20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

Picking Items for Experimental Sets: Measures of Similarity and Methods for Optimisation

N. Boland [a], **R. Bunder** [a], **A. Heathcote** [b]

[a] *School of Mathematical and Physical Sciences, University of Newcastle, Callaghan, NSW, 2308*
[b] *School of Psychology, University of Newcastle, Callaghan, NSW, 2308*
Email: Rachel.Bunder@uon.edu.au

Abstract:

Experimental psychologists often conduct experiments in which subjects are exposed to sets of stimuli. For example, human subjects may be shown a sequence of written words, and their response times recorded in order to understand the effect of one attribute, such as the frequency of the word in spoken language, on human response time. The psychologists designing the experiment will construct several sets of words so that each set contains only words within a specified range of frequencies in the spoken language. To reduce the risk of bias in the experiment, the psychologists would like each set of words selected to be similar in terms of other confounding attributes that could affect response time, such as the number of letters in the word, or the number of syllables. A challenge for the psychologists is that the sets they select may need to contain many words, the words may be selected from a set of thousands, and a large number of potentially confounding attributes may need to be considered. This daunting task, which we dub the problem of *Picking Items for Experimental Sets (PIES)*, is usually performed manually by experimental psychologists.

To assist in this task, both metaheuristic and mixed integer programming (MIP) approaches have recently been developed. Such automated approaches require a systematic definition of "similarity" of sets; the degree to which sets of items are similar with respect to some attribute can no longer be assessed objectively by the psychologist designing the experiment(Forster, 2000). To illustrate this issue, consider two sets of words, B_1 and B_2, where $B_1, B_2 \subseteq W$, the set of words available for selection, and the attribute given by the number of letters in each word. For each word $w \in W$, let f_{wl} denote the number of letters, l, in word w. One approach to measuring the similarity of the two sets is to compare the average value of the attribute across the sets, i.e. measure based on the difference $|\frac{1}{|B_1|}\sum_{w\in B_1} f_{wl} - \frac{1}{|B_2|}\sum_{w\in B_2} f_{wl}|$. However it is well known that very different distributions can have the same average value. For example, defining the attribute value count vectors $\eta_{B_i c} = |\{w \in B_i : f_{wl} = c\}|$ for each $i = 1, 2$ and each positive integer value c that could be the length of a word, say ranging from 1 to 5 letters. This approach would consider two sets with $\eta_{B_1} = (3, 3, 3, 3, 3)$ and $\eta_{B_2} = (0, 0, 15, 0, 0)$ to be very similar, whereas clearly the experience of a human subject to these two sets might be very different: the former has an even spread of word lengths whereas the latter has all words of identical length. The existing metaheuristics address this issue by using group characteristics, such as average or standard deviation, which take into account the relative values of the heuristics. However, as we have shown, these group characteristics do not adequately measure the similarity of the sets.

Recent MIP approaches measure similarity between sets using the entire histogram, i.e. they measure based on the difference $|\eta_{B_1 c} - \eta_{B_2 c}|$ for each c. Whilst this provides a richer measure of similarity than simple averages, it does not take into account the relationships between attribute values. To return to the word length illustration, the length count vectors $(0, 3, 3, 3, 6)$ and $(3, 4, 5, 0, 3)$ are "equally" different from $(3, 3, 3, 3, 3)$ component-wise. But it is common sense that words of length 2 or 3 are more similar to words of length 4 than words of length 1 are to words of length 5, so the vector $(0, 3, 3, 3, 6)$ "replacing" three words of length 1 with three of length 5 is less similar to $(3, 3, 3, 3, 3)$ than is $(3, 4, 5, 0, 3)$, which "replaces" three words of length 4 with two of length 3 and one of length 2. The component-wise histogram measure does not take into account similarities and differences *between* attribute values.

This paper briefly reviews the existing approaches to automate picking items for experimental sets, and then discusses new MIP approaches that address the entire distribution of attribute values across sets while also taking into account the relationships between attribute values. Numerical results on psycholinguistic data sets are analysed, and the alternative approaches compared.

