

---

# IP CLUSTERING AS EXPLORATORY DATA ANALYSIS

12/18/2009

Michael Brooks, Katie Kuksenok

Our project focuses on (1) analyzing the Grötschel-Wakabayashi (1989) Integer Programming (IP) formulation of the clustering problem and (2) providing a framework for performing IP clustering tasks that is efficient, informative, and flexible. We discuss related Artificial Intelligence and Operations Research work, as well as its implications for our project. We describe the application architecture and principles motivating its design as well as issues related to the efficiency of the IP approach to the clustering problem.

## THE CLUSTERING PROBLEM

“Clustering” is a general category of problems wherein the goal is to divide a set of data into groups that reveal the emergent structures or properties of the data. Clustering is a method of computational data analysis: the computer is given the data and will output a clustering without any direct indication of the natural structures in the data having been provided by the user. This makes clustering an exploratory analysis technique, since the computer is not given a specific hypothesis to test, but rather is asked to produce a hypothesis about the data itself.

The clustering problem has been defined in several ways. We focus on one such definition:

- **Input:**  $N$  entities and  $M$  features where each entity has a value for each of the features
- **Goal:** Find an optimal partitioning of the entities based on the features

Some additional notes and limitations:

- **Similarity and dissimilarity:** since every entity has a value for every feature, these values are used to decide how similar and different entities are. Clusters should tend to group similar entities, and separate dissimilar ones.
- **Optimality:** given a clustering on a particular data set, a score can be assigned reflecting how well it separates dissimilar entities and how well it groups similar entities. The goal, then, is to find the clustering with the best score.
- **Computation:** In general, the calculation of similarity, dissimilarity, and score given a clustering and a data set is assumed to be a trivial computation, or at least negligible.

There has been an enormous amount of work on clustering problems in different research communities including Operations Research, Artificial Intelligence, and Theory of Computation. The following sections address how this project relates to some of the approaches that have emerged from prior work.

## 1. ALTERNATE CLUSTERING PROBLEMS

This project is concerned with clustering problems where the goal is to divide the entities into distinct partitions. We do not address clustering formulations that do not have a partitioning of entities as a goal; these alternate formulations include hierarchical clustering, where the goal is to arrange the entities into hierarchical groups, and probabilistic (or “fuzzy”) clustering, where the goal is to place the entities into fuzzy sets, with entities being assigned some probability of belonging to a set.

In addition, some algorithms for clustering specify a particular number of clusters into which the entities must be grouped. We will refer to this variant as  $k$ -clustering. With these approaches, a previously known  $k$  number of clusters must be specified before the clustering is attempted. Thus, the goal is: “Find an optimal partitioning of the entities into exactly  $k$  partitions based on the features.”

## 2. THE GRÖTSCHEL-WAKABAYASHI INTEGER PROGRAMMING FORMULATION

This project uses the formulation of the clustering problem as an Integer Program that is given by Grötschel and Wakabayashi in (1). This section provides an overview of how this formulation works.

The approach taken here is to view the entities as nodes in a complete, weighted, undirected graph. The weight of the edge between any two entities is taken to be the output of some similarity or dissimilarity function that takes into account the values of each feature for the two entities. The problem can then be defined as selecting a subset of all of the edges such that (a) the subset selected is a valid clustering and (b) the clustering is optimal.

Defining the clustering problem in this way allows the feature values to be dealt with as a preprocessing step. The meat of finding a solution lies in using the pre-computed edge-weight for every pair of entities to optimize the quality of the clustering. This formulation uses a weight calculation and an objective function that together serve to maximize the agreement on features between entities in the same cluster and to minimize the disagreement of entities in the same cluster. The dissimilarity between every two entities is defined to be the number of features on which they disagree in value, while the similarity is the number of features on which they agree in value. Specifically:

