Technical Appendices to
Extracting Summary Piles from Sorting Task Data

Simon J. Blanchard
McDonough School of Business, Georgetown University, Washington, DC 20057, USA
sjb247@georgetown.edu

Daniel Aloise
Department of Computer Engineering and Automation, Universidade Federal do Rio Grande do Norte, CEP 59072-970 Natal (RN) Brazil
aloise@gmail.com

Wayne S. DeSarbo
Department of Marketing, Pennsylvania State University, University Park, PA 16802, USA
desarbows@aol.com

ONLINE TECHNICAL APPENDIX 1 – MONTE CARLO SIMULATION FOR "PILE RECOVERY"

The Monte Carlo simulation we perform has two objectives. First, we wish to demonstrate the robustness of the proposed VNS algorithm by showing how it can solve problems with significant numbers of consumer piles within minutes. Second, we characterize the relationship between problem structure and computational complexity as to provide guidance on VNS parameters.

Monte Carlo Design. We prepared a collection of synthetic datasets in a partial factorial design by manipulating independent factors reflecting different data, parameters, and error conditions. We employ a fractional factorial design to study the main effects of each factor in Monte Carlo settings (see DeSarbo & Carroll, 1985; DeSarbo & Cron, 1988; Jedidi & DeSarbo, 1991). The fractional factorial design is shown in Table A1.

[INSERT TABLE A1 ABOUT HERE]

We varied the total number of consumers (I = 100, 500), the number of sorted items (J = 25, 50), the number of *summary piles* (K = 10, 20), whether items can appear in multiple piles (yes/no). For each consumer in the dataset, we sampled a Poisson distribution with rate λ ($3\ or\ 6$) to determine the number of piles c_i. Each of the c_i was obtained by sampling c_i piles (with replacement) from the *K summary piles*. Error was adding by randomly permuting the value of each y_{il_ij} with a predetermined probability (5% or 20%). Finally, we manipulated and ensured that the *summary piles* and consumer piles would only reflect multiple items per piles if allowed by the design. Although this did not influence dataset generation, we also included factors relating to the VNS parameters: we varied t_{max} (10 or 20) and total allowed

computational time as a termination (60 or 300 seconds). We performed three executions of the VNS for each design. The number of *summary piles* (*K*) was assumed to be known, consistent with research for clustering procedures that has performed similar Monte Carlo simulations (e.g., Blanchard, Aloise, and DeSarbo 2012; Brusco, Cradit, & Tashchian, 2003; Helsen & Green, 1991).

Performance of the VNS Algorithm for Category Covering. The first objective of the Monte Carlo simulation is to establish whether the VNS algorithm proposed provide excellent prediction of consumer piles. A summary of the results, along with expected error rate, is presented in Table A2. We find that the VNS algorithm obtains solutions that provide a percentage of mis-predictions very close to that of the randomly generated error, for each trial, with an average of 12.74% mis-prediction when the average expected would be around 12.50%. For example on dataset 1, the percentage of mis-prediction for VNS was 5.04%, close to the probability that each y_{il_ij} data is permuted (from 0 to 1 or from 1 to 0) of 5%.

[INSERT TABLE A2 ABOUT HERE]

Robustness Analyses. The second objective concerned the robustness of the VNS algorithm to various sorting task structures, dataset sizes, and VNS parameters. To investigate these issues, we performed a linear regression with the percentage mis-prediction as dependent variable and with the design factors as binary coded independent variables. The coefficients and significance are presented in Table A3.

[INSERT TABLE A3 ABOUT HERE]

Unsurprisingly, the number of mis-predictions increases as the error added to the y_{il_ij} increased. Yet, the VNS fit is fairly robust to this increase. The number of items, the number of *summary piles*, the structure of the sorting task (allowing multiple cards or not), and the two VNS parameters seem have no significant effect on the performance of the VNS. The first significant factor was the number of consumers, such that the addition of an additional 100 consumers increased the percentage of mis-prediction by approximately .014% ($\beta = .0055; t(39) = 4.62, p < .01$). The second significant factor is with regarding to the average (and variance in the) number of piles consumers made. We find that the model performs better when consumers make a larger number of piles ($\beta = -.0029; t(39) = -2.38, p = .02$). Finally, we note that the model also seems to perform well without needing many random starts to the procedure, with an average percentage mis-prediction standard deviation between the three executions of just 0.15%. Three executions thus seems appropriate.

