Clustering for set partitioning with a case study in ridesharing

Cathy Wu, Ece Kamar, Eric Horvitz

***Abstract*— By exploring alternative approaches to combinatorial optimization, we propose the first known formal connection between clustering and set partitioning, with the goal of identifying a subclass of set partitioning problems that can be solved efficiently and with optimality guarantees through a clustering approach. We prove the equivalence between classical centroid clustering problems and a special case of set partitioning called metric k-set partitioning. We discuss the implications for k-means and regularized geometric k-medians, and we give several future extensions and applications. Finally, we present a case study in combinatorial optimization for ridesharing, in which we use an efficient Expectation Maximization (EM) style algorithm to achieve a 69% reduction in total vehicle distance, as compared with no ridesharing.**

I. Introduction and combinatorial optimization problems

Set partitioning is the optimization form of exact cover, one of Karp's 21 NP-complete problems, and it arises commonly in planning applications where it is used to identify the best partition of a set of objects. Examples include scheduling airline flight crews [1], [2], sharing rides [3], and kidney swapping programs [4] [5]. Optimizing for the best set partitioning solution is NP-hard, and classical combinatorial optimization methods are cumbersome (e.g. integer programming [6], branch-and-cut [1]), restricted (e.g. triangle-packing approximation [7], [8], m-sets [9], [10]), or do not provide guarantees (e.g. simulated annealing [11], genetic algorithms [12], metaheuristics [13]). At the same time, clustering methods also solve a partitioning problem, and although the underlying optimization problem is again NP-hard, methods in common use provide efficient iterative procedures that offer probabilistic guarantees and that are amenable to parallelization. On the other hand, while classical combinatorial optimization approaches are suitable for extremely general and expressive problems, clustering methods are typically restricted to specific models and objectives.

We are interested in the following questions:

- What optimization problems can be expressed at the intersection of the two approaches?
- How can these problems benefit from the formalism of one and the methods of the other?

We formalize the connection between set partitioning and clustering, with the goal of identifying a subclass of set partitioning problems that can be solved efficiently and with optimality guarantees through a clustering approach. We offer a demonstrative example of the connection: we demonstrate the correspondence between a restricted setting of set partitioning and classical centroid-based unsupervised clustering methods, for instance the k-means algorithm [14].

UC Berkeley, EECS, cathywu@eecs.berkeley.edu; this work was done during an internship at Microsoft Research
Microsoft Research
Microsoft Research

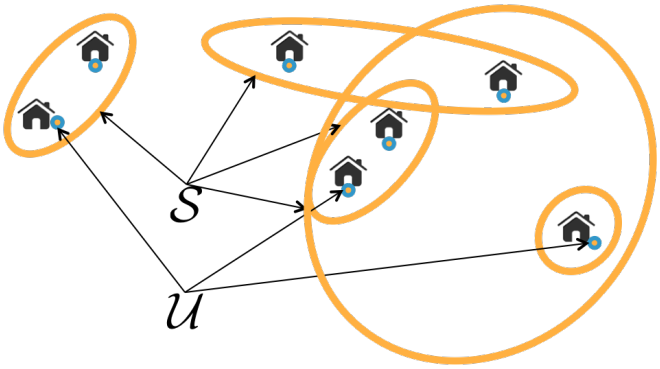

Fig. 1. Set partitioning problem.

A. Set partitioning

First, consider the problem of *set partitioning* (see Figure 1), where the goal is to find the best disjoint cover within some collection $\mathcal{S} \subseteq 2^{\mathcal{U}}$, where $\mathcal{U}$ is the universe of elements. This is a restricted setting of set packing, where feasible solutions must cover the whole universe, rather than a subset. Best is defined as minimizing the overall cost in terms of weights assigned to each subset. We denote the cost of each subset $S \in \mathcal{S}$ by c_S. We denote the solution vector $x = (x_S)_{S \in \mathcal{S}}$, where $x_S = 1$ indicates that S is in our set partitioning solution. Then, the problem is defined as follows:

$$\text{opt} = \min_x \sum_{S \in \mathcal{S}} c_S x_S \tag{1}$$

subject to

$$\sum_{S: u \in S} x_S = 1, \quad u \in \mathcal{U} \tag{2}$$

$$x_S \in \{0, 1\} \quad S \in \mathcal{S} \tag{3}$$

B. Metric k-set partitioning and centroid-based clustering

Now, we specialize to the special case of metric spaces and k-covers (covers of exactly size k). We let $\mathcal{S} = 2^{\mathcal{U}}$, and we endow universe $\mathcal{U}$ with a metric space $(\mathcal{X}, d)$ such that $\mathcal{U} \subseteq \mathcal{X}$ and define the cost of a subset to be minimal relative to a centroid point, that is $c_S := \min_{x \in \mathcal{X}} \sum_{s \in S} d(s, x)$. Additionally including a k-set constraint, we then have the following optimization problem, which we call the *metric*

k-set partitioning problem:

$$\min_{x} \sum_{S\in\mathcal{S}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x')x_S = \min_{x} \sum_{S\in\mathcal{S}} \sum_{s\in S} d(s,\mu_S)x_S \tag{4}$$

subject to

$$\sum_{S:u\in S} x_S = 1, \quad u \in \mathcal{U} \tag{5}$$

$$\sum_{S\in\mathcal{S}} x_S = k \tag{6}$$

$$x_S \in \{0,1\} \quad S \in \mathcal{S} \tag{7}$$

where we define the subset centroids $\mu_S := \arg\min_{x\in\mathcal{X}} \sum_{s\in S} d(s,x)$. Note that under a metric, without the size k constraint, the optimal solution would be all the singleton sets. This is true for clustering as well.

Constraint (5) implies that each element is measured with respect to exactly one centroid (and in fact, the closest one, due to the objective) and Constraint (6) encodes that there are exactly k centroids (and in fact, they minimize the distance with respect to the elements assigned to it, due to the objective). With these observations, the previous optimization problem then collapses neatly into the following well-studied optimization problem for *centroid-based clustering*:

$$\min_{(T_1,T_2,\ldots,T_k)} \sum_{j=1}^{k} \sum_{x\in T_j} d(x,\mu_j) \tag{8}$$

subject to

$$\cup_{j\in[k]} T_j = \mathcal{U} \tag{9}$$

$$T_i \cap T_j = \emptyset, \quad \forall i \neq j \tag{10}$$

$$\mu_j = \arg\min_{x\in\mathcal{X}} \sum_{s\in T_j} d(s,x) \tag{11}$$

When the metric $d(\cdot,\cdot)$ is restricted to the sum of squares loss, also knowns as k-means clustering, the resulting optimization problem may be solved efficiently and with probabilistic guarantees with `k-means++` initialization [15] and the k-means algorithm. Thus, the *metric k-set partitioning problem*, an instance of set partitioning, benefits tremendously from a clustering approach. We now prove the result.

II. A formal connection between clustering and set partitioning

Proposition 1 (Equivalence): Metric k-set partitioning and centroid-based clustering are equivalent. That is, Problem (4)-(7) and Problem (8)-(11) are equivalent.

Before proving the proposition, we first make the following definition and claim:

Definition 1 (k-partition): If $\mathcal{P}$ is a k-partition constraint of $\mathcal{U}$, then $\mathcal{P} = (P_1, P_2, \ldots, P_k) \subseteq \mathcal{S}$ such that:

$$P_i \subseteq \mathcal{U}, \quad \forall i$$
$$\bigcup_i P_i = \mathcal{U}$$
$$P_i \cap P_j = \emptyset, \quad \forall i \neq j$$

Claim 1 (k-partition constraint): The constraints of Problem (4)-(7) is a k-partition constraint.

Proof: [of claim] $(\rightarrow)$: We first show that we may construct such a partition $\mathcal{P}$ from a solution $x = (x_S)_{S\in\mathcal{S}}$ satisfying the constraints of Problem (4)-(7). Define $\mathcal{P} := \{S : x_S = 1, S \in \mathcal{S}\}$. By construction, $\forall P \in \mathcal{P}, P \in \mathcal{S} \implies P \subseteq \mathcal{U}$. Constraint (6) ($\sum_{S\in\mathcal{S}} x_S = k$) implies that $|\mathcal{P}| = k$. Constraint (5) ($\sum_{S:u\in S} x_S = 1, \quad u \in \mathcal{U}$) implies both coverage and mutually disjoint conditions. Thus, x may be represented as a k-partition $\mathcal{P}$.