- **Variables:**  $x_{ij} = 1$  if entity  $i$  and entity  $j$  are in the same cluster, otherwise 0
- **Data:**  $w_{ij} = \text{dissimilarity}(i,j) - \text{similarity}(i,j)$
- **Minimize:**  $\text{SUM} \{ w_{ij} x_{ij} : \text{over all } i, j \}$

In order to guarantee that the clustering resulting from the  $x_{ij}$  is a valid clustering, constraints are added. Since a cluster is an equivalence relation which satisfies symmetry, reflexivity, and transitivity for every part of entities, we add constraints for these three properties. Grötschel and Wakabayashi reduce these constraints to the following:

$$\begin{aligned}x_{ij} + x_{jk} - x_{ik} &\leq 1 \\x_{ij} - x_{jk} + x_{ik} &\leq 1 \\-x_{ij} + x_{jk} + x_{ik} &\leq 1 \\\forall 1 \leq i < j < k \leq n\end{aligned}$$

Note that if there are  $n$  entities in the data set, this defines  $O(n^2)$  variables and  $O(n^3)$  constraints.

## 3. INTUITION AND HEURISTICS

Perhaps the most satisfying thing about this IP formulation is that it does not make any assumptions based on intuition or heuristics. For example, intuitively, one might say that clusters should be of about the same size as one another, or look sort of like circular, spherical, or hyper-spherical blobs. These intuitions about what clusters should look like, when used as heuristics in algorithms, impose fundamental restrictions on what clusters those algorithms will produce. Many existing algorithms do use intuitive heuristics, making them more specialized, and shifting them away from exploratory data analysis towards more concrete, problem specific data analysis which is only meaningful when there is some *a priori* understanding of the data. In seeking information on other clustering algorithms, we found a great variety of resources, published primarily in the Artificial Intelligence / Machine Learning communities; however, Xu and Wunsch (2) provide a clear and thorough survey of such algorithms, the most salient sections of which are summarized in this section.

A particular weakness of many heuristic-based clustering algorithms is *high-dimensional* data. The intuition that clusters should be similarly-sized blobs seems sensible when clustering is happening to points in a 2-D or 3-D space; these spaces are easy to visualize and interpret. A clustering algorithm built with this intuition in mind, however, struggles as the number of dimensions grows. In addition, the rising number of dimensions makes the resulting clustering increasingly difficult to interpret or evaluate by inspection. To this end, work has been done to compress high-dimensional data into human-understandable dimensions. There has also been work in reducing dimensions in the original dataset prior to clustering, which addresses the fundamental limitations of such sensitive algorithms. (2)



associated with the learning technique in question. We believe that for this clustering formulation to be most useful, it must be available *via* some “non-expert tool,” which does support “the entire exploratory and iterative process.” We wanted to make something which can, given understandable data (eg., Figure 1), produce an understandable result, such as Figure 3 shows:

|  | Cluster 1 (2)                                 | Cluster 2 (6)                   | Cluster 3 (1)       | Cluster 4 (21)                              |
|--|---|---------------------------------|---------------------|---|
| Aspect of the pelt:                          | without spots, uniformly colored with stripes | with spots                      | with spots          | without spots, uniformly colored with spots |
| Fur:   | short-haired                                  | short-haired long-haired        | short-haired        | short-haired long-haired                    |
| Ears:  | rounded                                       | rounded                         | rounded             | rounded                                     |
| Height (H) up to shoulder:                   | H > 70 cm                                     | H > 70 cm 50 cm < H < 70cm      | H > 70 cm           | H <= 50 cm                                  |
| Weight (W):                                  | W > 80 kg                                     | W > 80 kg 10 kg < W <= 80 kg    | 10 kg < W <= 80 kg  | W <= 10 kg                                  |
| Length (L) of body:                          | L > 150 cm                                    | 80 cm < L <= 150 cm             | 80 cm < L <= 150 cm | L <= 80 cm                                  |
| Length of tail compared with length of body: | median  | long short                      | long                | short median                                |
| (Teeth) canines:                             | very developed                                | very developed                  | little developed    | little developed                            |
| (Larynx) Lingual bone:                       | present                                       | present absent                  | absent              | absent                                      |
| Retractile claws:                            | yes   | yes                             | no                  | yes   |
| Predatory behavior:                          | diurnal nocturnal                             | diurnal and nocturnal nocturnal | diurnal             | diurnal and nocturnal nocturnal             |
| Type of Prey:                                | big prey                                      | big or small prey               | big or small prey   | small prey                                  |
| Climbs trees:                                | no  | yes                             | no                  | yes   |
| Chases after or lies in wait for the prey:   | chase wait                                    | wait                            | chase               | wait  |

