Risk & Logistics - Assignment 2

Darragh Ferguson, Mayez Haris, Simeon Horner, Niharika Peddinenikalva

March 2024

Contents

1 Introduction

Tartan Trade are looking to improve the layout of one of their warehouses to minimise total walking distance when collecting products from the shelves and delivering them to the packaging area. Their current warehouse floor plan is shown in Figure 1. Given a warehouse layout and a list of forecasted orders, this report aims to provide recommendations about methods that could be used to determine a suitable initial allocation of product groups to shelves and possible improvements to these initial allocations. This allocation problem is solved in conjunction with an attempt to obtain optimal routes for pickers to take for each order.

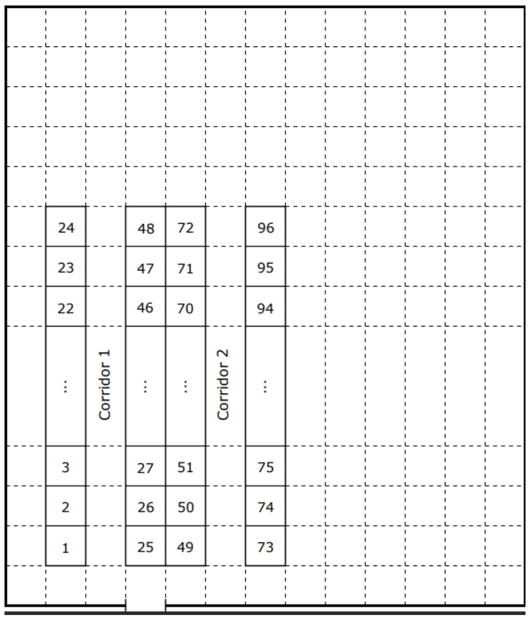


Figure 1: Initial floor plan of Tartan Trade's warehouse.

This problem is an Integer Optimization problem which involves two subproblems:

- Routing Problem: Given an allocation of products to shelves, we find the optimal route that the pickers would use to pick products from shelves for each order such that the total distance covered across all orders is minimized.
- Storage Assignment problem: Optimizing the allocation of products to shelves to minimize the shortest routes to pick products for the order.

The routing problem is similar to the travelling salesman problem (TSP), where finding the optimal route for each order is akin to solving an individual travelling salesman problem. Since the TSP is known to be NP-hard, the routing subproblem is NP-hard. We first formulated the Integer Programming model of the combined problem (see Appendix A) and attempted to solve this in Xpress, where we solved each of the routing problems as subproblems separately for each order. However, it was quickly discovered that although such exact methods can be useful in providing optimal solutions, they are also very computationally expensive. Hence, heuristic algorithms were needed to maintain reasonable computation times. The heuristic algorithms used in this report are briefly described below.

- Given an allocation of products to shelves, we implemented a nearest neighbour (NN) algorithm to find a (greedy) shortest route for each order which can then be used to calculate the total walking distance over all orders.
- We used three construction heuristics to construct an initial allocation of products to shelves. One was a random allocation heuristic and two were greedy heuristics. We then compared the results of the total walking distances (calculated using NN) for the three heuristics.

- To try to improve a current allocation, we used a *local search heuristic (LSH)*, namely the *variable neighbourhood search (VNS)* algorithm, to introduce modifications to the storage allocation in search for better allocation. The objective was to accept neighbourhoods which could decrease the total walking distances.
- We also investigated how some differences in the layout of the warehouse may impact the total walking distances, e.g. adding horizontal aisles between shelves.

2 Nearest Neighbours Algorithm

The problem of calculating the total walking distance given an allocation of product groups to shelves could be viewed as an instance of the multiple travelling salesman problem. Here, given $2,000$ different orders, we had $2,000$ different routes to determine. While a solution approach could be taken with exact methods ideally, as discussed previously, obtaining an optimal solution for all $2,000$ orders was extremely computationally expensive and thus impractical. Computing the optimal routes for just 100 orders took upwards of $3600s$ (1 hour), so using this method for all orders was highly unreasonable. Hence, a heuristic approach was taken when constructing the route for each order. This guaranteed that the routes were feasible (as per the constraints defined in the integer programming formulation), but there was no guarantee that these routes were optimal for each order. We found that approximately 33% of the $2,000$ orders included 1 or 2 products. Thus, the *NN* algorithm could find optimal routes for these orders by the methodology used in this algorithm.

This path-finding greedy heuristic algorithm was called the nearest neighbours (*NN*) algorithm. Beginning at the packaging area (the door), for each order, the distances to each shelf corresponding to the products in that order were compared and the closest shelf was selected as the next shelf to visit. Then, the next shelf visited was the closest shelf which contained a product in the order that was yet to be picked. This process was then repeated until all required products for the order had been picked, at which point the picker returned to the packaging area. This was implemented while accounting for certain constraints in the problem, such as pickers being unable to go from a shelf back to itself, visiting each shelf in the order exactly once, not visiting shelves that were not in the order, and starting and finishing the route at the packaging area. While *NN* did not necessarily result in the optimal route for each order, it was significantly quicker in obtaining routes than exact methods. Given Tartan Trade's current allocation (*CA*) of product groups to shelves, the total walking distance (calculated with *NN*) for all 2000 orders was 290,706m (290.71km), calculated in 0.056s.

