[10pt]article [doublespacing]setspace mathtools array booktabs float amsmath amsfonts amssymb amsthm enumitem graphicx xcolor . named..5em3 named *nthmTheorem Optimizing the Robotics Closet Sasha Krassovsky, John Lim, Andrey Ryabtsev document

Introduction The Husky Robotics team is a team of both undergraduate and graduate students from UW building a Mars rover to compete in the University Rover Challenge. The rover must perform a variety of tasks, such as autonomous navigation, soil collection, and typing on a keyboard with a robotic arm. The team is divided into subsystems: Chassis, Arm, Science, Electronics, Software, and Manufacturing. The subsystems work independently but collaborate to create the robot. Given all these tasks and the relatively short amount of time given to build out the functionality, efficiency is key when it comes to the build process.

The robotics team has only a small closet in the Mechanical Engineering Building. The closet stores the robot along with all of the tools, electronics, sheet metal, carbon fiber, and plastics that go into building the robot. Each item belongs to one of the subsystems, and is usually stored near other items belonging to that subsystem. Unfortunately, the robotics closet is not particularly organized, and members often spend on the order of 10 minutes looking for the thing they needed. Our goal is to find a reorganization plan for the items in the closet such that expected duration of each closet visit is minimized and items belonging to the same subsystem are in the same connected component on the shelves.

The closet has two shelves, one on each side. One side stores items in uniformly-sized boxes while the other side has several other classes of items, such as soldering irons, and boxes of various sizes. The rover is stored in the middle of the room on a big cart. Behind it is the robotic arm and the vscience station. In the back are the sheet metal, various metallic cylinders, wood, and rolls of carbon fiber. By examining the frequencies of retrieval by each team and treating the closet as a grid, we should be able to produce a better layout of the closet.

center [scale=0.15]closet.jpg

Related Work Deepesh Singh's A Beginners guide to Shelf Space Optimization using Linear Programming india explains an integer programming formulation for finding the optimal layout of a store that maximizes sales. The guide organizes the store into several racks, each with a certain number of shelves. One key assumption is that only one product can fit on a rack. The model uses a table showing how many sales are generated by each item depending on which rack it is placed. The objective function is then the profit generated by a given arrangement. Arrangements are represented by a matrix, with each a_{ij} representing whether product j is on shelf i. If it is, the a_{ij} is 1 and 0 otherwise. The guide shows how to maximize this profit function. Although this approach is a good start, our problem differs in several ways: we can fit more than one box on each shelf, and our goal is to minimize time of retrieval rather than maximize profits.

Simplifying Assumptions Most items in the robotics closet are in boxes. As a result, we assume that all items are rectangular prisms. Although items such as the chemicals or soldering irons are not strictly rectangle-shaped, it would be strange for another item reached into the empty areas of the items' bounding boxes. Taking advantage of this, we approximate all items as being rectangular prisms, and measure their bounding boxes.

We also assumed that certain parts of the robotics closet are not to be moved. For example, the shelf containing aluminum brackets and blocks for machining was excluded from the model entirely. This is due to the cumbersome nature of moving hundreds of unboxed chunks of metal, and due to opposition to its transportation by team members who frequently use it. The food pantry portion as well as some non-food items that made their way into it was similarly excluded after brief deliberation. Excluding parts of the closet from the model does not reduce the validity of the remaining optimization since the access to other closets is unaffected by the contents of these.

We also assumed that there can be on average 3 items in the same 1-D linear segment. This means that by using some combination of placing items behind and above other items, we can fit 3 items in that segment. This is a rough estimate that we came to empirically. This makes sense because some shelves are very deep, and all shelves fit at least 2 of the boxes vertically, since the boxes are 12.5 cm tall, and all shelves are taller than 25 cm.

The previous assumption let us assume for our model is shelves have one unit of depth and one unit of height. In other words, the model does not have to take into account insertion of items one behind another. To model height, we make use of the previous assumption to "duplicate" shelves, defining the duplicates to be above the original one. We can model depth in the same way. Model We collected data from the robotics closet by measuring each item and shelf. Each shelf s is represented as an ordered triple (d, L, c), where d is the distance from the door in "shelf-lengths", L is the length of the shelf, and c is the connected component to which the self belongs.

Each item is represented with an ordered triple (l, p, s), where l is the length of the item, p is the probability that the item is selected, and s is the item's subsystem. Notice that we represent each item as a 1-dimensional length. This is valid because we assumed that each shelf had a single unit of depth and a single unit of height. As a preprocessing step, minimize the length. Since we cannot always rotate an item into a horizontal orientation, for an item with dimensions $a \times b$ the item's 1-dimensional length is given by

$$l = \min(a, b)$$

Let there be n_0 shelves and m items, k subsystems, and n_c be the number of connected components. In order to simulate stacking and placing behind, we multiply the number of shelves by our assumed average number of items per 1-dimensional block. In our model, we chose 3 as our constant. Thus, the total number of shelves in the model is $n = 3n_0$. Our decision variables lie in two matrices

$$W = (w_{ij}), 1 \leq i \leq n, 1 \leq j \leq m$$

$$C = (c_{ij}), 1 \leq i \leq l, 1 \leq j \leq k$$

Each w_{ij} and c_{ij} is a binary variable, i.e. each $w_{ij}, c_{ij} \in 0, 1$. Each w_{ij} is 0 if item j is not on shelf i and 1 otherwise. c_{ij} is 1 if subsystem j lies in connected component i and 0 otherwise. C is a $n_c \times k$ matrix because there are l connected components and k subsystems. Note about notation: (l^i_k indicates the length of item k while l^s_k indicates the length of shelf k. I is the set of items and S is the set of shelves. For any item x, p_x and s_x represent the item's probability and subsystem respectively. Let there be n_0 shelves and m items, k subsystems, and n_c be the number of connected components.)

The model is subject to the following constraints:

Explanation of Constraints: enumerate

T he sum the lengths of all of the items on a given shelf must be less than or equal to the maximum length that the shelf can support. We multiply by w_{ji} so that items not included on the shelf are not counted.

E ach column of W can only have one 1, because we cannot put an item on two shelves simultaneously. Since each row represents a shelf, we simply traverse the whole column and ensure its sum is exactly 1. Notice that if we had put ≤ 1, the constraint would have been invalid because then no items would have been included. Thus, this constraint implicitly enforces the constraint that every item be included.

E ach column of C can only have one 1 because all items belonging to a subsystem must be in the same connected component, and therefore a subsystem can be in exactly one connected component. Similarly to constraint 2, summing this and ensuring that the sum is 1 enforces the idea that every subsystem is included.

T he intuition behind this constraint is that either all items belonging to a subsystem are in the same connected component, or none are. Thus, we use the decision variable c_{ji} to choose whether the sum is 0 or not.