Algorithms for healthy and fair distribution of food

Catarina Toscano Monteiro Ritto Corrêa

Instituto Superior Tecnico

1 Introduction

The fair distribution of goods is a widely studied problem due to its social importance and its academic richness. REFOOD is an association which has the redistribution of food as its core values. In this thesis, it will be studied different distribution techniques in the REFOOD environment. Namely, we will try to understand what is the best way to make nutritional choices when limited with the food available that was donated.

To accomplish this we will need to simulate a set of easy to follow heuristics. Starting the division by aliment, or filling each basket separately are examples of possible solutions. Besides that, we will define a cost function which will evaluate the performance of the heuristics results, knowing that the one that has the lowest cost is the one that if followed by the volunteers will provide the best nutritional results. In order to have a comparable perspective of the result, we will also create an algorithm which computes the optimal solution given any load. This will allow us to see how much the heuristic solution deviates from the optimal one.

2 Background

In this thesis, we will need to handle optimization problems to compute the optimal solution. In the optimal solution algorithm, the goal is to return the result that minimizes its distance to the reference value.

Considering $\boldsymbol{r}$ as the result vector of final nutrients, $\boldsymbol{d}$ as the DRI vector, we aim to minimize the euclidean distance between $\boldsymbol{r}$ and $\boldsymbol{d}$. This is the same as to minimize the norm of $\boldsymbol{r - d}$, which is the same as minimizing $\boldsymbol{(r-d)^T(r-d)}$.

Our problem is then to minimize a quadratic function and is linear constrained since the solution for each food item needs to be between zero and a defined load. There are multiple ways to minimize linear constrained quadratic function [1].

CVXOPT [2] is a solver available in python specific for minimizing quadratic functions subject to linear constrains. The solver receives problems in the form:

$$\begin{aligned} &\text{minimize } (1/2)x^T Px + q^T x \\ &\text{subject to } Gx \leqslant h \\ &\qquad Ax = b. \end{aligned}$$

Where $\boldsymbol{x}$ is the optimization vector solution, $\boldsymbol{P}$ and $\boldsymbol{q}$ define the quadratic space and $\boldsymbol{G}$, $\boldsymbol{h}$, $\boldsymbol{A}$ and $\boldsymbol{b}$ define the linear equality and inequality constrains of the problem.

In Figure 1, considering the ellipses as the representation of the quadratic problem and the shadowed polygon as the linear constrains, we can see that the polygon does not intersect the middle of the ellipses. This means that there is not a perfect solution. However, all the values inside the polygon are acceptable values for the problem. Despite that, the $\boldsymbol{x'}$ point is the one that has the minimal

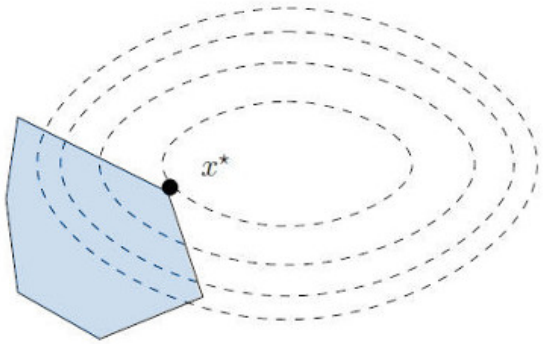


Fig. 1: Linear constrained quadratic problem [3]

distance to the center of the ellipses and therefore, the lowest error. This point is what the solver will return.

3 Proposal

Our proposed solution is divided in different phases. First, we calculate the optimal solution algorithm and a cost function. The algorithm receives a load containing the foods to be divided and returns the best distribution it can find. The cost function receives the final distribution and calculates how far away it is from the parameters baseline. Then, we create the algorithms that simulate different heuristics of dividing the load. These algorithms also return a distribution of goods, but according to their heuristics.

3.1 Cost Function

To calculate the cost of each solution, we will consider the euclidean distance between the vector with the nutritional results of the solution and the DRI vector.

However, since the scale is not the same for each nutrient and the range differs quite a lot, we will normalize the model for each nutrient.