$(\leftarrow)$: Now we show that a partition $\mathcal{P}$ may be used to construct a solution $x = (x_S)_{S\in\mathcal{S}}$ satisfying the constraints of Problem (4)-(7). We define $x \in \{0,1\}^{|\mathcal{S}|}$ such that

$$x_S := \begin{cases} 1 & \text{if } S \in \mathcal{P} \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\bigcup_i P_i = \mathcal{U} \implies \sum_{S:u\in S} x_S \geq 1, \quad u \in \mathcal{U}$$
$$P_i \cap P_j = \emptyset, \quad \forall i \neq j \implies \sum_{S:u\in S} x_S \leq 1, \quad u \in \mathcal{U}$$

Together, this implies Constraint (5) ($\sum_{S:u\in S} x_S = 1, u \in \mathcal{U}$). Constraint (6) is trivially satisfied by construction. Thus, $\mathcal{P}$ may be represented as a length-$|\mathcal{S}|$ binary vector satisfying the constraints of Problem (4)-(7). Finally, we conclude that the constraints of Problem (4)-(7) are equivalent to that of a k-partition. ∎

Proof: [of proposition] We perform a change in variable in Problem (4)-(7) from $x = (x_S)_{S\in\mathcal{S}}$ to $\mathcal{P}$ (established in Claim 1), taking $\mathcal{P} := \{S : x_S = 1, S \in \mathcal{S}\}$.

$$\begin{aligned} \min_{x} \sum_{S\in\mathcal{S}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x')x_S &= \min_{\mathcal{P}} \sum_{S\in\mathcal{P}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x')1 \\ &\quad + \sum_{S\in\mathcal{S}\setminus\mathcal{P}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x')0 \\ &= \min_{\mathcal{P}} \sum_{S\in\mathcal{P}} \min_{x'\in\mathcal{X}} \sum_{s\in S} d(s,x') \end{aligned}$$

subject to the k-partition constraint. We observe that our sum is reduced from $|\mathcal{S}|$ terms to k terms. To make the size of $\mathcal{P}$ more explicit, we may write equivalently,

$$\min_{\mathcal{P}=(P_1,P_2,\ldots,P_k)} \sum_{i=1}^{k} \min_{x'\in\mathcal{X}} \sum_{s\in P_i} d(s,x')$$

Finally, we define $\mu_i := \arg\min_{x'\in\mathcal{X}} \sum_{s\in P_i} d(s,x')$, and we arrive at a common form of the centroid-based clustering problem (as in Problem (8)-(11)):

$$\min_{\mathcal{P}=(P_1,P_2,\ldots,P_k)} \sum_{i=1}^{k} \sum_{s\in P_i} d(s,\mu_i)$$

subject to

$$P_i \subseteq \mathcal{U}, \quad \forall i$$
$$\bigcup_i P_i = \mathcal{U}$$
$$P_i \cap P_j = \emptyset, \quad \forall i \neq j$$

Since all operations are reversible (equivalences), we have established equivalence of the two problems. ■

Some examples of centroid-based clustering problems include the k-means problem, the k-median problem, and the geometric k-median problem. Under equivalence given by Proposition 1, the algorithms for one problem may apply also to the other. We now present several special cases and extensions, and then we present an application that makes use of these settings.

A. Special case: k-means

A special case of the equivalence is the k-means objective [16], where $d(s,x) = \min_{x\in\mathcal{X}} \sum_{s\in S} \|s-x\|_2^2$.

Corollary 1 (k-means++ for set partitioning): When restricted to sum of squares loss, `k-means++` attains a $\mathrm{O}(\log k)$-approximate solution in expectation to the metric k-set partitioning problem `k-means++`.