TABLE A1 – Monte Carlo Simulation Design

Trial	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	1	1	1
3	0	1	0	0	1	0	1	1
4	1	1	0	0	1	1	0	0
5	0	0	1	0	1	1	1	0
6	1	0	1	0	1	0	0	1
7	0	1	1	0	0	1	0	1
8	1	1	1	0	0	0	1	0
9	0	0	0	1	1	1	0	1
10	1	0	0	1	1	0	1	0
11	0	1	0	1	0	1	1	0
12	1	1	0	1	0	0	0	1
13	0	0	1	1	0	0	1	1
14	1	0	1	1	0	1	0	0
15	0	1	1	1	1	0	0	0
16	1	1	1	1	1	1	1	1

Note: Factor 1: consumers 1=500, 0=100. Factor 2: Number of items 1=50, 0=25. Factor 3: Number of *summary piles*: 1=20, 0=10. Factor 4: Multiple cards per item allowed 1=yes, 0=no. Factor 5: Rate parameter for number of piles per individual (poison distribution) 1=6, 0=3. Factor 6: Error rate on y_{il_ij} 1=20%, 0=5%. Factor 7: VNS parameter: t-max 1=40, 0=20. Factor 8: VNS parameter: cpu time (sec) 1=540sec, 0=60sec. # piles indicates the total number of piles generated by the random generation process.

TABLE A2 – Monte Carlo Simulation Results – Mis-predictions (Numbers and Percentage)

			VNS	VNS
Data	**Number of Piles in Dataset**	**Expected Error**	**Number of Mis-predictions**	**Percentage error**
1 - 1	361	5%	455	5.04%
1 - 2	361	5%	455	5.04%
1 - 3	361	5%	455	5.04%
2 - 1	1807	20%	9784	21.66%
2 - 2	1724	20%	9490	22.02%
2 - 3	1724	20%	9277	21.52%
3 - 1	638	5%	1643	5.15%
3 - 2	638	5%	1643	5.15%
3 - 3	638	5%	1643	5.15%
4 - 1	3052	20%	30653	20.09%
4 - 2	3052	20%	31624	20.72%
4 - 3	3052	20%	31921	20.92%
5 - 1	658	20%	3227	19.62%
5 - 2	658	20%	3276	19.91%
5 - 3	658	20%	3227	19.62%
6 - 1	3112	5%	3835	4.93%
6 - 2	3112	5%	3835	4.93%
6 - 3	3112	5%	3835	4.93%
7 - 1	355	20%	3603	20.30%
7 - 2	355	20%	3603	20.30%
7 - 3	355	20%	3575	20.14%
8 - 1	1831	5%	5168	5.65%
8 - 2	1831	5%	4587	5.01%
8 - 3	1831	5%	4587	5.01%
9 - 1	580	20%	2845	19.62%
9 - 2	580	20%	2979	20.54%
9 - 3	580	20%	2948	20.33%
10 - 1	3104	5%	3977	5.13%
10 - 2	3104	5%	3977	5.13%
10 - 3	3104	5%	3977	5.13%
11 - 1	362	20%	3595	19.86%
11 - 2	362	20%	3595	19.86%
11 - 3	362	20%	3595	19.86%
12 - 1	1831	5%	4616	5.04%
12 - 2	1831	5%	4616	5.04%
12 - 3	1831	5%	4616	5.04%
13 - 1	371	5%	434	4.68%
13 - 2	371	5%	434	4.68%
13 - 3	371	5%	434	4.68%
14 - 1	1779	20%	9212	20.71%
14 - 2	1779	20%	9506	21.37%
14 - 3	1779	20%	9651	21.70%
15 - 1	597	5%	1454	4.87%
15 - 2	597	5%	1454	4.87%
15 - 3	597	5%	1454	4.87%
16 - 1	3048	20%	30932	20.30%
16 - 2	3048	20%	31059	20.38%
16 - 3	3048	20%	30709	20.15%
Mean	1464	12.5%	7239	12.74%

TABLE A3 - Monte Carlo Simulation Design Factors as Predicting VNS Percentage Mis-prediction Rate

	B	t	$Sig.$
(Constant)	.0512	28.56	.00
Factor 1: Number of Consumers	.0055	4.62	.00
Factor 2: Number of Items	-.0017	-1.45	.16
Factor 3: Number of *Summary Piles*	-.0019	-1.58	.12
Factor 4: Multiple Cards per Item Allowed	-.0017	-1.39	.17
Factor 5: c_i: Rate Parameter (Poisson Distributed)	-.0029	-2.38	.02
Factor 6: Error Rate	.1547	129.36	.00
Factor 7: VNS Parameter: t-max	-.0005	-.39	.70
Factor 8: VNS Parameter: CPU time	.0007	.56	.58
Adjusted R^2	.9900		

ONLINE TECHNICAL APPENDIX 2 – MODEL COMPARISONS

In the present appendix, we wished to compare how a clustering algorithm (e.g., hierarchical clustering with Ward's (1963) method), Latent Dirichlet Allocation (LDA), and our proposed methodology performed on datasets of various types of structures and amounts of latent heterogeneity.