3.2 Optimal Solution

The optimal solution algorithm is a function that receives a load as an input and returns the best distribution possible that optimizes the daily reference intake. This function serves merely as a baseline solution on which the heuristics solutions can be evaluated. However, its use is especially important when the optimal solution deviates considerably from the daily reference intakes.

Our goal is to develop an algorithm where the nutritional values of the solution are as close as possible to the DRI ones. Having

$\boldsymbol{N}$ as the matrix with the nutritional information by aliment,

$\boldsymbol{d}$ as the daily reference intake vector,

$\boldsymbol{x}$ as the result distribution vector,

we want to $\boldsymbol{minimize} \quad \|\boldsymbol{N.x} - \boldsymbol{d}\|$ while ensuring that the quantity given of each food is between zero and the load available.

To calculate the minimization, we used CVXOPT which is a free software package for convex optimization based on the Python programming language [2]. We will use a solver fit for quadratic cone programs. The solver accepts problems of the form:

$$\begin{aligned} &\text{minimize } (1/2)x^T Px + q^T x \\ &\text{subject to } Gx \leqslant h \\ &\qquad\qquad Ax = b. \end{aligned}$$

In order to have our formula in this shape, we will need to manipulate it.

$$
\begin{aligned}
\|N.x - d\|^2 &= (N.x - d)^T (N.x - d) \\
&= x^T N^T N x - x^T N^T d - d^T N x + d^T d \\
&= x^T [N^T N] x - d^T N x - d^T N x + d^T d \\
&= x^T [N^T N] x - 2 d^T N x + d^T d \\
&= x^T [N^T N] x - 2 [N^T d]^T x + d^T d \\
&\propto (1/2) x^T [N^T N] x - [N^T d]^T x
\end{aligned}
$$

3.3 Heuristics

The implemented heuristics needs to be easy enough so the volunteers can follow them without having to run a program. It will be considered two different splitting algorithms: the give-everything and the menus.

The give-everything heuristic is the most easy one and consists in splitting everything by everybody, not taking into account any limit. We computed it by dividing the amount of each aliment by the number of people.

There is another variation of the above heuristic. Since the biggest problem of that heuristic is the lack of an upper limit, we created other heuristic that is limited by aliment, having that it will not give more that certain quantity of each food item.

The menus heuristic is computed through a set of pre-defined menus. The algorithm then assigns one menu for each person. The choice of the menu is made by sorting the menus by cost in advance and choosing the first one that is possible from that list. If it is not possible to give any of the selected menus, the algorithm will give what it cans of the first menu.

4 Results

In this chapter we will present the results from our simulations and make the comparisons between the heuristics.

We can easily simulate this different scenarios just by changing the load. The scenarios will have different load values depending on whether it is being considered one or ten people.

In order to find the better "normal" values, a few tries were made. Ultimately, we found 200 grams to be a compatible value for this scenario and build the other scenarios according to this one. The scarcity scenario has half the load with 100 grams available and the abundance scenario has 500 grams available of each food. The unbalanced scenario is based on the normal one, however we purposely made it weak on protein. This was done by removing all foods that have more than 7.5g of protein for each 100g of food.

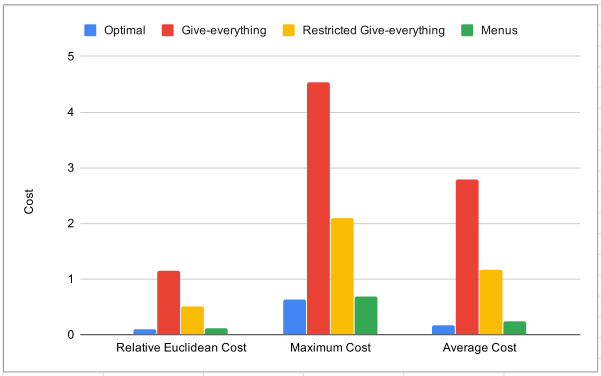
For the ten people simulations, we multiplied the above values by ten.