3 Construction Heuristics

To construct an initial allocation of products to shelves, we used 3 different construction heuristic algorithms. These included the *random allocation (RA)* algorithm, the *most common product (MCP)* algorithm, and a *greedy allocation (GA)* algorithm.

Each algorithm was implemented with some considerations. Mainly, since there were only 90 products and 96 shelves, it was important to consider whether all shelves needed to be filled. As per Tartan Trade's requirements, each product could be placed on a maximum of 2 shelves in an allocation. It was found that in general, filling all shelves minimised the total walking distance, as more potential routes were available to pickers for all orders. To validate this, we performed 10 tests of adding random products to the 6 empty shelves one at a time in *CA* and investigated the change in total walking distances as each product was added to an empty shelf. The results of this validation method are seen in Figure 2. The black dotted line represents the total walking distance under the current allocation *CA* when only 90 shelves were filled. Each solid, coloured line corresponds to a test. We observed that, in each test, adding a random product increased the number of remaining shelves that had been filled and consequently resulted in either a decrease or no change to the total walking distance.

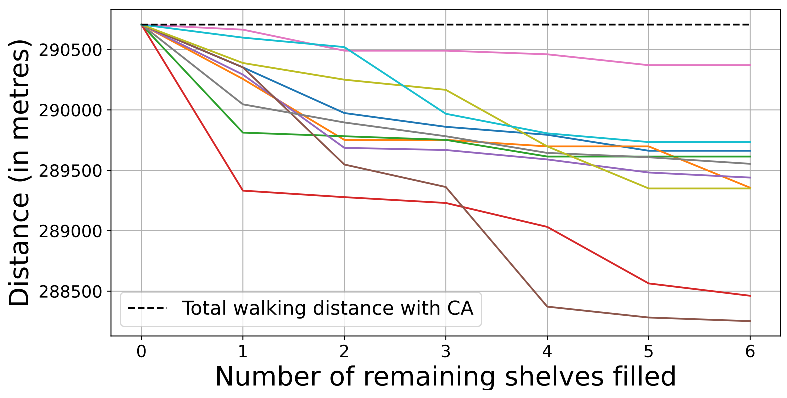


Figure 2: Total distance when products are added randomly to the empty shelves in *CA*.

Random Allocation Algorithm

This algorithm was fairly straightforward to implement. It began by assigning the 90 products to one shelf each at random. Then, if we required the extra shelves to be filled, the algorithm could assign random products to the remaining 6 shelves while ensuring that no product was assigned to more than 2 shelves and each product was assigned to at least one shelf. Obtaining an allocation using this method was extremely quick, taking only 0.0002s to allocate products to the shelves, with all extra shelves being filled. Using this initial allocation, the total walking distance was 278,652m (278.65km), calculated in 0.065s using *NN*. The results from this and the other construction algorithms are detailed in Table 1.

Most Common Product Algorithm

This algorithm was a greedy construction algorithm. First, an ordered list of the most common products among all of the orders was obtained, where the first product on the list was most commonly ordered. Then, an ordered list of the shelves closest to the door was obtained where the first shelf on the list was the closest to the door. Then, for the first 90 shelves, the i-th product in the most common products list was assigned to the i-th shelf in the ordered list of shelves closest to the door. Ultimately, the more common products were assigned to shelves closer to the door. If the extra shelves were required to be filled, the algorithm allocated the most common products in descending order of popularity again to the remaining shelves.

As with *RA*, obtaining an initial allocation of products to shelves was very quick with this algorithm too. When we added only 90 products to 90 shelves (i.e., we did not fill any extra shelves, hereby referred to as *MCP0*), the initial allocation was obtained in only 0.0030s. Using this initial allocation, the total walking distance was 205,038m (205.04km), calculated with *NN* in 0.054s.

Similarly, when we constructed an initial allocation using *MCP* with 6 products added after the first 90 shelves were filled (hereby referred to as *MCP6*), the initial allocation was obtained in 0.0032s. Using this initial allocation, the total walking distance was 203,808m (203.81km), calculated with *NN* in 0.098s.

The times taken to construct these solutions were negligible and comparable to *RA*. Furthermore, the total walking distances for both initial allocations were significantly lower than the total walking distance for *CA* by nearly 85,000m to 87,000m. Thus, *MCP* was effective in constructing an allocation which reduced the walking distance compared to *CA* and was computationally efficient.

It can also be seen that adding extra products to the 6 empty shelves in *MCP* decreased the total walking distance from 205,038m to 203,808m. This may suggest that it could be beneficial to fill the 6 empty shelves with the most commonly ordered products.

