yst** of the rest. The resulting clusters are given at our web site.[c]

The resulting clusters show without a doubt that biologically similar experiments actually cluster together (most of the cell-cycle experiments are clustered in separate groups from the rest, although there are more than one such clusters).

For each cluster of experiments (i.e. set-partitions), we calculated the consensus clustering. The resulting Avg. SODs show better consensuses, i.e. tighter sets, than the overall integration of the previous section. We noticed that for smaller number of experiments in a cluster, the Avg. SOD becomes larger than the one of all 246 set-partitions. This is due to normalization issues with the Avg. SOD of consensuses of small sets rather than the tightness of the consensus (as mentioned above for $k < 20$ this becomes an issue).

Another interesting thing is that time course experiments from both data sets clustered together and yield very nice consensuses. This, arguably, points toward the utility of microarray data as genome-scale tools for monitoring periodic gene change. Please refer to our website for more insight into the clusters.

The procedure described above is in fact a general method for meaningful microarray data exploration and integration: (1) identify similar experiments, and (2)

[c]http://www.cs.ucdavis.edu/∼filkov/integration/

compute the consensus for the similar groups of experiments. As we see the obtained consensus partitions are likely to be very meaningful biologically.

8. P-values for Set Partitions

A common approach to evaluating the quality of a given clustering of a given data set is to check whether known properties of the items are reflected within the clustering. For example, in clustering microarray gene expression data, we would expect a subset of genes with identical promoter binding sites to appear within the same experimentally-derived cluster. However, without rigorous associated probabilities (p-values) the significance of such observations can be unclear.

8.1. *Fixed Size Partitions*

Our first class of p-values evaluates the likelihood that the given cluster occurs within a part of a random partition of n items into k parts.

We note that the number of such partitions is counted by $\{{n \atop k}\}$, the *Stirling numbers of the second kind*, which are defined by the recurrence

$$\{{n \atop k}\} = k\{{n-1 \atop k}\} + \{{n-1 \atop k}\}. \tag{15}$$

The *Bell numbers* $B(n)$ count the total number of set partitions of n items, and are given by

$$B(n) = \sum_{k=1}^{n} \{{n \atop k}\}. \tag{16}$$

We define $p1(n, k, a, b)$ to be the probability that there exists a part containing at least a members of a predefined subset of b items in a random partition of n items into k parts.

We claim that $p1(n, k, a, b)$ can be efficiently computed for $a > b/2$ using the formula:

$$p1(n, k, a, b) = \frac{\sum_{i=a}^{b} \binom{b}{i} \sum_{j=0}^{n-b} \binom{n-b}{j} \{{n-(i+j) \atop k-1}\}}{\{{n \atop k}\}}. \tag{17}$$

Our argument is based on counting the number of set partitions containing such a prescribed block. When $a > b/2$, such a block can occur at most once in any given partition. The elements of this block consist of $i \geq a$ elements from the subset of b elements and j of the $n - b$ remaining elements. The remaining $n - (i + j)$ elements must form a set partition with $k - 1$ parts.

8.2. *Fixed Shape Partitions*

Our second class of p-values evaluates the likelihood that the given cluster occurs within a part of a random partition of prescribed shape. Note that the the

probability which a given cluster appears within a partition into $n-1$ elements and a singleton set is much larger than a bipartition into equal-sized subsets. By conditioning on the shape of such partitions we can correct for such phenomena.

Let $p = \{p_1, \ldots, p_k\}$ be an integer partition of n, i.e. a multiset of positive integers such $\sum_{i=1}^{k} p_k = n$.

We define $p2(p, a, b)$ to be the probability that there exists a part containing at least a members of a predefined subset of b items in a random set partition of shape p.

We claim that $p2(p, a, b)$ can be efficiently computed using the recursive formula:

$$p2(p,a,b) = \frac{\sum_{i=a}^{b} \binom{b}{i}\binom{n-b}{p_1-i}}{\binom{n}{p}} + \frac{\sum_{i=0}^{a-1} \binom{b}{i}\binom{n-b}{p_1-i} p2(\{p_2,\ldots,p_k\},a,b-i)}{\binom{n}{p}}. \tag{18}$$

where $p2(\{\},a,b) = 0$ for $a > 0$ and $p2(\{\},a,b) = 1$ if $a = 0 \leq b$.

Our argument is based on counting the number of ways that the first block of size p_1 contains the given cluster, by selecting the $i > a$ of b prescribed elements, and $p_1 - i$ non-prescribed elements. If the first block does not contain the requisite number, we count the number of sub-partitions of shape $\{p_2, \ldots, p_k\}$ which do, conditioned on the number of unused prescribed elements.

9. Discussion and Future Work

In this paper we described a combinatorial approach for integrating heterogeneous data based on consensus clustering, and developed very fast heuristics for it. Because of their generality and speed, our algorithms are of independent interest in data mining, classification and consensus theory, especially since they work on large instances. On artificial data the algorithms perform very well, identifying a good consensus even under significant noise. We were able to get interesting results by applying them to real data, although more experiments are needed to assess the real value of the method. In the last section we presented a general method for microarray data integration, hypothesis generation and discovery, similar to two dimensional clustering, but independent of specific data similarity metrics.