By examining the operations of the k-means algorithm, we may gain intuition on how the steps translate back into context of the set partitioning problem.

Atomic operations of Lloyd's algorithm: Lloyd's algorithm for k-means clustering is an iterative method with two main steps [17]:

1) Assignment step:

$$P_i^{(t)} = \left\{ s : \left\| s - \mu_i^{(t)} \right\| \leq \left\| s - \mu_j^{(t)} \right\|, \forall j \in [k] \right\} \tag{12}$$

   where each $s \in \mathcal{U}$ is assigned to exactly one $P_i^{(t)}$.

2) Update step:

$$\mu_i^{(t+1)} = \frac{1}{\left| P_i^{(t)} \right|} \sum_{s_j \in P_i^{(t)}} s_j \tag{13}$$

In the context of set partitioning, the two steps have the following interpretation:

1) Assignment step: Select a (better) feasible binary solution vector $x = (x_S)_{S\in\mathcal{S}}$. Recall that a feasible x implies that x satisfies a k-partition constraint.
2) Update step: Update the objective to reflect the cost of the selected subsets, i.e. update $\sum_{S\in\mathcal{S}} c_S x_S$ by computing c_S for each selected subset S.

The algorithm terminates when no change is made during the assignment step; that is, a better feasible binary solution vector cannot be found by the algorithm. We observe that, after each iteration of Lloyd's algorithm (after the update step), the algorithm maintains a feasible solution to the corresponding set partitioning problem. This observation also motivates related methods for solving set partitioning problems, such as local search methods, which maintain feasible solutions at each iteration.

B. Extension: regularized centroid-based clustering

We can now discuss our first extension beyond the classical centroid-based clustering to the regularized setting.

Definition 2 (Relative cost): We call $c_{S,r}$ a relative cost if it takes the form

$$c_{S,r} := \min_{x\in\mathcal{X}} \sum_{s\in S} d(s,x) + r(x)$$

where $r : \mathcal{X} \to \mathbb{R}$ denotes a *relative term*.

Corollary 2 (Regularization is relative-cost): Denote regularizer $r : \mathcal{X} \mapsto \mathbb{R}$. Then adding $\sum_{j=1}^{k} r(\mu_j)$ to the objective of centroid-based clustering (Problem (8)-(11)) and the corresponding definition of μ_j is equivalent to the k-set partitioning problem (4)-(7) with the relative cost $c_{S,r} := \min_{x\in\mathcal{X}} \sum_{s\in S} d(s,x) + r(x)$. That is, the regularized centroid-based clustering problem is equivalent to the *relative-cost* metric k-set partitioning problem.

C. Extension: k-median on graphs

Instead of embedding our universe $\mathcal{U}$ in a metric space, we can embed it in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Then, set partitioning has an equivalence to clustering according to graph distance instead of a metric. We define an analogous cost called *graph median*.

Definition 3 (Graph median): Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $v \in \mathcal{V}$ is the graph median of a subset $\mathcal{S} \subseteq \mathcal{V}$ if

$$v = \arg\min_{u\in\mathcal{V}} \sum_{s\in S} \tilde{d}(s,u)$$

where $\tilde{d}(s,u)$ denotes the shortest path distance between nodes s and u.

Claim 2 (Median on graphs): The graph median for a subset S of cardinality m can be computed in $\mathrm{O}\big(\log(m(|V|+|E|)\log|V|)\big)$.

D. Embeddable cost substructures

In the more general setting of metric set packing, where the solution need not cover the entire universe, we may draw a connection to how clustering methods handle outliers. Additionally, by making use of recent advances in semi-supervised clustering and spectral clustering, we propose to further identify a subclass of set partitioning problems that may be embedded in a cluster learning framework for efficient computation and probabilistically optimal set partitioning. When we characterize set partitioning problems in terms of their underlying cost structure and additional constraints, we conjecture a formal parallel between set partitioning and clustering with the following *embeddable substructures*: metric spaces, relative cost, pairwise affinity, m-sets, and singleton sets. For each embeddable substructure, we plan to prove the equivalence to their clustering counterpart and give an example efficient and often parallelizable algorithm. We also discuss when the substructures and algorithms may be combined and study clustering methods beyond centroid-based methods, and we hope to contribute a unifying theory of semi-supervised clustering in terms of the underlying optimization problems.