Greedy Allocation Algorithm

As the name suggests, this algorithm was also greedy. Suppose there is another storage facility outside Tartan Trade's warehouse where all the products are currently stored. The algorithm begins by allocating one product to a shelf at a time, beginning at the shelf closest to the door. It calculates the contribution of the allocation of each product to this shelf to the total walking distance and selects the product allocation which minimizes the total walking distance. This process is repeated for the next shelf in the ordered list of shelves closest to the door, where products are assigned to shelves one by one such that the chosen product allocation minimizes the distance added to the total walking distance at each stage. Similar to *MCP*, if extra shelves were required to be filled, the same methodology was implemented for the remaining 6 shelves as for the first 90 while ensuring that no product was assigned to more than 2 shelves.

This algorithm was considerably more computationally expensive in comparison to *RA* and *MCP*, taking around 424.6231s to construct an initial allocation of products to shelves with all extra shelves filled. Using the initial allocation obtained from this algorithm, the total walking distance was 201,888m (201.89km), calculated using *NN* in 0.084s.

Firstly, it can be seen that the total walking distance from the allocation constructed by *GA* is the shortest of the distances corresponding to allocations from *CA, MCP*, and *RA*. This is because it takes into account relationships between products commonly ordered together. However, in terms of computational costs, it is significantly more expensive than *MCP*. Considering both algorithms provide exactly one allocation once, if the current computational times for *GA* are not too long for Tartan Trade, this algorithm produces a better allocation with shorter total walking distances for all 2000 orders. However, Section 4 discusses the use of local search heuristics which improve upon these allocations. Thus, we look into a further comparison of *GA* and *MCP* in that section.

Construction Heuristic	Total Walking Distance	Run time (s)
RA	278,652	0.0002
MCP0	205,038	0.0030
MCP6	203,808	0.0032
GA	201,888	424.6231

Table 1: The time taken to produce an initial allocation and the corresponding total walking distances for all the construction heuristics.

Comparing The Algorithms

As is evident from the results in Table 1 above, *MCP* and *GA* are construction heuristics that yield higher quality allocations than *CA* as they reduce the total walking distance significantly. While *GA* did provide a better total walking distance for its initial allocation, it was only marginally better than MCP. While *RA* constructed an allocation in negligible time, it is not reproducible and hence not recommended for usage.

4 Local Search Heuristic

Since the initial allocations from our construction heuristics were only feasible and not necessarily optimal, we investigated whether we could introduce modifications to these initial allocations to reduce the total walking distance for all orders. This was carried out by the use of local search heuristics which take an initial feasible solution as input and introduce modifications which may improve the quality of the provided solution. In particular, we implemented a *variable neighbourhood search (VNS)* algorithm (Algorithm 1).

In *VNS*, neighbourhoods are created by performing defined swaps in the provided product-to-shelf allocations. We defined a problem-specific method of performing swaps.

Defining Neighbourhood Structures

We arranged the 2000 orders in descending order of walking distance per order. Then, we considered the first k orders with the largest contribution to the total walking distance, i.e., with the longest

Algorithm 1: VNS

```
input: Initial feasible solution $x$
input: $k$ longest orders to check
Define a set of neighbourhood structures $N$;
while $|N| > 0$ do
    Explore neighbourhoods $N$ of $x$ ;
    Find the best solution $x' \in N$, i.e., $f(x') = \min_{x \in N} f(x)$ ;
    if $f(x') < f(x)$ then
        Set $x \leftarrow x'$;
    end
end
return the best solution $x$
```

routes. Let this ordered list be $\mathcal{K}_d$. For each order $k_d \in \mathcal{K}_d$, we considered the products in the order and the corresponding shelves that were visited. Suppose k_d contains product p_1 on shelf A. We then fix p_1 on shelf A and try to swap another product $p_2 \in k_d$ from shelf C to shelf B (where shelf B is opposite shelf A). To maintain feasibility while carrying out this swap, we exchange the products on shelf C and B so that p_2 now lies on the shelf opposite shelf A. Then, instead of the picker walking from shelf A to shelf C, which may be far apart in the warehouse, now the picker can pick two products in the order from shelves opposite each other. As is shown in Figure 1, shelves opposite each other had no distance between them, and so the total distance for that order was hopefully reduced significantly. We implement such swaps by fixing each product in the order on a shelf and bringing other products in the order to the opposite shelf one at a time. If the swap results in an allocation which reduces the total walking distance, we add this to the set N of neighbourhood structures. This process is repeated for the k longest orders in terms of walking distance.

Implementing VNS

Given an initial allocation x, we define the neighbourhood structures N. Then, if this set is not empty, we choose the allocation $x' \in N$ which has the lowest total walking distance and set this to be the best solution x. Then, we repeat the process of defining and exploring the neighbourhood structures of x in search for a better solution. The algorithm terminates when no better solution has been found (as $|N| = \emptyset$) or if the user-defined time limit has been reached. We implemented a time limit of 3600 seconds (1 hour) in this study.

It is important to note that this method of swaps was not possible for all orders in the problem, as *VNS* was found to be very computationally expensive. Hence, it was only implemented for the k orders with the largest contributions to the total walking distance, i.e. the k longest orders. The value of k was set to be 5 or 10 in all implementations of *VNS* in this study.

