Gift Matching

Ewoenam Kwaku Tokpo

May 2019

1 Introduction

The Santa Gift Matching Challenge is an optimization problem which was originally posted as a kaggle competion in 2017. The problem can generally be described as a linear programming problem or more specifically a variation of the assignment problem. In the problem, Santa clause has 1000 gift types he wants to distribute to 1000000 children. For each gift type, there exist 1000 instances of them. The challenge is to find an optimal assignment of gifts such that both Santa Claus and the children are as happy as possible.

In this report, I have captured details of the initial solution I started with and an improved solution I have developed to improve the performance.

2 Problem

In the problem, 2 datasets are provided; the child_wishlist dataset and the gift_goodkids dataset. The child_wishlist dataset is a 1000000×101 sized dataset. Each row in this dataset represents a child and is uniquely identified by a child_id from 0 to 999999 in the first column. The 100 remaining columns in each row represent 100 preferred gifts of the associated child in their order of preference such that the most prefered gift occupies the first index, followed by the second most preferred gift to the 100^{th} most preferred gift.

The gift_goodkids dataset is 1000×1001 in size. Here, each row represents a gift type and is also identified by it's unique id in the first column. The remaining 1000 columns contain the id of Santa's most preferred children for each gift in the order of preference such that the most preferred child occupies the first index followed by the next most preferred gift until the 1000^{th} most preferred child for that gift.

In addition to the above, 0.5% of the children are triplets; from index 0 to 5000, and 4% of the children are twins; from index 5001 to index 40000. The task is to find a unique assignment of each gift type to each child such that both the children and Santa are as happy as possible. Since there are 1000 gift types and 1000 of each gift type, 1000×1000,it is expected that each child gets exactly one gift. To further complicate the problem, there is the constraint that each set of triplets receive the same type of gift and each pair of twins also receive the same type of gift.

2.1 Evaluation

The evaluation of the solution is based on a formula provided to find the Average Normalized Happiness of both Santa and the children. This is given by the formula:

$$ANH = ANCH^3 + ANSH^3$$

ANH = Average Normalized Happiness
ANCH = Average Normalized ChildHappiness
ANSH = Average Normalized SantaHappiness

$$ANCH = \frac{1}{n_c} \sum_{i=0}^{n_c-1} \frac{ChildHappiness}{MaxChildHappiness}$$

$$ANSH = \frac{1}{n_g} \sum_{i=0}^{n_g-1} \frac{GiftHappiness}{MaxGiftHappiness}$$

- n_c is the number of children. n_g is the number of gifts
- $MaxChildHappiness = len(ChildWishList) \times 2$,
- $MaxGiftHappiness = len(GiftGoodKidsList) \times 2$.
- $ChildHappiness = 2 \times GiftOrder$ if the gift is found in the wish list of the child.
- $ChildHappiness = -1$ if the gift is out of the child's wish list.
- Similarly, $GiftHappiness = 2 \times ChildOrder$ if the child is found in the good kids list of the gift.
- $GiftHappiness = -1$ if the child is out of the gift's good kids list.
- For example, if a child has a preference of gifts $[5, 2, 3, 1, 4]$, and is given gift 3, then
- $ChildHappiness = [len(WishList) - indexOf(gift_3)] \times 2 = [5-2] * 2 = 6$

3 Formulating the problem

The problem is a maximization problem that can generally be formulated as a multi-objective linear programming optimization problem.

$$max \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_c-1} c_{ij} a_{ij}$$

$$max \sum_{j=0}^{n_c-1} \sum_{i=0}^{n_g-1} g_{ji} a_{ij}$$

subject to

$$\sum_{j=0}^{n_g-1} a_{ij} = 1$$

$$\sum_{i=0}^{n_c-1} a_{ij} = 1000$$

$$a_{ji} = \{0, 1\}$$

$A \in R^{n \times m}$ is a binary matrix such that $a_{ij} = 1$ if and only if the i^{th} child gets to the j^{th} gift, otherwise $a_{ij} = 0$. c_{ij} and g_{ji} which are the coefficients of a_{ij} represent the happiness value of the i^{th} child and the j^{th} gift respectively

4 Data Analysis

4.1 statistical insight

- Total number of children = 1000000
- Number of gift types = 1000
- Number of gifts for each type = 1000
- Total number of gifts = 1000000
- Number of unique children most prefered (number 1 position) by santa for each unique gift type = 998.