FIGURE 3: A FIRST STEP TO AN INFORMATIVE VISUALIZATION OF RESULTING CLUSTERS

Figure 3 presents a preliminary visualization that is produced by an application we wrote as part of this project (Appendix A). While dense, this visualization is more informative than a matrix of 0’s and 1’s, or even a list of cluster entities; though not shown in Figure 3, producing a list of entities in each cluster is certainly possible.

Across the top, we list the cluster number and size. In this case, there were four clusters; the biggest, #4, contained 21 entities (cats), and the smallest, #3, contained 1 cat. At the first glance, this is unsatisfying, as it violates the natural intuition that clusters should have similar size. However, when we consider the features, the names of which are listed down the first column, and their values for each of the entities, the clustering begins to make sense.

The cells contain a list of every single feature value that appears in the cluster; font size reflects the number of entities who have that value for that feature. For example, in cluster #4, for the “Predatory Behavior” feature, “nocturnal,” “diurnal and nocturnal,” and “diurnal” are listed, with different sizes. That means that in cluster #4 there were some cats with each of those values, but there were more cats that were nocturnal than those that were diurnal. Through inspection, we see that the singular cat in cluster #3 really should be there: it is the only cat with no retractile claws; some of its size dimensions agree with those of cats in each of the other clusters, but no one cluster more than another; and so on. In addition, the large cluster #4 contains all the smaller cats that go after smaller prey. We can also see that some features, such as the “aspect of the pelt” feature, do not seem to be important in the clustering, as there is no cluster (other than the trivial #3 cluster) with a majority of the entities sharing any particular value. The opposite seems to be true for the size-related features (“height,” “weight,” etc.).

A conclusion drawn from this clustering could be that many of the features in the dataset are ones that are correlated with cat size, or that clustering this dataset is equivalent to clustering on the basis of size alone. This may, in turn, suggest that more features might result in a different clustering. This interpretation by inspection is very easy with this type of output. Clustering, together with visualization, provides a summary of the data that is otherwise much more difficult to obtain. One of our goals is to create a publicly-available web application that would allow on-the-spot, user-friendly clustering (Appendix A).

## COMPUTATIONAL LIMITATIONS OF CLUSTERING

In addition to ease of interpretation, expedience is of value in an exploratory analysis technique. A user of an application that enables clustering should be able to quickly prepare and run variations of his or her data in order to iteratively explore data; perform sensitivity analysis; etc. However, the general IP problem is notoriously difficult computationally. It would be preferable to instead rely on its polynomial-time LP relaxation.

### 1. WHEN IS THE LP RELAXATION SUFFICIENT?

Since the clustering problem can be reduced to a graph, and since optimally partitioning a generalized graph is an *NP*-hard problem, so is our formulation (5). This means that we probably would not be able to find an IP formulation whose LP relaxation provably produces integer solutions every time.

However, as we tested our application with Grötschel and Wakabayashi's data and several other datasets we collected, we found that the LP relaxation resulted in an integer solution. Although Grötschel and Wakabayashi report having to add cutting plane constraints in addition to those specified in the original formulation to achieve an integer solution in one instance, we were able to obtain integer solutions without additional constraints. This is because the authors added constraints as they ran the program, likely because of limited computational capabilities at the time of the work, whereas we pre-load all constraints before running the LP relaxation solver.