Results

As shown in Table 2 and Figure 3, our implementation of *VNS* was successful in reducing the total walking distance for all initial allocations. However, there was a significant drop in the level of improvement for better initial allocations. As shown in Figure 3a, the improvement from *VNS* on *MCP* and *GA* is relatively low compared to the improvement from *VNS* on *CA* and *RA*. Since *RA* is random and non-reproducible in nature, the results are very inconclusive, as it is difficult to determine whether a different random allocation might be significantly different from the one we have considered. Similarly, since the methodology behind *CA* does not consider the popularity of products or the distances between shelves and the packaging area, it is difficult to justify its use instead of heuristics such as *MCP* and *GA*.

The two non-random construction heuristics, *MCP* and *GA*, had initial allocations with relatively shorter total walking distances. Due to the initial allocation being a good solution, *VNS* struggled to find neighbourhoods which could further improve these solutions. This could be attributed to the fact that better neighbourhoods might be harder to find as the construction heuristic produces a high-quality allocation already. This is in agreement with empirical results seen in *VNS*, where the algorithm performs better on lower-quality initial solutions.

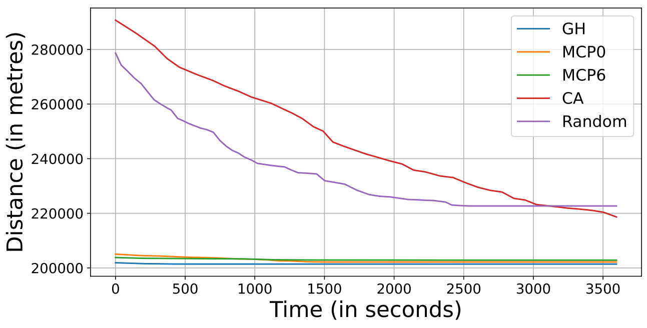
This pattern of better solutions improving less with the implementation of *VNS* is visible in Figure 3b as well. We see that while *MCP0* had a larger initial walking distance than *MCP6*, *VNS* was eventually able to find an allocation for *MCP0* which had a higher quality than the best allocation found for *MCP6*. Figure 3b also provides more evidence to show that *VNS* is relatively less effective in improving solutions from construction heuristics that have a low initial total distance, as we see an extremely small drop in the total walking distance for *GA*. Additionally, *VNS* stops improving the allocations from *GA* in less than 500 seconds, suggesting that it has exhausted its search for neighbourhoods in around 8 minutes, while *CA* was still seemingly going at the 3600s (1 hour) mark.

Further, we see that all neighbourhoods for *RA, GA, MCP0* and *MCP6* were explored within the time limit of 3600 seconds. This suggests that the final allocations were the best ones that *VNS* was capable of finding under the current neighbourhood structures. However, there was still scope for *VNS* to improve the allocation for *CA* if the time limit was longer than 1 hour.

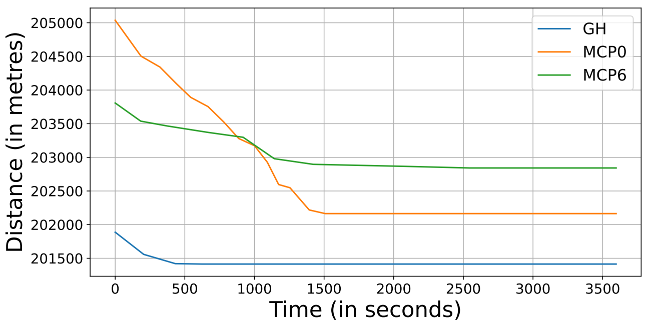
Despite *VNS* finding the best allocations for *RA, GA, MCP0* and *MCP0*, we see that *GA* still yields the best allocation. It can be seen that the final total walking distances for *RA, MCP0* and *MCP6* after VNS was applied were still greater than the initial total walking distance from the allocation using *GA*. Despite the run time for *GA* being longer than the other construction heuristics, Tartan Trade could use this algorithm to yield the best allocations amongst all provided in this study if these run times are within practical timeframes.

Initial Allocation	Total Distance before *VNS* (m)	Final Total Distance after *VNS* (m)	% Decrease	Run time (s)
CA	290,706	218,664	24.78	3,596.74
RA	278,652	222,684	20.08	2,540.17
MCP0	205,038	202,164	1.4	1,507.76
MCP6	203,808	202,842	0.47	2,544.27
GA	201,888	201,414	0.23	801.92

Table 2: The initial and final total distances before and after implementing *VNS* on different initial allocations, respectively.



(a) *VNS* decrease in total walking distance: all allocations.



(b) *VNS* decrease in total walking distance: heuristic allocations.

Figure 3: The effect of *VNS* on different initial allocations over time.

5 Changing The Warehouse Layout

The current layout of the warehouse does not occupy the entire floor area. Since the shelves have been split over 4 columns, the current warehouse layout does allow for some flexibility in the routes, as the shelves are not all in one large column. However, a possible modification that could be made to the layout was the addition of middle aisles, to give the pickers an additional avenue to travel between corridors. The addition of 1 middle aisle to the original warehouse layout is shown in Figure 4.