Keywords: *Mixed integer programming, Stimulus selection, Factorial designs, Experimental psychology*

1 BACKGROUND

The ability to create matching experimental sets is needed by many different researchers. There have been three main attempts to produce an automated approach to solving PIES in recent years: Stochastic Optimization of Stimuli (SOS)(Armstrong et al., 2012), Match (Van Casteren et al., 2007) and a mixed integer programming approach (Boland et al., 2013).

SOS is a flexible approach that uses simulated annealing and to find solutions for PIES and statistical tests to evaluate how near optimal these solutions are. As well as selecting matching items, in both a groupwise and pairwise manner, it allows for many different relationships between lists, such as maximising the difference between attributes or matching attribute values to a given distribution.

To do this, SOS uses both hard and soft constraints to determine which items should be selected. Hard constraints are constraints that need to be met. For example, a hard constraint can be used to ensure that in one list, there are no items with low value for frequency. Soft constrains are used to define relations between lists that ideally should be met. For example, soft constraints can be used to match two lists on a given attribute, match a selection's attribute value to a given distribution or to maximise the difference between two lists. For each soft constraint there exists a penalty function. SOS aims to reduce the total sum of the penalty functions.

The penalty function for matching two lists on a given attribute uses a Minkowski norm (p-norm) to determine how similar the selections' are. We let f_a^B be a vector containing all the attribute values for attribute a in list B and w_a be some weight applying to the attribute. SOS uses a measure, $\mathcal{M}$, either the average or the standard deviation to measure the similarity of the lists, before applying the Minkowski norm. The penalty function for groupwise matching first applies the measure to each chosen list before taking the squared difference, that is, $w_a|\mathcal{M}(f_a^B) - \mathcal{M}(f_a^{B'})|^p$, hence it is only comparing the measure of the two lists. Conversely pairwise aims to match pairs by taking the difference of the two sets' items' attributes, then applying some measure and taking the norm. The penalty function is given by $\frac{w_a}{k}(\sum_{i=1}^{k} |\mathcal{M}(f_{ai}^B - f_{ai}^{B'})|)^p$.

Match uses a depth first tree selection to find optimal solutions using pairwise matching. In order to find matching sets, Match finds tuples of matching items, one item taken from each lists. Match continues to find matching tuples until there are enough items selected. It then proceeds to backtrack in order to try and find better solutions. If the current tuple selection produces a solution worse than the current known best solution, Match will abandon the current selection and backtrack until it finds a better option.

The similarity of a tuple of items is measured by using the Euclidean distance over each of the items' attributes. For example a tuple might have two items, whose attribute vectors are $(1, 2, 3)$ and $(2, 2, 3)$. The similarity value of this tuple would be $(|(1, 2, 3) - (2, 2, 3)|)^2 = 1$. The total measure of all the items currently selected is given by the sum of all the tuples' similarity values.

Lastly, Boland et al. (2013) describe an approach to solving PIES using Mixed Integer Programming (MIP). This approach uses a variation on groupwise matching to define similarity. However, rather than use group characteristics such as average or standard deviation, this approach aims to match the histogram for each attribute in the lists, i.e. the same number of items in each list and attribute value. This MIP approach uses the ℓ^1 and ℓ^2 norms to compare solutions. In addition to these, there is an option to compare each attribute count to the overall average count for that attribute value, or compare pairs of attribute value counts. For example, consider the selection of words in Figure 1 being matched over the number of syllables in each word, S. For this selection, the average number of words with one and two syllables is 2 and 3 respectively. When comparing the histograms, there are two choices: compare each list's count to the *average* count or compare each *pair* of lists' counts. Using the ℓ^2 norm and averages to compare the number of words with one syllable gives $(3-2)^2 + (2-2)^2 + (1-2)^2 = 2$. In this case using ℓ^1 and averages would produce the same result. However, if we compare using pairs, ℓ^1 gives $|3-2| + |3-1| + |2-1| = 4$ and ℓ^2 gives $(3-2)^2 + (3-1)^2 + (2-1)^2 = 6$. The MIP approach allows the researcher to choose whether to match using pairs, averages and ℓ^1 and ℓ^2.