We quickly found an example dataset (Figure 4) that verified that the LP relaxation does not always produce an integer solution. However, the nature of the dataset does suggest that it's possible that the meaninglessness of the LP relaxation result might itself stem from a "natural property" of the data. Figure 4 presents is an 8-entity dataset, where the entire space allowed by the features is filled. Though it could be pictured as a cube, this interpretation is misleading, as the similarity calculation is nominal, not in Euclidean space. Viewed that way, this dataset does not actually lend itself to clustering, as keeping everything in one big cluster makes no more sense than breaking everything into single-entity clusters. This can be explained using a property specific to the clustering problem: *density*, defined as the percent of the potential space specified by the features that is filled by unique entities. In the example dataset in Figure 4, density would be equal to 1.

### 2. DATASET DENSITY AND INTEGER SOLUTIONS

In order to explore potential significance of density as defined, we iteratively tested random datasets on whether or not the resulting solution was integer. We generated 282 random datasets; random numbers came from random.org, which uses atmospheric noise and is superior to deterministic pseudorandom number generators (6). Since we wanted to see if density affected whether integer solution were obtained, we generated various-size datasets of unique entities.

$$S = \prod_{i=1}^M S(i)$$

The size of the space specified by features of a particular dataset is:

where  $S(i)$  is the range of the  $i^{\text{th}}$  feature, and where  $M$  is the total number of features. We used the following algorithm to generate the datasets:

```

for i := 2 ... 10
  for j := 2 ... 10
    for k in {8,16,32,64}
      space =  $i^j$ 
      if space >= k then create k strings of i random numbers, each between 0 and j-1
  
```

```

FEATURES, 3
A, 1, nominal, 0, 1
B, 1, nominal, 0, 1
C, 1, nominal, 0, 1
DATA, 8, numeric
e1, 0, 0, 0
e2, 0, 0, 1
e3, 1, 0, 0
e4, 0, 1, 0
e5, 1, 1, 0
e6, 1, 0, 1
e7, 0, 1, 1
e8, 1, 1, 1
  
```

FIGURE 4: A SMALL DATASET WITH 34 BRANCH & BOUND NODES

Notice that the theoretical space may be larger than the effective space if some possible feature values are not realized in the entities. In analyzing the results, we calculate the effective space by multiplying the ranges of each feature, where the range of a feature is the number of distinct values that appear for it within a particular dataset.

The application we constructed allowed us to automatically solve the LP relaxations for the generated datasets, recording whether or not the solution was integer, and what the solution was. Out of the 282 datasets, 2 had to be aborted since they were taking an inordinate amount of time without any change in the objective value of the LP relaxation. Out of the remaining 280, only 190 had non-trivial clustering results, where trivial means that nothing was clustered with anything else. The trivial runs are included in the figures below for a richer picture of our experiment. Their existence makes sense, as small datasets in large spaces are unlikely to have entities with much in common with each other.

Our results, presented in the two figures on the following page, suggest that density and dataset size are related to whether or not a solution is integer. The second figure represents the data in the first, supplemented with a representation of how long each run took. The table below summarizes the runtime data in the second figure. As we were running only LP relaxations, the relatively long runtime on some occasions is unsettling.

| <b>Runtime statistics (sec.)</b> | <b>Trivial Runtime</b> | <b>Integer Runtime</b> | <b>Non-Integer Runtime</b> | <b>All Runtimes</b> |
|----------------------------------|------------------------|------------------------|----------------------------|---------------------|
| <b>avg</b>                       | 3.511111               | 4.992424               | 35.56897                   | 10.85               |
| <b>min</b>                       | 1                      | 1                      | 1                          | 1                   |
| <b>max</b>                       | 47                     | 45                     | 1256                       | 1256                |
| <b>stddev</b>                    | 7.750672               | 9.989686               | 167.1207                   | 77.02366            |