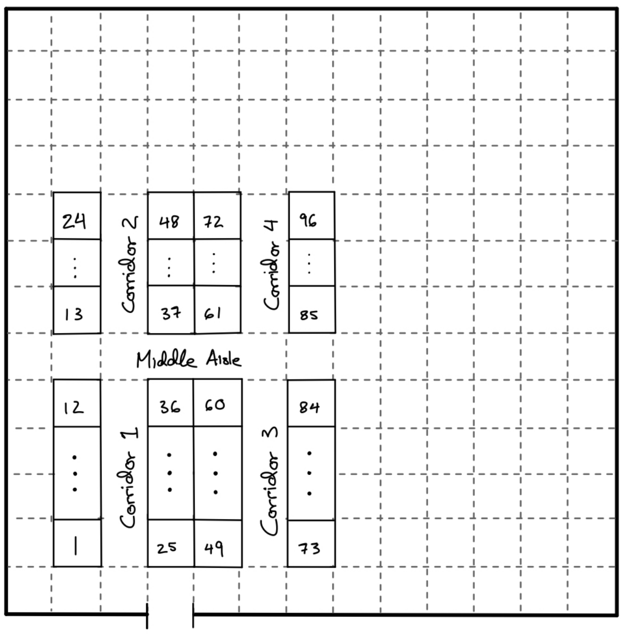


Figure 4: Warehouse layout with 1 middle aisle.

While this was not the only type of modification that could be introduced to the warehouse layout, it is worth noting that it retained the basic column structure from the original layout. Hence, performing swaps to bring products to the opposite shelf was still possible without our *LSH* needing to be modified as well. While this was very restrictive in terms of the different layouts that we could explore, restructuring or redefining our *LSH* was challenging within the timeframe of this project. However, these layout changes prevented the need for changing our *LSH*.

We included the introduction of up to 4 middle aisles, as that is the maximum number of middle aisles the warehouse area allowed for. Since each column contained 24 shelves, it was relatively easy to split the columns evenly into sections with the addition of middle aisles. Table 3 shows the size of the sections for each number of middle aisles added to the warehouse layout. We investigated all 4 layouts to see how they compared to the original layout and to each other.

No. of aisles	**Splits per column**
1	2 sections of 12 shelves each
2	3 sections of 8 shelves each
3	4 sections of 6 shelves each
4	5 sections: 4 sections of 5 shelves each and 1 section of 4 shelves (furthest from packaging)

Table 3: How the introduction of different numbers of middle aisles split each column.

5.1 Calculating New Distance Matrices

Once the new warehouse layouts had been defined, we had to construct new pairwise distance matrices for each of the new layouts. While calculating distances between squares on the grid was fairly straightforward and could be achieved by calculating the Manhattan distances between squares [1], the inclusion of shelves between corridors meant there were obstacles on the grid that the pickers needed to avoid. To account for this, a pathfinding algorithm which involved obstacle avoidance was necessary. Hence, the A* algorithm was found to be appropriate for this task [2]. This algorithm not only accounted for all other shelves as obstacles in the way of the pickers, but it also considered all possible paths to select the shortest one in the end. The distances were all still calculated using the Manhattan distance. This resulted in a new 97×97 (with the inclusion of the door) pairwise distance matrix for all new layouts, as required.

5.2 Using Construction Heuristics on New Layouts

Once the new distance matrices had been calculated, they were used to investigate whether different warehouse layouts impacted the results from the construction heuristics and the *VNS*.

First, we implemented our construction heuristics to find initial allocations given the different warehouse layouts. Then, using *NN*, we calculated the total walking distances for these allocations for each warehouse layout. The initial total walking distances for each warehouse layout for each allocation are provided in Table 4.

Initial Allocation	No aisles (original layout)	1 aisle	2 aisles	3 aisles	4 aisles
CA	290,706	284,820	288,474	294,822	303,012
MCP0	205,038	186,138	186,066	189,336	192,162
MCP6	203,808	185,976	186,036	189,282	192,114
GA	201,888	185,514	185,472	188,088	191,106

Table 4: The initial total distances from different initial allocations for varying layouts.

We see that adding 1 middle aisle to the original layout yields a lower initial walking distance for all four initial allocations. This decrease in walking distance varies between approximately 6,000m to 19,000m which is a significantly large range. Further, we see yet again that the initial allocation using *GA* yields a lower walking distance than all three other initial allocations. We also see that *MCP* in general yields a better initial allocation than *CA* and *MCP6* yields a better initial allocation than *MCP0*.

For *CA, MCP0,* and *MCP6*, adding 2 or more aisles does not reduce the initial total walking distance. It is worth noting that the initial allocations constructed by each of these algorithms stayed the same regardless of the layout. The different walking distances for different layouts arise due to the distances between shelves that are distinct in each layout.

On the other hand, adding 2 aisles to the original layout results in the best allocation obtained by *GA*. This was not the case with the other three allocation types because unlike *GA*, they do not account for the distance between shelves in the construction of an initial allocation. Adding 3 or 4 aisles to the original layout did not improve the initial allocation constructed by *GA*.