The attributes which PIES are to match over are called *soft attributes*. This approach also allows for *hard attributes*. For each hard attribute value, the researcher specifies how many items with that value should be selected from each list.

Each SOS, Match and the MIP approaches allows for similar sets to be created. SOS offers the greatest flexibility in defining the relationships between the sets, yet does not give a guaranteed optimal solution. Match has the capability of giving an optimal solution, but it needs to search an entire solution tree which takes an extremely long time. Additionally, it gives no indication of the optimality of the intermediate solutions. The

List 1		List 2		List 3	
Item	**S**	**Item**	**S**	**Item**	**S**
Fawn	1	Doubts	1	Lurks	1
Crumpled	2	Yellow	2	Tamping	2
Flour	1	Hath	1	Spiders	2
Spiders	2	Hanging	2	Rampant	2
Huffs	1	Muffling	2	Captain	2

Figure 1. Example words

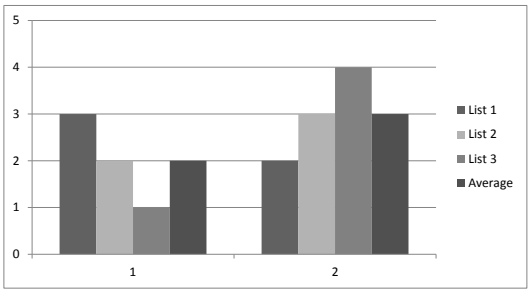

Figure 2. Histogram for Table 1, including the average attribute count

MIP approach produces superior results in a shorter amount of time when compared to SOS and Match. However, it does not allow for the flexibility that SOS does when defining relations between lists and can struggle with data sets that contain many hundreds of attributes. Also, due to the discrete nature of histograms, it considers a word with four syllables as equally similar to a word with five syllables as to a word with seven syllables. It is this last shortcoming we are aiming to resolve in this paper.

2 PROBLEM DEFINITION

We consider a problem in which we have a set of *items*, W, from which a subset needs to be selected. For example, this set could be 20,000 words to be used in an identification task or a group of volunteers in a medical study. The set of all items is partitioned into different *lists* of items. The set of all lists is given by $\mathcal{B}$, such that $\cup_{B\in\mathcal{B}}B = \mathcal{B}$ and $B \cap B' = \emptyset$ for all $B, B' \in \mathcal{B}, B \neq B'$.

Each item has a *value* for each of a set of attributes, A. The value for attribute a for item w is given by f_{wa}. For example, a word may have the number of syllables or the average age it is learnt as an attribute. If a is the number of syllables and w is banana we say that $f_{wa} = 3$. The set of all attribute values for an attribute a is given by $C_a = \{f_{wa} : w \in W\}$.

There are two types of attributes: a single *hard attributes*, h and a set of *soft attributes*, S. For the hard attribute, the researcher can specify how many items with a particular value are to be chosen from each list, as given in Equation 1k. For example, if the hard attribute is the number of letters in a word, the researcher may decide that each list should have five words with four letters, and three words with five letters. Thus, the hard attribute is used to define how many items are to be selected from each list. Soft attributes are the attributes that the selected lists should be matched over.

The problem is to select (or pick) items so that the items selected from each list are as similar as possible in terms of the items' soft attributes. We have defined similarity as how well each list's histograms match for each attribute (Boland et al., 2013). We let η_{Bac} be the count of selected items in list B with value c for attribute a and $\overline{\eta}_{ac}$ be the average count of items per list. The constraints that define a and $\overline{\eta}_{ac}$ are given in Equations 1i and 1j.

Similarity can be defined in a variety of ways: by comparing each count to the *average* count or by comparing each *pair* of counts. Both of these techniques can be used in conjunction with the ℓ^1 (*linear*) and ℓ^2 (*quad*) norms. The measures for average-linear (HAL), average-quad (HAQ), pair-linear (HPL) and pair-quad (HPQ) are given in Equations 1a, 1d, 1e and 1h. In each case, as we are considering each soft attribute and attribute value separately, we will sum the similarity measure over each of these values to give us the objective value.