  Thus, 0.0998% of the children were the first priorities in the gift list. children 469831 and 465382 both appeared twice in the first priority for santa's gift list all the other children appeared exactly once.
- Total number of children in all of santa's gift list = 567377.

  This implies that only 56.73% of children are in Santa's gift list.

  Thus, 432623 children (about 43%) were completely left out of the entire gift list indicating that about 43% of the children will be completely unhappy with Santa's gift list.

- Number of unique gift tpyes most preferred by each child = 999.

  99.9% of gifts were in the first priority in the children wishlist number of children in the entire of santa's gift list = 567377. "Gift" 494 was the only gift not in the first priority of any of the children's wish list

- Total number of children in all of children's wishlist = 1000. Thus, all gifts were in the wishlist of at least one child.

4.2 Proposition 1

maximum possible NSH < 0.4057

4.2.1 proof

This is true even from the relaxed version. From the data, 998 of the children were prioritised first in Santa's list. 432623 children are not in santa's gift list at all.

- $$ANSH = \frac{1}{n_g}\sum_{i=0}^{n_g-1}\frac{GiftHappiness}{MaxGiftHappiness}$$

- $MaxGiftHappiness = len(GiftGoodKidsList) \times 2$.

- $GiftHappiness = -1$ if the child is out of the gift's good kids list.

- For the 998 children prioritised, we can find the sum of their normalised happiness to be:
  $$\frac{2(1000-0)}{2000} \times 998 = 998$$

- This leaves 567377-998=566379 children in Santa's gift list. At best, these children will occupy the next 568 columns after the first column i.e. from column 1 (first column is 0) to column 569. This makes the median column for this approximately 285. So in the best possible case, we can calculate the sum of the normalised happiness for these as:
  $$\frac{2(1000-285)}{2000} \times 566379 = 404960.985$$

- Finally, 432623 children are not in Santa's gift list at all. their Gift Happiness will be -1 each. So that:
  $$\frac{-1}{2000} \times 432623 = -216.3115$$

- Putting all together and dividing by the total number of gift($gifttypes \times numberofgiftsforeachtype$), we can generate the ANSH:
  $$\frac{1}{1000000} \times (998 + 404960.985 - 216.3115) = 0.405742$$

Thus, the maximum possible Normalized Santa happiness that can be achieved even for the relaxed version of the problem will be less than 0.405742

5 Solution

My solution was to reduce the problem to a minimum cost flow problem. I found the Google OR-Tools library very useful in solving the problem as it provides many packages for solving the minimum cost flow function.

5.1 Bipartite Graph

The previous solution posed a couple of issues. The first being that the network graph was constructed by iterating through the children's wish list. This made the final score overly dependent on the ANCH value; we'd later see that the ANSH value has no relevant impact even in the optimal soluion howver, I wanted to address this issue. The second was that a bipartite graph performed better.

I represented the problem as a mincostflow problem using a bipartite graph with no transshipment nodes. The solution can be broken down into three main phases which are stated below and will be subsequently explained in much more details.

1. Building and solving bipartite graph
2. Reassigning sets of twins and triplets with different gifts
3. Re-optimizing the assignment of single children.