4.1 One person distribution

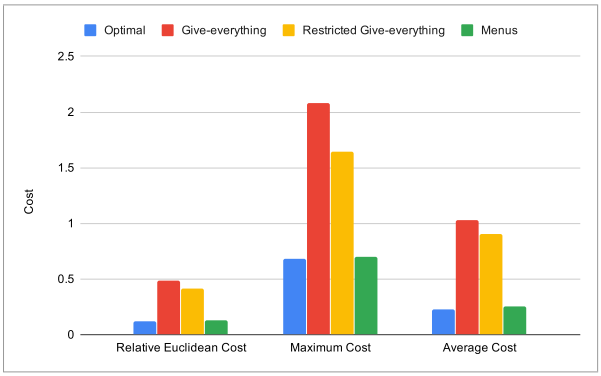
Figure 2 shows the results of the different scenarios only taking one person into account.

In the normal scenario, present in Figure 2(a) we can see that the menus is the heuristic that performs the best. As it was anticipated, the give-everything heuristic is not very good since it has no filter whatsoever. However, it improves deeply in the restricted form.

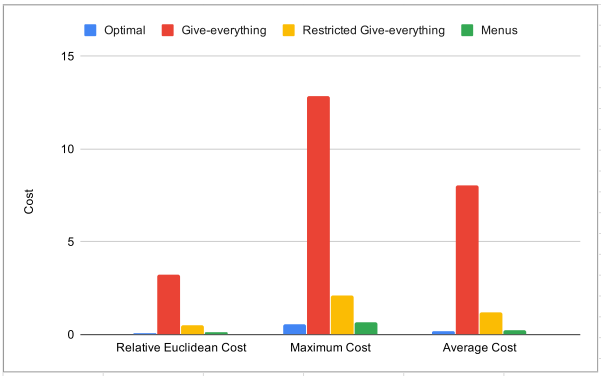
In comparison, in the scarcity scenario, displayed in Figure 2(b), the cost values are much more similar. This is due to the fact that not only is the cost value from the give-everything much lower but also



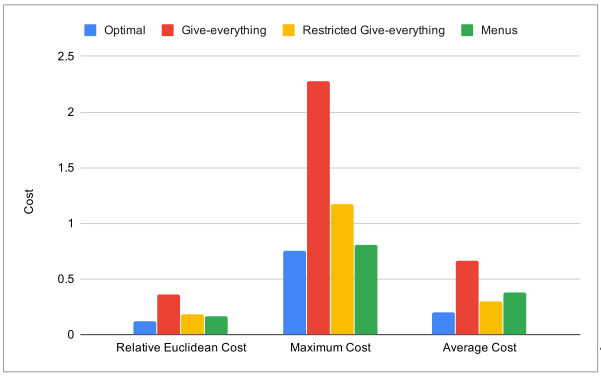
(a) Normal Scenario



(b) Scarcity Scenario



(c) Abundance Scenario



(d) Unbalanced Scenario

Fig. 2: Simulation results for one person

the optimal solution decreases a little since it has more limited choices. Both of the give-everything type heuristics perform better since in the normal scenario there are some foods that deeply affect the cost in a negative way. The menus heuristic performed similarly since in both cases it was possible to deliver the first menu available.

The abundance scenario, showed in Figure 2(c) only changes the results from the optimal solution and the give-everything heuristic. In this last one, the results are obviously terrible since the nutritional values are greatly exceeded. On the other hand, the optimal solution increases its performance since it has much more freedom to optimize the problem.

The menus distribution was the Menu 1 for the normal and abundance scenario and the default option in the scarcity and unbalanced scenario. In these last two scenarios, there was not enough for neither one of the menus so it was given what was possible from the Menu 1.

4.2 Ten people distribution

The optimal solution has different results depending on the load available. Since it is the minimization of a quadratic function, giving an extra amount of a certain food results in an increase of the cost that is more than twice the cost if only half the increase of the extra amount was considered. Therefore, the algorithm will never give more food to one person than another because it will unbalance the cost and it will not be the optimal solution. Due to this, the optimal solution will be the same for each person if considering one person with a certain load or if considering ten people with ten times the load.