Using these as a definition of similarity has its shortfalls, in particular with attributes whose values have order. Consider four lists of words being matched over the number of syllables, s. Let the vector $(\eta_{B_1s1}, \eta_{B_1s2}, \eta_{B_1s3})$ denote the number of items with each syllable count. Using histograms, the vectors $(5, 0, 0)$ and $(0, 5, 0)$ are considered just as dissimilar as $(5, 0, 0)$ and $(0, 0, 5)$. However, most would consider the former more similar as one syllable words are more like two syllable words than they are to three. Hence, our problem is to find a measure of similarity for sets when their attributes values have some inherent ordering.

$$\min \quad \sum_{B\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} d_{Bsc} \tag{1a}$$

$$\text{s.t.} \quad d_{Bsc} \le \eta_{Bsc} - \overline{\eta}_{sc} \qquad \forall B\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{1b}$$

$$d_{Bsc} \le \overline{\eta}_{sc} - \eta_{Bsc} \qquad \forall B\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{1c}$$

$$\min \quad \sum_{B\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} (\eta_{Bsc} - \overline{\eta}_{ac})^2 \tag{1d}$$

$$\min \quad \sum_{B,B'\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} d_{BB'sc} \tag{1e}$$

$$\text{s.t.} \quad d_{BB'sc} \le \eta_{Bsc} - \eta_{B'sc} \qquad \forall B,B'\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{1f}$$

$$d_{BB'sc} \le \eta_{B'sc} - \eta_{Bsc} \qquad \forall B,B'\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{1g}$$

$$\min \quad \sum_{B,B'\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} (\eta_{Bsc} - \eta_{B'ac})^2 \tag{1h}$$

$$\eta_{Bsc} = \sum_{\substack{w\in B\\ f_{ws}\le c}} x_w, \qquad \forall B\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{1i}$$

$$\overline{\eta}_{sc} = \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}} k_{Bsc}, \qquad \forall s\in S, \forall c\in C_s \tag{1j}$$

$$h_{Bc} = \sum_{\substack{w\in B\\ f_{wh}=c}} x_w \qquad \forall B\in\mathcal{B}, \forall c\in C_h \tag{1k}$$

3 Measures of Similarity

Creating a measure based on histograms that allows for continuous values is problematic. Simply grouping together attribute values and adding these groups as additional attributes will produce results biased towards some of the attribute values. For instance, for some vector of attribute value counts $(\eta_{Bs1}, \eta_{Bs2}, \eta_{Bs3}, \eta_{Bs4})$, as well as matching these counts, we would also match $(\eta_{Bs1} + \eta_{Bs2}, \eta_{Bs2} + \eta_{Bs3}, \eta_{Bs3} + \eta_{Bs4})$.

Using a grouping like this means that each attribute value will contribute to two pairs of values, with the exception of the extreme attribute values. This will create a bias towards matching the middle values, as they have more representation in the pairs. To counterbalance this bias, we could have an additional pair for each of the extreme values. Nevertheless, this will now create a bias towards matching the extreme values.

Instead, we propose the use of the cumulative histogram, e.g., instead of trying to match the count of each individual attribute value, match the cumulative count of attribute values. As with the previous histogram approach, we will use four different variations for cumulative histogram: pairs-quadratic (CPQ), pairs-linear (CPL), average-quadratic (CAQ) and average-linear (CAL).

Rather than aiming to match each attribute value, matching using a cumulative histogram aims to "even up" any discrepancy in the counts as quickly as possible. Specifically, consider the histograms in Figures 3 and 5. For this case the former lists would be considered more similar compared to the latter. Yet, when we consider the same values, but instead as a cumulative histograms, given in Figures 4 and 6 respectively, the later is considered more similar as the differences in count for values 1 and 3 are corrected for in the subsequent attribute values.