5.1.1 Building and solving bipartite graph

As stated above, this implementation is based purely on a bipartite graph with no transshipment node. The graph comprises of two sets of nodes. The first set of nodes are nodes that represent each gift and the second set of nodes represent the children. Each node is assigned a supply or demand which is the quantity of the item available at the node or needed by the node. In this project, I represent a supply by a positive supply and a demand by a negative supply. In this work, because each gift type has 1000 instances, I assigned each gift with a supply of 1000. To solve the constraint of each set of triplet and twins receiving the same gift, I introduced an adjustment in which the first child in each of those sets is assigned a supply of -3 and -2 respectively whilst the remaining children of the set are assigned 0 to mean they receive no gift. The single children are all assigned a supply of -1

The two sets of nodes in the bipartite grah, are connected by edges which connect a node on one side of the graph(gift) to a node(child) on the other side. Each edge between these nodes are assigned two values; the cost of the edge and the capacity. the capacity is the maximum number of items that can be transported over the given edge. In this project, the capacity for each edge to a single child is 1 and for each set of triplets and twins, the capacity of the edge to the first child of the set is set to 3 and 2 respectively whilst edges to the remaining children in their sets are given capacities of 0 since they are not receiving any gifts. The cost of the edge is the cost it takes to transport items over the edge.

In the traditional sense, an edge will be created between each node on one side of the graph to every node on the other side of the graph, a cost of 0 is assigned to an edge if no relationship exists between them. This implementation is however very slow to construct and solve and consumes a lot of memory as explained earlier especially for sparse graphs where very few nodes on each side are connected, which is the very situation in this problem. To overcome this problem and to make sure the performance is not compromised, I built the graph in two stages. Firstly by iterating through the children's wish lists and secondly by iterating through Santa's gift list. These two stages are explained into details in the following subsections.

5.1.1.1 iterating through the children's wish-lists

In this stage I started the graph by iterating through the dataset for the child whishlist. For every child(row) of the dataset, I created an edge between the node of that child and each gift in the child's wishlist(gift in each column in the row). This is shown in algorithm 1. The cost of each edge was calculated as the normalised child happiness(NCH) using a formula which is defined below but before assigning the cost for an edge, the algorithm checks if the child also exists in Santa's list for that particular gift. if this exists then the cost of the edge becomes the sum of both costs(NCH + NGH) using the formula which is defined as:

$$NCH = \lambda * (20 * (100 - gift\_index))$$

$$NGH = (1 - \lambda) * (2 * (1000 - child\_index))$$

Another important thing here is that since it is a mincost flow problem and at the same time we are interested in maximizing the happiness, the cost would have to be negated in order to achieve this.

Algorithm 1 depicts the process of iterating through the children's wish-lists. In addition, For each set of triplets and twins, I calculated the sum of their costs for a particular gift which I averaged and assigned as the edge cost from the particular gift node to the first child of the set The capacity of that edge is also set to 3. Since the remaining children of those sets receive no gifts, the cost assigned to them are irrelevant. This is shown in algorithm 2.

```
foreach $child$ do
    foreach $gift \in child's\_wishlist$ do
        $cost1 \longleftarrow -2 \times (lengthofchild's\_wishlist - indexofgift)$
        if $child \in goodkid\_list[gift]$ then
            $cost1 \longleftarrow -2 \times (lengthofgift's\_goodkid_list - indexofchild)$
            $cost \longleftarrow cost1 + cost2$
        else
            $cost \longleftarrow cost1$
        end
    end
end
End Function
```