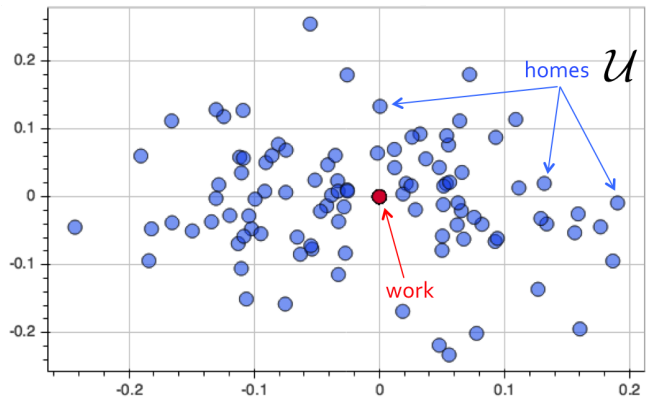

Fig. 2. Ridesharing meetup problem. 100 normally distributed users, with a single destination at the origin $(0, 0)$.

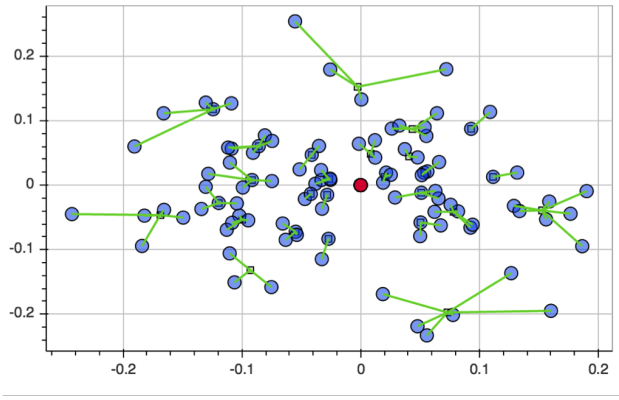

Fig. 3. Regularized geometric k-median for the ridesharing meetup problem results in 26 ride share groups from 100 users.

III. Case Study: Ridesharing Meetup Problem

We have been studying the cluster learning approach to solving a large-scale ridesharing problem. The ridesharing problem is classically formulated as a set partitioning or set cover problem [18], with complex costs dependent on a road network and the constraints of the participants. When these costs are decomposed into embeddable substructures, we hope to demonstrate the ability of fast clustering methods to solve classically combinatorial problems.

We first study a simplified setting of sharing rides when commuting to work in the morning. We restrict to a single destination area. Consider the setting where users in a ride share group agree to meet up at a location, the users travel there individually, and then they share a single vehicle from the meetup point to work. We formally define the ridesharing meetup problem (see Figure 2):

Definition 4 (Ridesharing meetup problem): Let $\mathcal{U}$ denote the universe of users $u \in \mathcal{U}$, who live in $\mathcal{X} = \mathbb{R}^2$ and work at the origin $(0, 0) \in \mathcal{X}$. Let $\mathcal{S}$ be the collection of possible ride shares. We wish to select the ride share groups and their respective meetup points that minimize the total vehicle distance.

This is very naturally a set partitioning problem. The cost of each subset $S \in \mathcal{S}$ can be written as $c_S = \min_{x \in \mathcal{X}} \sum_{s \in S} \|s - x\|_2 + \|x\|_2$. The first term is the total distance traveled by each member of the ride share to the meetup point, and the second term is the distance traveled by one shared vehicle. Assume (for now) that vehicles have infinite (or large) capacity. By Corollary 2, we observe that we have a relative cost that is equivalent to an ℓ_2-regularized loss in the clustering optimization.