Unlike grouping similar values, this measure suffers no discernible bias towards particular attribute values. This issue with using cumulative histograms comes when an attribute's values are not evenly spaced. For instance, consider $C_a = \{1, 2, 3, 10\}$ for some attribute a. As it has been presented, using the cumulative histogram in this case would consider 3 and 10 to be just as similar as 2 and 3. Moreover, consider the frequency attribute for words. In the database provided in the SOS examples, the most frequent word is *the*

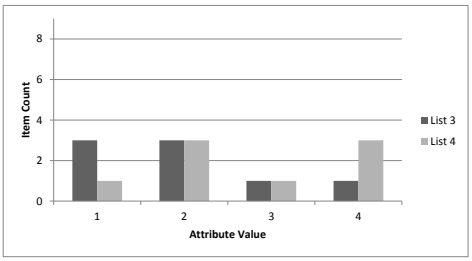

Figure 3. Histogram 1

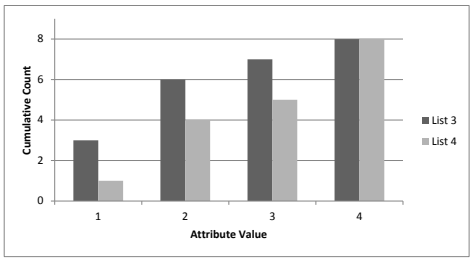

Figure 4. Cumulative Histogram 1

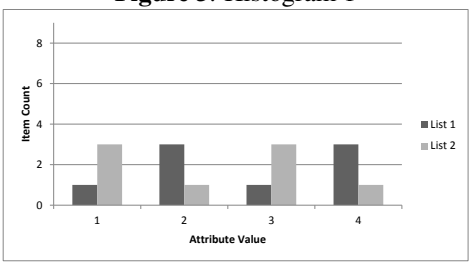

Figure 5. Histogram 2

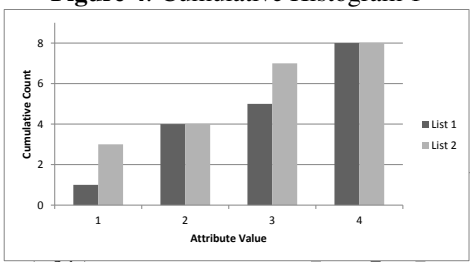

Figure 6. Cumulative Histogram 2

with a frequency value of $69,971$. The next most frequent is *and* at $28,852$ and the third most frequent is *that* with $10,595$. It isn't clear in this case if the difference in frequency between *the* and *and* is really twice that of the difference between *and* and *that*. Should these words be considered more different, in terms of frequency, than say *doctor*, with a frequency of 100 and *pemmican*[1] (frequency of 1)?. It is for these reasons we will leave it to the researcher to define C_a so as to best suit their experiment.

We let k_{Bsc} be the count items included in the cumulative count up to c for soft attribute s and $\overline{k}_{sc}$ be the average. These are defined by the constraints given in Equations 3a and 3b. The constraints defining cumulative frequency is used in every cumulative formulation, while the constraints defining $\overline{k}_{sc}$ is used in just the average formulations. Each formulation needs the constraint defined in Equation 3c to ensure the hard attribute conditions are met. The objective functions, and any additional constraints needed, for CAL, CAQ, CPL and CAQ are given in Equations 2a, 2d, 2e and 2h respectively.

$$\min \sum_{B\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} d_{Bsc} \tag{2a}$$

$$\text{s.t.} \quad d_{Bsc} \leq k_{Bsc} - \overline{k}_{sc} \qquad \forall B\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{2b}$$

$$d_{Bsc} \leq \overline{k}_{sc} - k_{,Bsc} \qquad \forall B\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{2c}$$

$$\min \sum_{B\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} (k_{Bsc} - \overline{k}_{ac})^2 \tag{2d}$$

$$\min \sum_{B,B'\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} d_{BB'sc} \tag{2e}$$

$$\text{s.t.} \quad d_{BB'sc} \leq k_{Bsc} - k_{B'sc} \qquad \forall B,B'\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{2f}$$

$$d_{BB'sc} \leq k_{B'sc} - k_{Bsc} \qquad \forall B,B'\in\mathcal{B}, \forall s\in S, \forall c\in C_s \tag{2g}$$

$$\min \sum_{B,B'\in\mathcal{B}}\sum_{s\in S}\sum_{c\in C_s} (\eta_{Bsc} - \eta_{B'ac})^2 \tag{2h}$$