Data Generating Process

We sought a data generating process that would not necessarily provide an advantage to one methodology over the others and that would isolate the role of heterogeneity. To do so, we first varied the number of consumers (*I*=100, 500), items (*J*=25, 50), the number of piles each consumer makes ($NP = 3, 8$). We varied heterogeneity by assigning each consumer to one of *N* group solutions ($NS = 1, 3, 5$), where for each solution the objects were randomly assigned to one of the *NP* piles. As an example, a solution where *I*=100, *J*=25, *NP*=3, *NS*=1 would involve a set of 100 consumers who sort 25 items into the same exact 3 solutions (i.e., partitions). In contrast, a solution where *I*=100, *J*=25, *NP*=8, *NS*=3 would involve 3 different ways of assigning the 25 items into 8 different piles. We note that the items in this simulation were not allowed to be assigned to multiple piles.

In addition, we also manipulated the amount of error added to the data. To ensure that noise what added in a way that did not result in items being assigned simultaneously into more than one pile (per consumer), we generated error by performing a random number of "swap moves" where two items' assignments to piles are exchanged. Specifically for each consumer, we first determined a number of moves following one of three error levels: 0 (the consumer's solution corresponds exactly to its group's solution), Poisson(2), and Poisson(4). Then, for this

consumer for the determined number of moves, two items are exchanged from one pile to the other.

This approach to generating experimental data has several advantages. First, none of the methods have any obvious advantage over the other. Second, all three methods would be expected to perform equally well under the conditions of homogeneous sorts (only one set of piles use to generate the data) and no error added. In fact, when $e = 0$ and $NS = 1$, we would expect the methods to be able to recover the data perfectly. Third, it allows us to illustrate how our proposed method can recover heterogeneous sorting data. For instance if $NS = 3$ and $NP = 3$ without error ($e = 0$), the data is generated such that there are exactly 9 different piles in the data. We expect that when our proposed model is executed at $NS \times NP$ (e.g., 9), and there is no error present, the model can *still* recover all the data perfectly. We do not expect this for clustering, which must produce only one set of homogeneous piles to summarize the data. Using the 5 factors described above, we generated an experimental design with 72 synthetic datasets ($2^3 \times 3^2 = 72$). The resulting design is presented in Table B1.

[INSERT TABLE B1 ABOUT HERE]

Competing Algorithms

For our investigation, we compare three procedures: our proposed method, LDA (Steyvers and Griffiths 2007), and Ward's (1963) clustering. For our model and LDA, we ran the models at level $K = NP \times NS$. For our model, we only allowed a single run, and only for 60 seconds. For LDA, we also only allowed a single run at each level of *K* and allowed their model to determine termination. For clustering, we used Ward's criterion after the data was first converted into a *J* x *J* pairwise count matrix (C) and second converted into distances by setting D=1./(1+C). We then obtained two different clustering solutions with: *K=NP* (i.e., average

number of piles a consumer has in the data), and *K=NP* x *NS* (i.e., true number of unique piles). This ensures that any discrepancy in fit could not be explained by additional complexity given to the method.

Results

First, we note that all approaches did very well when no error (e=0) was added to the data and when no heterogeneity was present in the way consumers sorted (*NS*=1) in trials 1-8. Clustering models recovered the data perfectly 6/8 times, LDA 4/8 times, and our proposed model all 8 times. Larger differences emerged when heterogeneity was added to the data. When no error was present and there was heterogeneity (trials 9-24), the average error rate for clustering *K=NP* was 18.06%, with clustering *K = NS* x *NP* at 13.81%, LDA at 10.25%, and our proposed model at perfectly recovering the data in all but one instance (trial 24). We note for the Ward's clustering model, allowing for additional clusters to accommodate for heterogeneity did not provide sufficient flexibility to perfectly recover the data.

Second, there was a clear ordering in terms of fit between the models. Whereas clustering with more clusters (*K=NS* x *NP*; 18.07%) performed better than with fewer (*K=NP*; 14.93%, $t(71)=9.91, p<.01$), LDA did significantly better (9.75%, $t(71)=7.94$, $p<.01$). Yet, our proposed model outperformed these others including LDA (3.71%, $t(71)=11.45$, $p<.01$).