We are working to improve our consensus clustering methods in several ways, but mostly to establish a fully autonomous, objective way to assess when data sets can be integrated and yield a more informative consensus than the individual classifications. In a forthcoming paper[26] we compare our best heuristic to a popular heuristic for consensus clustering, *the majority rule*, and conclude that ours betters it. We also follow-up on refining the consensus clustering by using clusters of experiments, as in Sec. 7.2. Finally we address missing data with our method by imputing the missing values based on values of genes with which they are co-clustered in the consensus.

We mention here that there is interest for similar problems in AI and data mining communities. Recently, Strehl and Ghosh[27] addressed a similar problem as ours, which they also call consensus clustering. Their optimization problem considers a different function, one based on information theoretic concepts of commonness between clusters, and their solution is based on different heuristics. Although the methods are different (our methods are independent on the number of clusterings, theirs are insensitive to missing data) it would be interesting to compare them systematically.

References

1. M. Marcotte, M. Pellegrine, M. J. Thompson, T. Yeates, and D. Eisenberg. A combined algorithm for genome wide prediction of protein function. *Nature*, 402:83–86, 1999.
2. P. Pavlidis, J. Weston, J. Cai, and W. Noble. Learning gene functional classifications from multiple data types. *J. Comp. Bio.*, 9:401–411, 2002.
3. R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2:65–73, 1998.
4. M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.*, 85:14863–14868, 1998.
5. W. Rand. Objective criteria for the evaluation of clustering methods. *J. Amer. Stat. Assoc.*, 66(336):846–850, 1971.
6. L. Hubert and P. Arabie. Comparing partitions. *J. Class.*, 2:193–218, 1985.
7. E. Fawlkes and C. Mallows. A method for comparing two hierarchical clusterings. *J. Amer. Stat. Assoc.*, 78:553–584, 1983.
8. M. Downton and T. Brennan. Comparing classifications: An evaluation of several coefficients of partition agreement, June 1980. Paper presented at the meeting of the Classification Society, Boulder, CO.
9. V. Filkov. *Computational Inference of Gene Regulation*. PhD thesis, State University of New York at Stony Brook, 2002.
10. S. Régnier. Sur quelques aspects mathematiques des problemes de classification automatique. *ICC Bull.*, 4:175–191, 1965.
11. B. Mirkin. The problems of approximation in spaces of relations and qualitative data analysis. *Information and Remote Control*, 35:1424–1431, 1974.
12. M. Bender, S. Sethia, and S. Skiena. Efficient data structures for maintaining set partitions. In *Proc. SWAT*, 2000.
13. Y. Wakabayashi. The complexity of computing medians of relations. *Resenhas IME-USP*, 3:323–349, 1998.
14. M. Krivanek and J. Moravek. Hard problems in hierarchical-tree clustering. *Acta Inform.*, 23:311–323, 1986.
15. Y. Wakabayashi. *Aggregation of Binary Relations: Algorithmic and Polyhedral Investigations*. PhD thesis, Universitat Augsburg, 1986.
16. J.-P. Barthélemy and B. Leclerc. The median procedure for partitions. In I. Cox, P. Hansen, and B. Julesz, editors, *Partitioning Data Sets*, volume 19 of *DIMACS Series in Descrete Mathematics*, pages 3–34. AMS, Providence, RI, 1995.
17. U. Tüshaus. *Aggregation Binärer Relationen in des Qualitativen Datenanalyse*, volume 82 of *Mathematical Systems in Economics*. Berlin: Athenäum, 1983.
18. M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem.

Math. Program., 45:59–96, 1989.
19. M. Shader and U. Tüshaus. Ein subgradienten-verfahren zur klassification qualitativer daten. *O.R. Spektrum*, 7:1–5, 1985.
20. S. de Amorim, J.-P. Barthélemy, and C. Ribeiro. Clustering and clique partitioning: Simulated annealing and tabu search approaches. *J. Class.*, 9:17–41, 1992.
21. A. Nijenhuis and H. Wilf. *Combinatorial Algorithms*. Academic Press, 1978.
22. P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Bio. Cell*, 9:3273–3297, 1998.
23. A. Gasch, P. Spellman, C. Kao, O. Carmen-Harel, M. Eisen, G. Storz, D. Botstein, and P. Brown. Genomic expression programs in the response of yeast cells to environment changes. *Mol. Bio. Cell*, 11:4241–4257, 2000.
24. V. Filkov, S. Skiena, and J. Zhi. Analysis techniques for microarray time-series data. *J. Comp. Bio.*, 9:317–330, 2002.
25. T. Chen, V. Filkov, and S. Skiena. Identifying gene regulatory networks from experimental data. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proc. of 3rd Ann. Int. Conf. Comp. Mol. Bio.* , pages 94–103. ACM Press, 1999.
26. V. Filkov, S. Skiena Heterogeneous Data Integration with the Consensus Clustering Formalism In *Proc. of DILS* (to appear), 2004.
27. A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining partitionings. In *Proc. of AAAI*, pages 93–98. AAAI/MIT Press, 2002.