Algorithm 1: Algorithm for building graph by iterating through the children's wish-lists

```
foreach $child$ do
    foreach $gift \in child's\_wishlist$ do
        if $child \leq 5000$ then
            if $modulus(child, 3) == 0$ then
                $CH1 \longleftarrow happinessfor(child, gift)pair$
                $CH2 \longleftarrow happinessfor(child_{+1}, gift)pair$
                $CH3 \longleftarrow happinessfor(child_{+2}, gift)pair$
                $GH1 \longleftarrow happinessfor(gift, child)pair$
                $GH2 \longleftarrow happinessfor(gift, child_{+1})pair$
                $GH3 \longleftarrow happinessfor(gift, child_{+2})pair$
                $CH \longleftarrow CH1 + CH2 + CH3$
                $GH \longleftarrow GH1 + GH2 + GH3$
                $cost \longleftarrow cost(CH, GH) * -1$
            else
                $cost \longleftarrow X \geq 1$; since all other children in the set have
                  supply of 0
            end
        end
        if $\geq 5001 child \leq 45000$ then
            if $modulus(child, 2) == 1$ then
                $CH1 \longleftarrow happinessfor(child, gift)pair$
                $CH2 \longleftarrow happinessfor(child_{+1}, gift)pair$
                $GH1 \longleftarrow happinessfor(gift, child)pair$
                $GH2 \longleftarrow happinessfor(gift, child_{+1})pair$
                $CH \longleftarrow CH1 + CH2$
                $GH \longleftarrow GH1 + GH2$
                $cost \longleftarrow cost(CH, GH) * -1$
            else
                $cost \longleftarrow X \geq 1$; since all other children in the set have
                  supply of 0
            end
        end
    end
end
End Function
```

Algorithm 2: Algorithm for calculating costs of triplets and twins

5.1.1.2 iterating through Santa's gift list

In this stage the graph is constructed by iterating through Santa's gift list. For each gift, the algorithm creates an edge between the gift node and each child in it's list if the gift does not exist in the child's wishlist since that would have been catered for in the the first stage. The cost of the edge is assigned based on the normalised santa happiness(NSH) formular defined earlier. Algorithm 3 captures this process.

```
foreach $gift$ do
    foreach $child \in gift's\_list$ do
        if $gift$ $not$ $in$ $children\_wishlist[child]$ then
            $cost \longleftarrow -2 \times (lengthofgift's\_goodkidlist - indexofchild)$
        end
    end
end
End Function
```

Algorithm 3: algorithm for graph by iterating through Santa's gift list

When the entire graph has been constructed, we solve for the assignment with a mincost flow solver. In this solution, we used the mincost flow solver from the Google OR-Tools library. Since we are only interested in the single children together with only one child from each set of triplets and twins, we expect to get an output of 976666 gift-child matchings from this stage:

$$(1000000 - n\_triplets + n\_twins) + \frac{n\_triplets}{3} + \frac{n\_twins}{2} = 976666$$

Upon solving the graph, the children with supply of -3 are assigned three gifts which in a few cases are different. This leads us to the second main phase of the solution where we deal with this problem.

5.1.2 Reassigning twins and triplets

To rectify the problem of children of twin and triplet sets being assigned different gifts, the algorithm first selects the appropriate gift for each set by calculating the cost for each child in the set and each of the different gifts it was assigned. The gift with the best average score among all the children in a particular set is selected and the other gifts that were assigned to the set are deleted. Next, for each set of triplet or twin, since we assigned three gifts to only one of the children in it and assigned no gifts to the rest, we need to take the assigned gifts from single children who have those gifts and reassign them to the children in the appropriate triplet-twin set. This was done by tripling or doubling the gifts for the assigned child for triplets and twins respectively and randomly reassigning single children with those gifts to gifts which were not assigned to any child. Because of the random reassignment of the single children, there was the need to re-optimize the assignment of single children.