Thus, in general, it may be recommended to consider the layout where only one additional middle aisle is introduced. However, if Tartan Trade intends to use *GA* to construct an initial allocation, adding two evenly-spaced aisles to the current layout may reduce the walking distance to the best case we have seen so far.

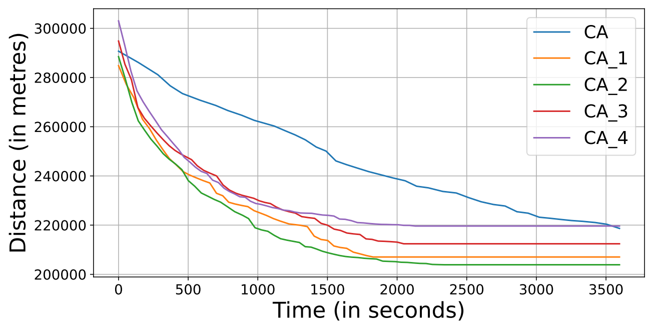
5.3 Implementing *VNS* on New Layouts

Given the initial allocations for the different layouts, we implemented *VNS* to investigate whether it could find improved allocations for each allocation type. It was found that VNS found at least one new allocation for *CA, GA, MCP0,* and *MCP6*, which improved the initial allocation in every layout type. This is demonstrated in Table 5. In Figure 5a, we see that the final allocations found through *VNS* for layouts with 1, 2, and 3 middle aisles are better than the final allocation found for the original layout. The decrease in walking distance is largest when there are 2 aisles added to the original layout. Thus, if Tartan Trade were to continue with the current allocation system, then adding two aisles and using the *VNS* heuristic could help to find a better allocation, reducing the total walking distance to 203,892m.

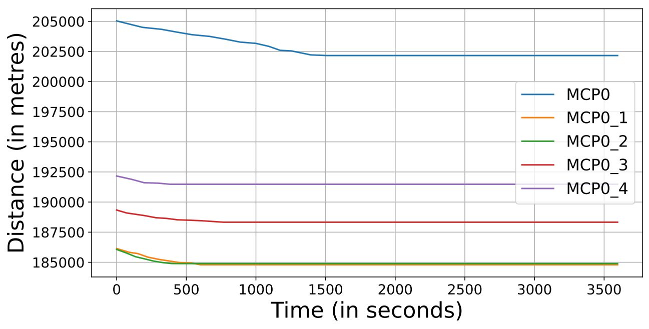
The trend seen in Figure 5b is similar to that seen when *VNS* is implemented on the initial allocations for *MCP6* and *GA* for the different layouts. We can see that it is most beneficial to add 1 or 2 aisles to reduce the walking distance when using the allocations provided by *GA, MCP0,* or *MCP6.* The

Initial Allocation	No aisles (original layout)	1 aisle	2 aisles	3 aisles	4 aisles
CA	218,664	207,028	203,892	212,418	219,630
MCP0	202,164	184,794	184,890	188,328	191,478
MCP6	202,842	184,938	184,993	187,998	191,208
GA	201,414	185,514	184,518	187,638	190,536

Table 5: The final total distances after *VNS* on different initial allocations for varying layouts.



(a) *VNS* on *CA* for varying layouts.



(b) *VNS* on *MCP0* for varying layouts.

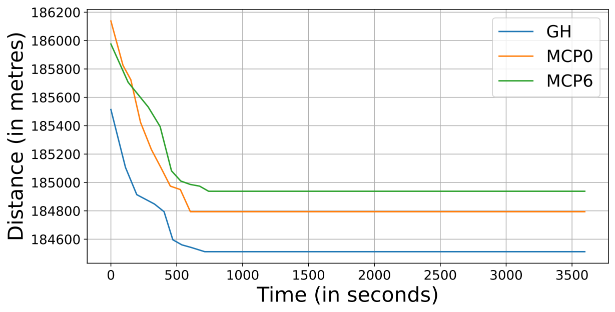
Figure 5: Impact of *VNS* on the *CA* and *MCP0* initial allocations for varying layouts.

decrease in walking distance due to the better allocations found using *VNS* for these initial allocations is nearly negligible for almost all of the new layouts. However, the three construction heuristics still yield better initial allocations (with walking distance $< 200,000$) than the best allocation found after *VNS* was implemented on *CA* in all layouts considered (with walking distance $> 200,000$). Hence, it is recommended that Tartan Trade consider one of the construction heuristics to reduce the current walking distances.

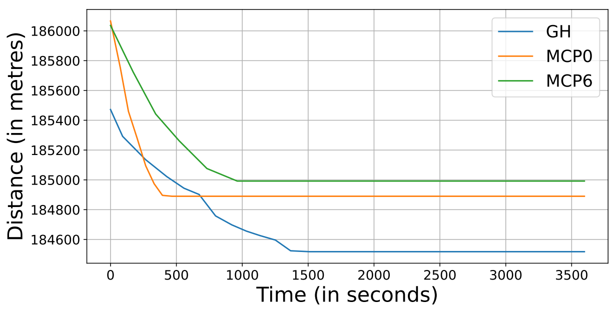
However, it is worth comparing how the three construction heuristics perform in each of the four new layouts introduced. Figure 6 demonstrates that implementing *VNS* on *GA* will always yield a better allocation than *MCP0* and *MCP6*. However, if we are to compare *MCP0* and *MCP6*, *VNS* is able to obtain a better final allocation for the former (or latter) when 1 and 2 (or 3 and 4) aisles are added to the original layout. This could be taken into account when considering layouts and possible algorithms to construct an initial allocation.