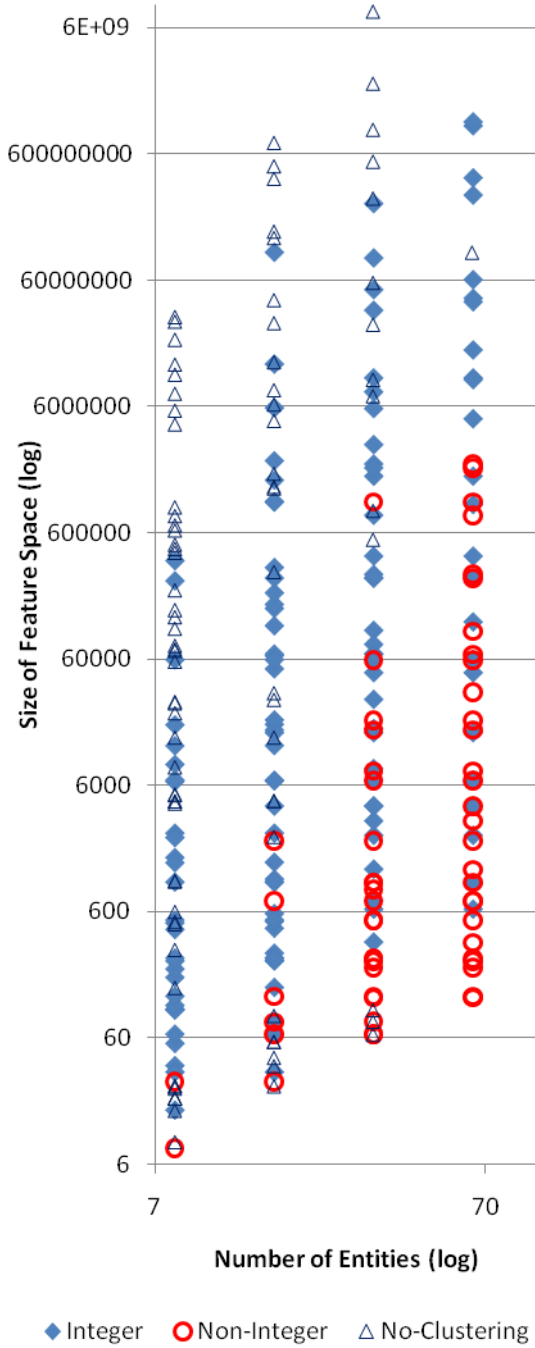
The goal was to see if density, or related measures such as space size, could be used to classify datasets into those that would, and those that would not, produce integer solutions using only the LP relaxation. We believe that properties of the data are important to making this classification, rather than just the number of entities, since the IP formulation has persistent constraints across same-size datasets but different object function coefficients. Although the resulting figures suggest some relationship between overall density and obtaining integer solutions in the LP relaxation, the data also show that there are other factors at work.

A possible problem with this preliminary analysis is that random data, especially truly random data, may not resemble real datasets, at least not ones someone might want to analyze using clustering. However, using anything other than truly random datasets in this kind of analysis is suspect as well.