[1] A nutritious food made from a mixture of fat and protein.

$$k_{Bsc} = \sum_{\substack{w \in B \\ f_{ws} \leq c}} x_w, \quad \forall B \in \mathcal{B}, \forall s \in S, \forall c \in C_s \tag{3a}$$

$$\overline{k}_{sc} = \frac{1}{|\mathcal{B}|} \sum_{B \in \mathcal{B}} k_{Bsc}, \qquad \forall s \in S, \forall c \in C_s \tag{3b}$$

$$h_{Bc} = \sum_{\substack{w \in B \\ f_{wh} = c}} x_w \qquad \forall B \in \mathcal{B}, \forall c \in C_h \tag{3c}$$

4 Numerical Results

4.1 Data

To compare the cumulative histogram to the previous histogram approach we are using two datasets of words. The first is the data base provided by SOS, a set of 22,461 words whose attributes include frequency[2], number of letters, number of phonemes[3], number of syllables, familiarity[4], concreteness[5], imagery[6] and the average age of acquisition[7]. Many of these attributes are normed to some value. Statistics about these attributes are summarised in Table 1. The second database we are using, AH, was provided to us by Andrew Heathcote. It

Attributes	Number of values	Average	Deviation	Min	Max
Frequency	505	65.15756	825.8636	1	69971
Letters	8	6.556431	1.926268	3	10
Phonemes	12	5.521393	1.910524	1	12
Syllables	5	2.117403	0.900049	1	5
Familiarity	433	169.9412	242.6823	0	657
Concreteness	443	131.7867	212.6101	0	670
Imagery	443	153.3857	224.9957	0	667
AoA	244	49.42353	134.8834	0	694

Table 1. Attribute Statistics for the SOS tests

contains $6,952$ Dutch words. The soft attributes are the monograms and bigrams of the words[8]. Most of the monograms have three or four attribute values: 0, 1, 2 and perhaps 3. Just two, *s* and *t* have four values, and three others just have two. Nearly all bigrams have two values: 0 and 1, with a few have three values. For each of these datasets, we have a single dummy hard attribute which is merely used to determine the number of items to be selected.

Our test data consisted of 47 different tests based on the SOS database and 4 based on the AH database. The variations on the SOS database include having the items randomly allocated to one of 2, 3, 4 or 10 lists, different attributes combinations such as just the phonemes, familiarity, concreteness, imagiability and age of acquisition or all the attributes. The number of items selected for each list can be any of 50, 100, 250, 1000, 1500, 2000 or 2500. Both the AH tests has 8 lists and either 250 or 125 words are to be chosen from each list. The attributes were either just the monograms or both the monograms and bigrams.

4.2 Results

In these tests we compare each of the cumulative formulations and compare them to the histogram formulations. Each of the formulations were implemented using Python 2.7 using `IBM IlOG CPlex 12.5` as the

[2]The frequency of a word in printed text
[3]The individual sound in a word, e.g. the phonemes of the word *the* are *th* and *e*
[4]The printed familiarity of a word
[5]How well defined the word is
[6]The ease that a word arouses a mental image
[7]The average age a word is learnt
[8]Monograms are the number of instances single characters in the words, bigrams are the number of pairs of characters. e.g. the monograms of *banana* are b, a, n with values 1, 3 and 2 respectively, the bigrams are ba, an and na, with values 1, 2 and 2

MIP solver. The tests were ran on a Dell PowerEdge R710 with dual hex core 3.06GHz Intel Xeon X5675 Processors and 96GB RAM running Red Hat Enterprise Linux 6. Each test was given a time limit of 2 hours.