5.4 Comparing *VNS* Run Times for Different Layouts

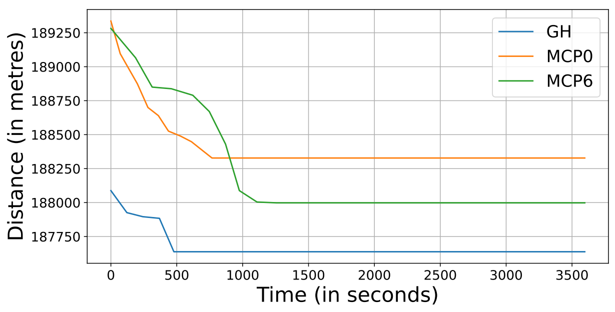
Finally, we can compare the effect of *VNS* on finding the best allocation using the run times. It can be seen that for all four initial allocations and in all 4 new layouts, *VNS* has terminated within the time limit, indicating that it has found the best possible allocation in each case. It should be noted that we considered swaps among the top $k = 5$ or $k = 10$ largest distance orders. This meant that we did not explore swaps for all orders. There is potential to introduce swaps for more orders, which was not investigated in this study due to time constraints. However, it is worth noting that the run times would increase tremendously as a result of increasing the value of k as more orders would then be explored for swaps. This may or may not be practical for Tartan Trade and hence should be taken into consideration when tuning the hyperparameter k. Further, introducing more swaps that are generally seen in the literature surrounding *VNS* for the storage assignment and order picking problem may help to explore more neighbourhoods that could result in even better allocations. However, this comes with computational costs that must be evaluated when redesigning the *LSH*. Exploring alternative *LSH* could also result in finding better allocations, as heuristics cannot guarantee the optimality of allocations found in this study.



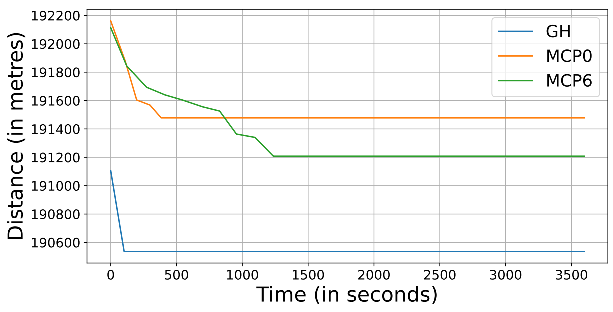
(a) *VNS* for 1 aisle.



(b) *VNS* for 2 aisles.



(c) *VNS* for 3 aisles.



(d) *VNS* for 4 aisles.

Figure 6: *VNS* on the three initial allocations from *MCP* and *GA* for varying layouts.

Initial Allocation	No aisles (original layout)	1 aisle	2 aisles	3 aisles	4 aisles
CA	3596.74	1827.99	2341.34	2047.41	2130.43
MCP0	1507.76	603.12	466.80	767.44	383.78
MCP6	2544.27	741.62	961.56	1256.34	1235.51
GA	801.92	712.34	1510.56	478.00	102.24

Table 6: The run times (in seconds) of the implementation of *VNS* on different initial allocations for varying layouts.

6 Conclusions

We have considered several heuristic algorithms in this study which are useful for calculating the routes pickers must take when picking orders, constructing initial product-to-shelf allocations, and finally implementing a local search algorithm to investigate whether the initial allocations can be improved such that the total walking distance is reduced.

We used the *NN* algorithm to calculate the total walking distance for 2000 orders, where the algorithm used a greedy method to minimize the route length for each order by choosing the next nearest shelf at each stage of the route. While this approximate method was not guaranteed to be optimal, it was computationally efficient and allowed us to focus on optimizing the product-to-shelf allocation instead. This yielded a total walking distance of 290,706m for the current allocation (*CA*) of products to shelves. There is scope to consider other algorithms which may be more effective in finding better routes for order picking that further reduce the walking distance for pickers. However, due to time constraints, we used this as our path-finding algorithm to focus on optimization of product-to-shelf allocation

With the ability to calculate the "shortest" route for each order, we designed construction algorithms to create an initial allocation of products to shelves. It was found that these algorithms, specifically *GA*, *MCP0* and *MCP6*, yielded initial allocations which improved upon the current system being followed by Tartan Trade as the total walking distance was reduced under the allocations from the construction algorithms to at most 205,100m. Further, while the computational costs for producing an initial allocation through *GA* were significantly higher than *MCP*, it did yield a better allocation with a total walking distance of 201,888m.