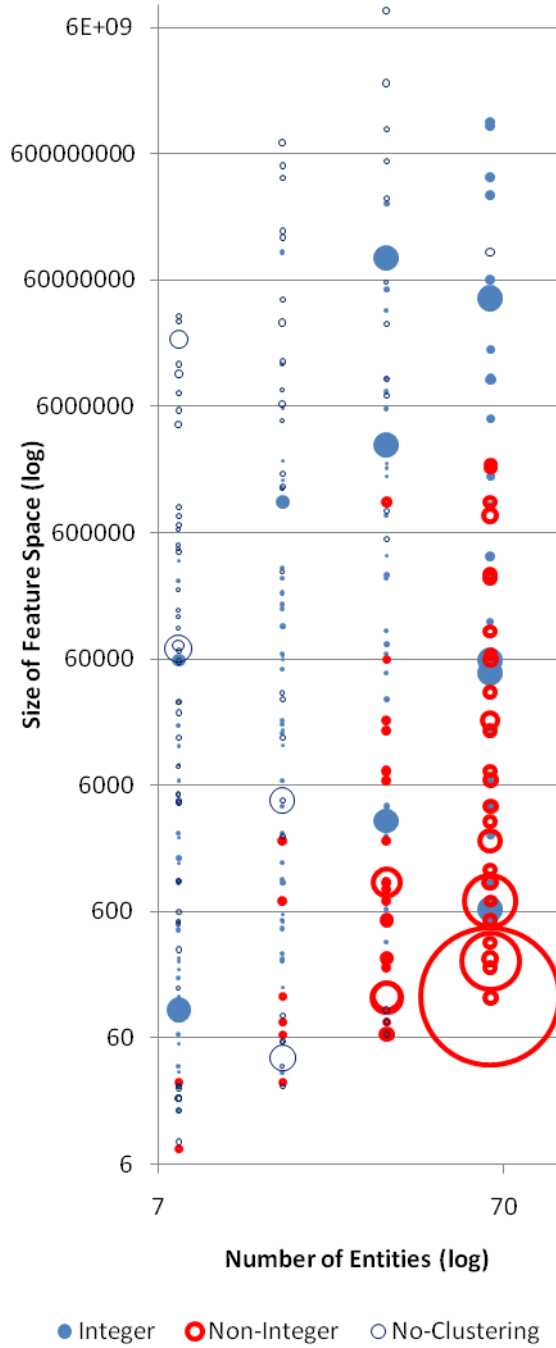
*For raw data and analysis details, please see:*

- <http://ipclusters.kukmbr.com/random/RandomNonZeroDatasets.xls> for the non-zero raw data and results
- <http://ipclusters.kukmbr.com/random/RandomAllDatasets.xls> for all raw data and results
- <http://ipclusters.kukmbr.com/random/RandomAnalysis.xls> for the source of the figures

### Problem distribution, size of feature space, number of entities



### Problem distribution, size of feature space, number of entities (size represents running time)



## EXPANDING THE GRÖTSCHEL-WAKABAYASHI FORMULATION

Since part of our initial goal was to understand and expand the IP formulation, we made sure to build into our application (Appendix A) the capacity for flexibility. We built a program that required a particular file as valid data input. The specifications are described in the table below:

|  |  |
|--|--|
| <b>This section is optional. In it, we can specify a weight for every feature group.</b>   | FEATUREGROUPS,K<br>[followed by K lines of the form as follows, where k identifies each feature group and name gives it a descriptive label:]<br>name of feature group,k   |
| <b>This section is required. It specifies the features in the dataset.</b>   | FEATURES,M<br>[followed by M lines of the form as follows, where name of feature group should be consistent with definitions above, type is binary, ordinal, or nominal, and n is the number of possible, 0-indexed values. Note that the labels specify fully the allowed range of the feature in question.]<br>name of feature,name of feature group,type,label 1,label 2,label 3..... label x |
| <b>This section is required. It lists all the entities. Feature values are specified either by a number, or by a word label.</b> | DATA,N,{numeric/label}<br>[followed by N lines of the form as follows, where values are specified using numbers in the range of the corresponding feature:]<br>name of entity,value 1,value 2,value 3..... value m   |

An example dataset follows in Figure 5:

|    | A                              | B                        | C                 |            |
|----|--------------------------------|--------------------------|-------------------|------------|
| 1  | <b>FEATUREGROUPS</b>           |                          | <b>2</b>          |            |
| 2  | morphological parameters       |                          | 1                 |            |
| 3  | behavioral parameters          |                          | 1                 |            |
| 4  | <b>FEATURES</b>                |                          | <b>14</b>         |            |
| 5  | Aspect of the pelt             | morphological parameters | nominal           | without    |
| 6  | Fur                            | morphological parameters | binary            | short-ha   |
| 7  | Ears                           | morphological parameters | binary            | rounded    |
| 8  | Height (H) up to shoulder      | morphological parameters | ordinal           | H <= 50    |
| 9  | Weight (W)                     | morphological parameters | ordinal           | W <= 10    |
| 10 | Length (L) of body             | morphological parameters | ordinal           | L <= 80 c  |
| 11 | Length of tail compared with   | morphological parameters | ordinal           | short      |
| 12 | (Teeth) canines                | morphological parameters | binary            | little dev |
| 13 | (Larynx) Lingual bone          | morphological parameters | binary            | absent     |
| 14 | Retractile claws               | morphological parameters | binary            | no         |
| 15 | Predatory behavior             | behavioral parameters    | ordinal           | diurnal    |
| 16 | Type of Prey                   | behavioral parameters    | ordinal           | big prey   |
| 17 | Climbs trees                   | behavioral parameters    | binary            | no         |
| 18 | Chases after or lies in wait f | behavioral parameters    | binary            | wait       |
| 19 | <b>DATA</b>                    |                          | <b>30 numeric</b> |            |

FIGURE 5: FEATURE AND FEATUREGROUP DEFINITIONS FOR THE WILD CATS DATASET

### 1. FEATURE GROUPS AND FEATURE GROUP WEIGHTS

In the original Wild Cats dataset, two types of features were listed: morphological parameters and behavioral parameters. In that case, as in many other cases, it may be desirable to separate features into groups. One way in which this separation can be made meaningful in context of clustering is by allowing a user to assign weights to feature groups, effectively making some features more or less important than others. For example, if we wanted to cluster the Wild Cats dataset on the basis of morphological features, with the behavioral features having only a

small influence, we would set feature group weights to 1 and 0.2, respectively. The similarity or dissimilarity measure between entities incorporates the weight factor for each feature.

## 2. VARYING PER-FEATURE SIMILARITY FUNCTIONS

Although using similarity calculations oriented at nominal features “saves” the Grötschel-Wakabayashi formulation from favoring hyper-spherical clusters, it does not seem appropriate for data that is ordinal or scalar in nature. Once again using the original Wild Cats dataset as an example, we notice that while some features, like “aspect of the pelt,” refer to a quality, while others, like “height,” measure a quantity. Qualitative features are more appropriately compared using a nominal similarity function; whereas a distance function seems more sensible for quantitative features. Here, “qualitative” means “possessing or not possessing a quality.” For example, if a cat possesses the quality of being mottled in coloring, it is (arguably) not appropriate to say in this case that it is more spotty than a uniform cat, or a striped cat. It is simply unlike them in its aspect of the pelt.

Since different features may require different distance measures, in the data specifications, we allow each feature to specify its “type,” and which similarity measure to use: “nominal,” “ordinal,” “binary,” and potentially other, custom features. Currently, we have not completely implemented this functionality, and all features are assumed to be nominal in our tests. However, the application architecture is sufficiently flexible to allow this in the future (Appendix A).

## CONCLUDING THOUGHTS

In working on this project, we have laid a solid groundwork for an application enabling clustering as exploratory data analysis. At this point, the application does not have a user interface, but the back end is flexible and modular. Thus, the next step is to finalize the user front-end, and publish the application online for use. We would like to expand the formulation to include other types of data. In addition, after our initial experiment, we would like to test other ideas to attempt to learn – both in a machine learning and theoretical sense – when a dataset will produce an integer solution *via* the LP relaxation.