Third, although our proposed model performed at least as well or better as LDA on all trials, we can regress the difference in LDA vs our model's performance (LDA's error minus ours) to investigate areas of sensitivity for both. Doing so revealed that there is no difference between the models with respect to their ability to recover data in the presence of a large number of consumers ($\beta = .0027, t(71) = -1.60, p = .11$), number of items ($\beta = .000, t(71) =$

$1.355, p = .18$), or number of average piles per participant ($\beta = .002, t(71) = .579, p = .57$). The results do suggest that our model performs comparatively better when there is less error in the data ($\beta = -.007, t(71) = -3.52, p = .01$); although, we do note that at all levels of error tested, our model performs better than LDA. Finally, we found a significant interaction between the number of true solutions (*NS*) and the average number of piles (*NP*; $\beta = -.004, t(71) = -3.72, p < .01$). Exploring this interactions, we found that there is no difference in performance due to solutions with more piles (*NP*=8 vs *NP*=4) between LDA and our proposed model when the solutions are homogeneous ($NS = 1; \beta = -.0018, t(71) = -.675, p = .50$); our proposed procedure performed better when the data has been generated using heterogeneous solutions with more piles ($NS = 3; \beta = -.0095, t(71) = -5.62, p = .01$), and even more so when heterogeneity increases ($NS = 5; \beta = -.0172, t(71) = -6.44, p = .01$).

Fourth, we note that we were able to use these 72 datasets to investigate whether a scree-plot could be successful used to determine the level of *K* for the proposed methodology. For each of the 72 trials, we also estimate the model from *K*=1,…, *NS* x *NP* (true number of unique piles) + 10) and gathered the error for each level of *K*. We then determined *K** as the point at which the last percentage substantial drop in error improvement that occurred by increasing *K** by one, and verified if this matched *NS*x*NP*. We present two samples of the tables used and whether *K* can be recovered in Table B2. Solutions where the number of unique piles was recovered by *K** is marked in the table by an asterisk. In sum, we find that using a scree plot perfectly recovered *K* in 79.82% of the trials (59/72).

Discussion

In the present simulation, we generated 72 datasets following an experimental design that varied the size of the data (number of consumers, number of items, number of piles made per consumer), and the amounts of heterogeneity and error present in the data. We found that our proposed model is able to recover complex heterogeneous structures with substantial amount of errors, even if the data do not allow consumers to assign items to one and only one pile. Overall, in this particular simulation, we note superior performance of our proposed methodology over LDA and Ward's clustering.