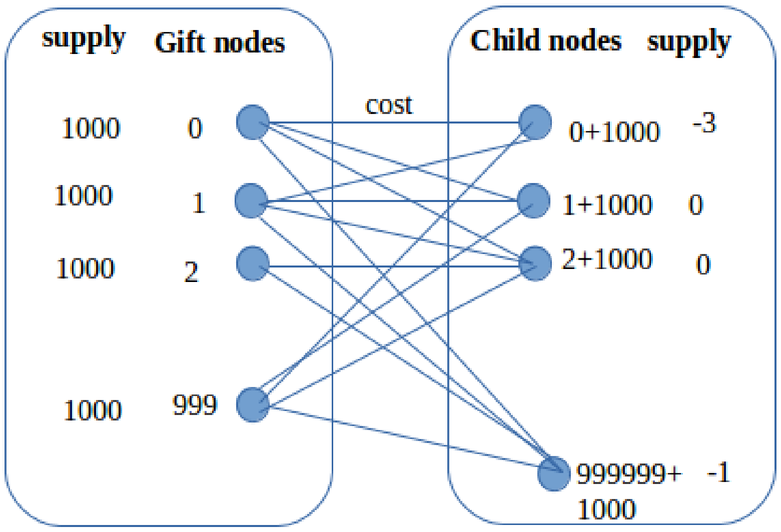

Figure 1: Bipartite graph for gift assignment

5.1.3 Re-optimizing the assignment of single children

To re-optimize the single children's assignment, a list of all the gifts assigned to single children is compiled with which we construct a second bipartite graph with these gifts forming one side of the graph and the single children forming the other side. The same solver as used in the first graph is used except that the capacities for all the edges are 1 in this case and the supply of all the children is -1. After the result is generated, the assignment for the single children in the initial result is replaced with the new assignment.

6 Evaluation of result

I evaluated the performance of the solution based on the performance measure given in section 2. In the following subsections, I describe the evaluation for both the initial solution and the current solution

6.1 evaluation of initial solution

In the previous solution, from the equations in 5.2.1.1, I adjusted the value of λ to different values. The highest value of λ which is 1 produced the best result of Average Normalized Happiness(ANCH) of 0.97636327 and an Average Normalized SantaHappiness score of 0.000005794999 producing an Average Normalized Happiness(ANH) was 0.93075271. The general trend observed was that the lower the value of λ, the more inferioir the result. The pareto front for the initial solution is shown in figure 3.

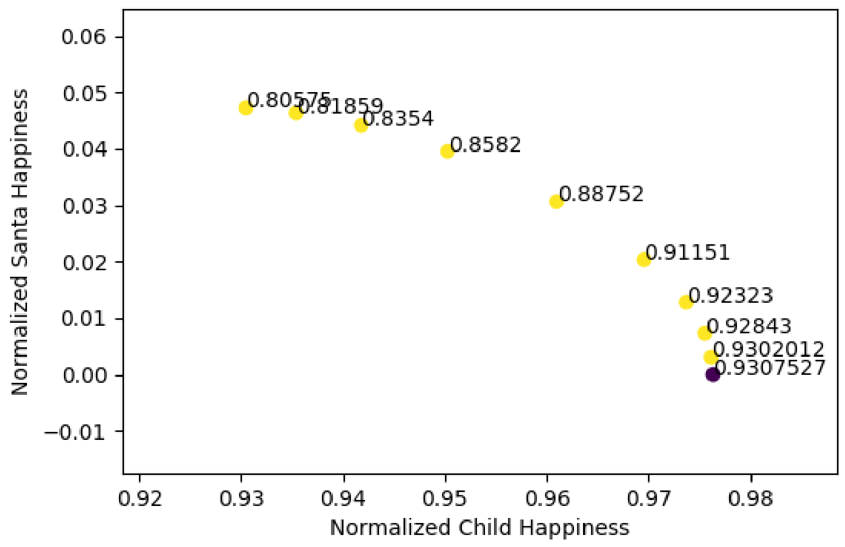

Figure 2: pareto front for the initial solution

Table 1: table of results for various λ values

λ **value**	**ANCH**	**ANSH**	**ANH**
1	0.9763638	0.000005794999	0.93075271
0.9	0.976170	0.00301999	0.930201203
0.8	0.97555	0.007325242	0.928434480
0.7	0.973727	0.0128153284	0.9232376
0.6	0.96958	0.0203495	0.91151883
0.5	0.96099	0.0306181615	0.88752222
0.4	0.95028	0.0395223725	0.85821274862
0.3	0.94181	0.0441553800	0.835475738
0.2	0.93542	0.046337133	0.818597462
0.1	0.93049	0.04723203	0.8057510404
0	0.04750	0.047894850	0.0002170468

However after running the analysis on the dataset, I realized the very low values of ANSH observed even upon varying the value of lambda was actually due to the implementation. The new solution rectifies this problem, however, as we will see from the experiment covered in the next subsection, the value of ANSH has no real impact on the final ANH score in the sense that disregarding the value of ANSH does not alter the optimal result to any significant extent.