## WORKS CITED

1. *A cutting plane algorithm for a clustering problem*. **M. Grötschel, Y. Wakabayashi**. s.l. : Springer-Verlag New York, Inc., 1989, Mathematical Programming, pp. Volume 45 , Issue 1, pp. 59 - 96.
2. *Survey of clustering algorithms*. **Rui Xu, D. Wunsch II**. 2005. IEEE Transactions on Neural Networks. pp. 645-678.
3. *A mixed-integer programming approach to the clustering problem with an application in customer segmentation*. **Burcu Saglam, F. Sibel Salman, Serpil Sayin, Metin Turkey**. 173, European Journal of Operations Research : Elsevier, 2009.
4. *Investigating Statistical Machine Learning as a Tool for Software Development*. **Kayur Patel, James Fogarty, James A. Landay, Beverly Harrison**. s.l. : ACM, 2008. CHI.
5. **Skiena, Steven S**. CSE 691 - Lecture 14 - Clustering. *SUNY Stony Brook Department of Computer Science*. [Online] 05 07, 2004. [Cited: 12 15, 2009.] <http://www.cs.sunysb.edu/~skiena/691/lectures/lecture14/lecture14.html>.
6. **Haahr, Mads**. HTTP Interface Description. *Random.org*. [Online] [Cited: 12 17, 2009.] <http://www.random.org/clients/http/>.

## SYSTEM ARCHITECTURE OVERVIEW

The program we built and used is written in Java, and uses LPSolve, a C library and API for linear and mixed-integer programming. Elements of the application are written in PHP. Data exchange occurs via a MySQL database, which maintains a table for every type of input or output. Data sets and other information can also be stored in the file system, but there are some advantages to the using the database.

The fundamental building block for all of the functionality of the application is the clustering dataset itself, which describes the entities and features. This dataset can either be in a format conforming to the specifications listed on page 8, or be a “minimal” dataset: one that contains a line for every entity, with a comma-separated number representing a value for each feature. The distinction between the two formats is human readability. Whereas the first format includes indication of the meaning of the features and values, the second is better suited to automatically-generated datasets that have no meaning or context. For example, the experiment we describe on pages 5-7 involved generating minimal datasets. The data can be saved either in a file and the filename given to the Java program, or it can be stored in the database. The database option is more flexible, since it abstracts the data storage away from the running application itself, simplifying remote tasks. In the experiment we describe, for example, the data was generated on Katie’s personal computer; saved to the database on our web server, kukmbr.com; and then subsequently called to action by the Java program running on Michael’s personal computer.

Once the data is given to the application, it can be manipulated in several ways. First, it is parsed, and an interpretation can be returned in a JSON (JavaScript Object Notation) format. This is a data interchange format which could be used by a web application, for example, to quickly and easily display the dataset without repeating the complicated parsing that the Java program has already done. In addition, an OPL data file can be returned. Because of the details of the formulation itself, only the data file and number of variables differs from dataset to dataset, so this output is sufficient to allow OPL Studio to do the actual Integer Programming. The alternative is to use LPSolve, which is run internally by the Java program. An entry is maintained in the database with live information about the session including the processing stage, such as “loading constraints” or “running LPSolve,” and progress, such as percent completion or current value for the objective function. Many instances can be running concurrently, and their respective progress can be viewed remotely. If the operation is successful, the result is saved in the database. The result of a successful operation is a matrix of variable values. If OPL Studio was used instead of LPSolve, manual insertion of the resulting matrix into the database is possible, as well.

Just as the original data is parsed and interpreted by the program, so can be the solution matrix. The Java program is able to parse and relate both, returning a friendlier and more comprehensive representation of the solution, again in a JSON format. In the visualization of the Wild Cats dataset clustering on page 4, a PHP script did some very basic processing on one such type of JSON output.

The Java program we built is highly modular, and we have been using it in a variety of ways. As a result of its potential power, we want to make an interface that is no less powerful, but cannot seriously harm the various computational sources we are using. For example, the website interface would be hosted on our website, kukmbr.com, but call a PHP script on Computer Science department’s OCCS, which would make a system call to start the Java program, which would then behave as we described, communicating with the original website solely through the database. This setup requires great care on our part to use OCCS as gently as possible, presumably by bottlenecking the number and duration of LPSolve sessions. In addition, the large number of interactions that can take place with the Java program make constructing a comprehensive yet functional interface a design challenge that is beyond the scope of this project as it stands. Nevertheless, we do hope to put the pieces we have been using and improving together in a more intuitive online application.

To download the Java program, please see <http://sites.google.com/site/aclusters/>