TABLE B1 – RESULTS (ERROR RATES) PER MODEL

#	*I*	*J*	*NP*	*NS*	Error	Clustering *K=NP*	Clustering *K=NPxNS*	LDA *K=NPxNS*	Proposed Model *K=NPxNS*	
1	100	25	4	1	0	0.00	0.00	0.00	0.00	*
2	500	25	4	1	0	0.00	0.00	0.08	0.00	*
3	100	50	4	1	0	0.00	0.00	0.00	0.00	*
4	500	50	4	1	0	0.00	0.00	0.00	0.00	*
5	100	25	8	1	0	0.04	0.04	0.05	0.00	*
6	500	25	8	1	0	0.03	0.03	0.03	0.00	*
7	100	50	8	1	0	0.00	0.00	0.00	0.00	*
8	500	50	8	1	0	0.00	0.00	0.02	0.00	*
9	100	25	4	3	0	0.18	0.16	0.14	0.00	*
10	500	25	4	3	0	0.22	0.17	0.10	0.00	*
11	100	50	4	3	0	0.24	0.18	0.17	0.00	*
12	500	50	4	3	0	0.24	0.18	0.14	0.00	*
13	100	25	8	3	0	0.12	0.09	0.06	0.00	*
14	500	25	8	3	0	0.11	0.09	0.05	0.00	*
15	100	50	8	3	0	0.12	0.09	0.08	0.00	*
16	500	50	8	3	0	0.12	0.09	0.06	0.00	*
17	100	25	4	5	0	0.24	0.20	0.12	0.00	*
18	500	25	4	5	0	0.24	0.20	0.12	0.00	*
19	100	50	4	5	0	0.25	0.19	0.18	0.00	*
20	500	50	4	5	0	0.28	0.20	0.15	0.00	*
21	100	25	8	5	0	0.13	0.09	0.06	0.00	*
22	500	25	8	5	0	0.13	0.09	0.06	0.01	*
23	100	50	8	5	0	0.14	0.10	0.08	0.00	*
24	500	50	8	5	0	0.13	0.09	0.07	0.00	*
25	100	25	4	1	2	0.26	0.26	0.08	0.07	*
26	500	25	4	1	2	0.30	0.30	0.13	0.07	*
27	100	50	4	1	2	0.22	0.22	0.03	0.03	*
28	500	50	4	1	2	0.27	0.27	0.04	0.04	*
29	100	25	8	1	2	0.13	0.13	0.06	0.03	*
30	500	25	8	1	2	0.15	0.15	0.06	0.03	*
31	100	50	8	1	2	0.07	0.07	0.04	0.02	*
32	500	50	8	1	2	0.14	0.14	0.04	0.02	*
33	100	25	4	3	2	0.27	0.23	0.17	0.07	*
34	500	25	4	3	2	0.29	0.23	0.15	0.07	*
35	100	50	4	3	2	0.25	0.23	0.17	0.04	*
36	500	50	4	3	2	0.32	0.25	0.14	0.04	*
37	100	25	8	3	2	0.14	0.09	0.08	0.03	*
38	500	25	8	3	2	0.15	0.09	0.07	0.03	
39	100	50	8	3	2	0.13	0.10	0.08	0.02	*
40	500	50	8	3	2	0.14	0.12	0.07	0.02	*
41	100	25	4	5	2	0.25	0.21	0.16	0.06	*
42	500	25	4	5	2	0.30	0.21	0.14	0.07	*
43	100	50	4	5	2	0.26	0.23	0.19	0.03	*
44	500	50	4	5	2	0.30	0.24	0.16	0.04	
45	100	25	8	5	2	0.14	0.09	0.07	0.03	
46	500	25	8	5	2	0.15	0.09	0.08	0.03	
47	100	50	8	5	2	0.13	0.10	0.08	0.02	*
48	500	50	8	5	2	0.15	0.11	0.07	0.02	
49	100	25	4	1	4	0.25	0.25	0.20	0.12	*
50	500	25	4	1	4	0.31	0.31	0.16	0.13	*
51	100	50	4	1	4	0.26	0.26	0.10	0.08	*
52	500	50	4	1	4	0.30	0.30	0.07	0.07	*
53	100	25	8	1	4	0.14	0.14	0.11	0.06	*
54	500	25	8	1	4	0.14	0.14	0.09	0.06	*
55	100	50	8	1	4	0.13	0.13	0.05	0.04	*
56	500	50	8	1	4	0.15	0.15	0.05	0.04	*
57	100	25	4	3	4	0.27	0.22	0.18	0.11	*
58	500	25	4	3	4	0.28	0.22	0.16	0.12	
59	100	50	4	3	4	0.26	0.24	0.18	0.07	*
60	500	50	4	3	4	0.32	0.24	0.17	0.09	
61	100	25	8	3	4	0.14	0.09	0.08	0.05	
62	500	25	8	3	4	0.14	0.08	0.08	0.05	
63	100	50	8	3	4	0.15	0.12	0.08	0.03	*
64	500	50	8	3	4	0.15	0.12	0.07	0.04	
65	100	25	4	5	4	0.30	0.21	0.17	0.12	
66	500	25	4	5	4	0.28	0.21	0.16	0.13	
67	100	50	4	5	4	0.29	0.24	0.19	0.07	*
68	500	50	4	5	4	0.31	0.24	0.17	0.07	*
69	100	25	8	5	4	0.14	0.09	0.08	0.05	
70	500	25	8	5	4	0.14	0.09	0.07	0.05	
71	100	50	8	5	4	0.14	0.11	0.09	0.04	*
72	500	50	8	5	4	0.15	0.11	0.08	0.04	

TABLE B2 – USING "ELBOW IN THE CURVE" TO DETERMINE K

K	Mis-predictions	Improvement (#)	Improvement (%)
1	14995	-	-
2	13025	1970	13%
3	10584	2441	19%
4	8978	1606	15%
5	8505	473	5%
6	7551	954	11%
7	6205	1346	18%
8	5721	484	8%
9	5132	589	10%
10	4199	933	18%
11	3648	551	13%
12	**3301**	**347**	**10%**
13	3260	41	1%
14	3229	31	1%
15	3206	23	1%

Panel A: Trial 34, True solution of K=12. K can be inferred a significant drop in % Improvement.

K	Mis-predictions	Improvement (#)	Improvement (%)
1	2397	-	-
2	2257	140	6%
3	2204	53	2%
4	2097	107	5%
5	1995	102	5%
6	1887	108	5%
7	1793	94	5%
8	1714	79	4%
9	1650	64	4%
10	1639	11	1%
11	1507	132	8%
12	1473	34	2%
13	1459	14	1%
14	1356	103	7%
15	1317	39	3%

Panel B: Trial 62, True Solution of *K*=12. *K* cannot easily be inferred from % Impbrovement.