First we compare how often the different formulations find an optimal solution and the time taken to do so. using the performance profile given in Figure 7, we can see how many instances were solved to optimality as those took less than 7200 seconds (2 hours) to solve. CPL and CAL were the best performing cumulative formulations finding optimal solutions to 37% and 35% of the tests respectively. This is compared to HPL and HAL which found solutions to 73% and 69% of the tests. In fact, HPQ, the worse performing histogram formulation still managed to find optimal solutions to 43% of the tests. In all cases, when a cumulative formulation found an optimal solution, the histogram formulations had also found an optimal solution is less time.

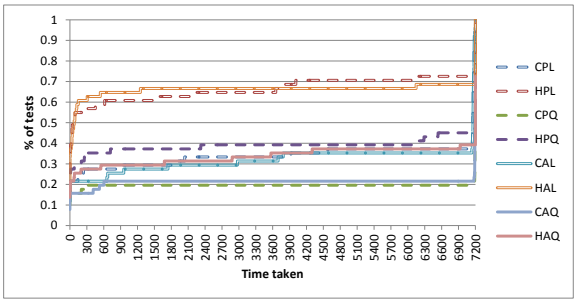

Figure 7. Time Performance Profile

In the SOS tests that were only matching over the phonemes, all formulations solved to optimality. The differences in time taken were negligible, as all tests solved in under one second. In SOS tests with four attributes, the cumulative formulations were mostly solved to optimality. However, it was significantly slower than the histogram formulations. The cumulative formulations solved very few of the other test instances, although many of the histogram linear formulations did.

As can be expected, the HPL formulation produces the solutions best matched for the HPL measure, shown in Figure 9. In fact, the HPL and HAL formulations produced most of the best solutions under every measure. In every test, when comparing over a cumulative measure, one of the histogram formulations always has a better result.

Thus, we conclude that while cumulative measures provide a superior way of comparing lists of items whose attributes have order, practically it is better to use histograms when finding matching sets. These test show that for the most part, using a histogram measure will produce a reasonable cumulative measure. The best approach would be to use a cumulative measure just for the attributes whose values have order, and use a histogram measure for all other attributes.

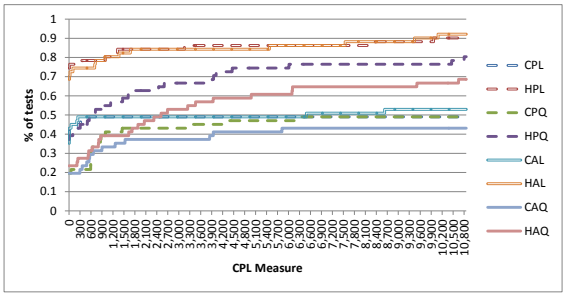

Figure 8. Performance Profile on CPL measure.

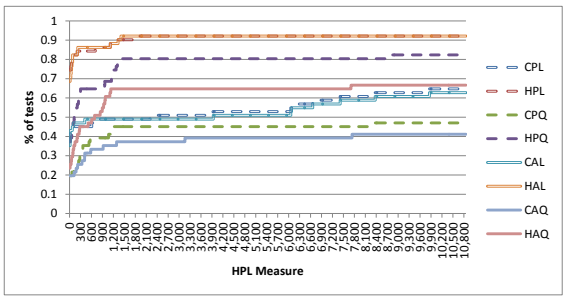

Figure 9. Performance Profile on HPL measure.

REFERENCES

Armstrong, B. C., C. E. Watson, and D. C. Plaut (2012, February). SOS! An algorithm and software for the stochastic optimization of stimuli. *Behavior research methods*.

Boland, N., R. Bunder, and A. Heathcote (2013). A Mixed Integer Programming approach to Picking Items for Experimental Sets. *To be published*.

Forster, K. I. (2000, October). The potential for experimenter bias effects in word recognition experiments. *Memory & cognition 28*(7), 1109–15.

Van Casteren, M., M. H. Davis, and B. S. Unit (2007, November). Match: a program to assist in matching the conditions of factorial experiments. *Behavior research methods 39*(4), 973–8.