Then, by Proposition 1 and Corollary 2, we formulate this problem as a regularized geometric k-median problem, which takes the following objective

$$\min_{(T_1, T_2, \ldots, T_k)} \sum_{j=1}^{k} \sum_{x \in T_j} \|x - \mu_j\|_2 + \lambda \|\mu_j\|_2 \qquad (14)$$

When $\lambda = 1$, this formulation exactly minimizes the total vehicle distance. Thus, we can solve the ridesharing problem with an Expectation Maximization (EM) style method. We present preliminary results in Figure 3, resulting in 69% less vehicle distance (compared against singleton sets, i.e. no ridesharing). We may further extend the ride share meetup problem to networks (instead of Euclidean space) by Claim 2.

IV. Conclusion

We have presented the first formal connection between set partitioning and clustering, with the goal of identifying a subclass of set partitioning problems that can benefit from algorithms and theoretical guarantees for clustering problems. We prove that classical centroid clustering problems are equivalent to metric k-set partitioning, and we present a simplified ridesharing problem that uses this formalism to achieve fast algorithms for a classical combinatorial optimization problem.

In reality, the constraints and costs of a large-scale ridesharing problem will be significantly more complicated than the setting we studied in this article. As suggested above, distances will be non-Euclidean, there will be multiple destinations and time windows, and participants will have different preferences about time, money, and social factors, as well as different types of constraints. A real-world ridesharing system may have extrinsic properties such as monetary incentives by means of payments between participants or time incentives by means of high-occupancy vehicles (HOV) lanes. In some settings, a linear pickup order may make more sense than a participant meetup scheme.

Moving forward, one promising approach is to decompose the complex overall ridesharing problem into simple problems such as those studied in this article, as a way to both compose principled building blocks for complex system design and achieve theoretical guarantees. Further investigation is needed to determine how solutions of the decomposed problems might be merged together meaningfully to solve the overall problem. In another direction of work, we seek to understand how multiple complex cost structures and embeddable substructures can be supported within the combinatorial optimization framework such that

efficient algorithms are still admissible.

REFERENCES

[1] Karla L Hoffman and Manfred Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993.

[2] Hai D Chu, Eric Gelman, and Ellis L Johnson. Solving large scale crew scheduling problems. In *Interfaces in Computer Science and Operations Research*, pages 183–194. Springer, 1997.

[3] Paolo Santi, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven Strogatz, and Carlo Ratti. Taxi pooling in new york city: a network-based approach to social sharing problems. *arXiv preprint arXiv*, 310, 2013.

[4] Péter Biro, David F Manlove, and Romeo Rizzi. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications*, 1(04):499–517, 2009.

[5] RR Vemuganti. Applications of set covering, set packing and set partitioning models: A survey. In *Handbook of combinatorial optimization*, pages 573–746. Springer, 1999.

[6] Manfred W Padberg. On the facial structure of set packing polyhedra. *Mathematical programming*, 5(1):199–215, 1973.

[7] Refael Hassin and Shlomi Rubinstein. An approximation algorithm for maximum triangle packing. *Discrete Applied Mathematics*, 154(6):971–979, 2006.

[8] Jianxin Wang and Qilong Feng. An o*(3.523 k) parameterized algorithm for 3-set packing. In *Theory and Applications of Models of Computation*, pages 82–93. Springer, 2008.

[9] Esther M Arkin and Refael Hassin. On local search for weighted k-set packing. *Mathematics of Operations Research*, 23(3):640–648, 1998.

[10] Barun Chandra and Magnus M Halldorsson. Greedy local improvement and weighted set packing approximation. *Journal of Algorithms*, 39(2):223–240, 2001.

[11] Kathryn A Dowsland. Some experiments with simulated annealing techniques for packing problems. *European Journal of Operational Research*, 68(3):389–399, 1993.

[12] Xavier Gandibleux, Xavier Delorme, and Vincent T'Kindt. An ant colony optimisation algorithm for the set packing problem. In *Ant Colony Optimization and Swarm Intelligence*, pages 49–60. Springer, 2004.

[13] Xavier Delorme, Xavier Gandibleux, and Joaquin Rodriguez. Grasp for set packing problems. *European Journal of Operational Research*, 153(3):564–580, 2004.

[14] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[15] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[16] Hugo Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.

[17] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[18] Ece Kamar and Eric Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. In *IJCAI*, volume 9, page 187, 2009.