Similarly, the give-everything heuristics do not change their results when the increase is the same in the number of persons and load. This because the algorithm splits everything by everyone equally. Since we are considering ten times more people and ten times more load available, the quantity given to each person is also the same.

Therefore, the only heuristic that changes its results is the menus heuristic. In the normal and abundance scenario all the menus delivered were the Menu 1. This makes sense since it was already possible to give the first menu, which is the one with the lowest cost, when considering only one person. In the unbalanced scenario, where there was a lack of protein available, the load available did not have enough quantities to give any of the menus, so the heuristic chose to distribute the default option for everybody. However, in the scarcity scenario, the algorithm was capable of delivering six Menu 1 and four Menu 2. It started to give the second menu when there was no more food left for the first menu but still enough for the second one. Despite this, the menus solution continues to be the one that performs the best.

4.3 Optimal distribution depending on the load

In order to have a better understanding of our problem and the results returned by the optimal solution algorithm we did a bit of exploratory work. The goal was to see how the food choices varied with the load available. To do this, we set the load equal for all foods and made it vary between zero and one kilo with one hundred grams dimension steps.

In Figure 3 we can see that as the load varies in the X axis, so does the quantity selected of each food item. Although this does not have a practical application, we believe it still holds a lot of interest. It allows us to see how the optimal solution adapts with a changing load and gain a better insight of the problem.

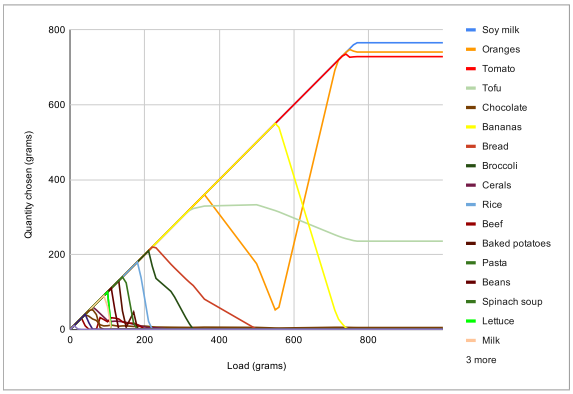


Fig. 3: Optimal distribution with an equal growing load

5 Conclusions

Under the REFOOD environment, this thesis tries to detect and compare different ways to distribute food. Our goal was to have simple heuristics that volunteers could follow when dividing the food. There was also the need to make this heuristics simple so that they would be easy to follow and without depending on a computer that would force the volunteers to maintain a digital stock. Two type of heuristics were created and with the cost function we were able to see that the menus heuristic was the one that performed better in our simulation scenarios. With the use of the optimal solution algorithm, we could see that the menus heuristic was really close to the optimal solution.

As it has been stated along this thesis, there are some future works that could improve and further develop this work. The data and model could be much more sturdy by widening the nutrients and

foods considered. Our current work relies in a small portion of the huge amount of data. It would be interesting to see if an increase of complexity in this matter would translate to better results. It would also be interesting to monitor the distribution over a larger time interval and maintain a memory so that some days could balance other days. Other proposed future work would be to change the model to optimize the solution to a range of possible solutions instead of just the recommended value. Nevertheless, we believe one of the biggest impacts would be to take family preferences into account, making this a social choice problem. Although it is no longer a problem under the REFOOD environment, it would be highly interesting to compare distributions algorithms under this new problem.

References

1. Y.-H. Dai and R. Fletcher, "New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds," *Mathematical Programming*, vol. 106, no. 3, p. 403–421, May 2006.
2. Martin Andersen, Joachim Dahl and Lieven Vandenberghe. (2019, May) CVXOPT python solver. [Online]. Available: http://cvxopt.org/#
3. Stéphane Caron. (2017, April) Linear Constrained Quadratic Problems Optimization. [Online]. Available: https://scaron.info/blog/quadratic-programming-in-python.html