6.2 evaluation of new solution

The pareto front for the new solution was very similar to that of the initial values although the values of NSH varied a lot more significantly in response to different the different λ values unlike in the initial implementation. In figure 4 the plot of the pareto front can be observed. Table 2 also gives more details of the various values in the experiment.

From the above, we observe that the optimal solution gave an ANCH score of 0.97664, an ANSH score of 0.000178 and a final ANH score of 0.931546. This corresponds to when λ is set to 1 so that only the NCH score is relevant. In a proof published by team "seed 71" on kaggle [1], they prove that the value of ANSH at the optimal solution is 0.0014091. Since $0.0014091^3 \approx 0$. We can conclude that Santa's happiness score has no significant impact on the performance thus, optimizing the children's happiness solves the problem.

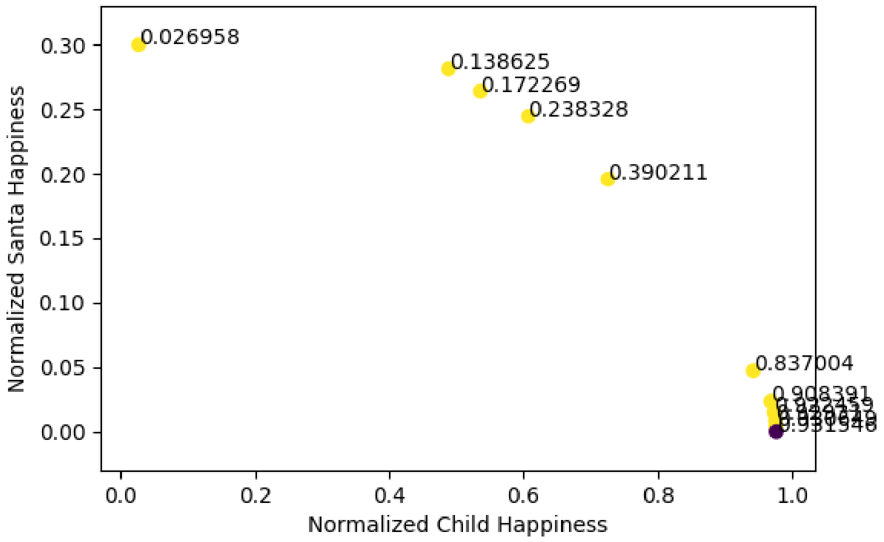

Figure 3: modified graph for gift assignment

6.3 evaluation of updated solution

Table 2: table of results for various λ values

λ **value**	**ANCH**	**ANSH**	**ANH**
1	0.976640	0.000178	0.931546
0.9	0.9763270	0.00449733	0.930649
0.8	0.975708	0.009010	0.929310
0.7	0.9735217	0.015107	0.922459
0.6	0.968530	0.023619	0.908391
0.5	0.9424661	0.047339	0.837004
0.4	0.726039	0.195674	0.390211
0.3	0.607091	0.244355	0.238328
0.2	0.535911	0.263784	0.172269
0.1	0.488292	0.281059	0.138625
0	0.267657	0.299775	0.026958

7 Conclusion

In this report, I have captured the details of the initial solution I came up with to solve the problem and the current improved solution I have come up with. In the new solution the final score is improved from 0.93075271 to 0.931546.

From the experiments and the evaluation of the results, it is apparent that the value of Santa's happiness has no significant impact on the final score. The performance hinges mainly on the children's happiness score(ANCH).

References

[1] https://github.com/KazukiOnodera/Santa2017/blob/master/solution/proof.pdf