Given these initial allocations, *VNS* was used with problem-specific neighbourhoods to improve all the initial allocations which reduced the walking distances. This improvement in the allocations was more significant for the *CA* than for *MCP0, MCP6* and *GA*. However, we found that changing the layout of the warehouse was also effective in reducing the walking distance if 1 or 2 aisles were introduced across the corridors such that evenly spaced sections were formed. This allows pickers to move from one corridor to the other without having to go to either end of the corridor entirely. Further, *VNS* was able to improve the initial allocations in these alternative layouts too. However, future considerations could involve exploring alternative layouts where additional corridors are added to reduce the length of each column. This was not investigated in the study due to time constraints and the need to redesign the *LSH*.

There are advantages and disadvantages to each of the construction heuristics considered, but if Tartan Trade finds the computational costs associated with *GA* to be suitable enough, this heuristic yields the highest quality of allocations amongst all allocations seen in this study. Otherwise, *MCP0* and *MCP6* yield allocations which outperform the current allocation system.

Future work could involve the study of alternative swaps to be considered for creating the set of neighbourhood structures in the *VNS* algorithm. Alternatively, considering other local search heuristics or path-finding algorithms to *VNS* and *NN*, respectively, may also prove beneficial in finding a better allocation system that can reduce the total walking distance or improve the picking routes so that Tartan Trade can increase the efficiency of operations. There is more scope to consider alternative warehouse layouts, which were beyond the scope of this study in the provided timeframe.

References

[1] Louise Golland. "Karl Menger and Taxicab Geometry". In: *Mathematics Magazine* 63.5 (Dec. 1990), pp. 326–327. DOI: https://doi.org/10.1080/0025570x.1990.11977548.

[2] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: https://doi.org/10.1109/tssc.1968.300136. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4082128.

Appendix

A Integer Linear Programming Problem

The problem is an integer programming problem in nature. Given a set of products $\mathcal{P}$, their location in a warehouse (i.e., on shelves), and a set of orders that need to be fulfilled, this problem describes the allocation of these products in the warehouse to minimise the distance a packer needs to walk to collect an order and return to the packing area at the warehouse's door.

Parameters

The given data provides information about the products, the orders and the shelves within the warehouse.

- $\mathcal{P}$: The set of product groups;
- $\mathcal{S}$: The set of shelves;
- $\mathcal{S}_0$: The set of shelves and the door;
- w_{ij} : The distance between shelves i and j, for $i, j \in \mathcal{S}$;
- $\mathcal{K}$: The set of of orders;
- D_k : The set of products required for order $k \in K$;
- V_k : The set of shelves in order $k \in K$;
- U_k : The vector of the route for picking order $k \in K$. If the route has shelf $i \to j$, then $u_{ik} < u_{jk}$ where i, j are points in the route;

Decision Variables

These variables represent the decisions that need to be taken to get an optimal solution.

- $x_{s,p}$ (binary) : $\begin{cases} 1, & \text{if a product } p \text{ is placed on shelf } s \in S, \\ 0, & \text{otherwise.} \end{cases}$
- $y_{i,j,k}$ (binary) : $\begin{cases} 1, & \text{if the packer travels from shelf } s_i \text{ to shelf } s_j \text{ for order } k, \\ 0, & \text{otherwise.} \end{cases}$

Constraints

The WAP considers the following constraints, which must be satisfied in each order.

- Each shelf can contain at most one product type:

$$\sum_{p \in \mathcal{P}} x_{s,p} \leq 1 \qquad \forall s \in \mathcal{S}.$$

- Each product must be allocated to at least one shelf and can be allocated to at most two shelves:

$$1 \leq \sum_{s \in \mathcal{S}} x_{s,p} \leq 2 \qquad \forall p \in \mathcal{P}.$$

- Each route must begin at the door (Shelf 0):

$$\sum_{j \in S} y_{0,j,k} = 1 \qquad \forall k \in \mathcal{K}.$$

- Each route must end at the door (Shelf 0):

$$\sum_{i \in S} y_{i,0,k} = 1 \qquad \forall k \in \mathcal{K}.$$

- For an order, $k \in K$, each route must only visit the shelves which contain products requested in the order:

$$\sum_{i \in S_0} y_{i,s,k} = \sum_{p \in V_k} x_{sp} \qquad \forall s \in \mathcal{S}, k \in \mathcal{K}.$$

- Each shelf should not be visited more than once per order:

$$\sum_{i \in S} y_{i,j,k} \leq 1 \qquad \forall\, j \in S, k \in \mathcal{K},$$

$$\sum_{j \in S} y_{i,j,k} \leq 1 \qquad \forall\, i \in S, k \in \mathcal{K}.$$

- A route for an order cannot contain an arc from shelf i to shelf i:

$$y_{iik} = 0 \qquad \forall\, i \in S_0,\ k \in \mathcal{K}$$

- No products can be placed at the door

$$\sum_{p \in \mathcal{P}} x_{0p} = 0$$

- The subtour elimination constraint:

$$u_{i_k} - u_{j_k} + 1 \leq (|V_k| - 1)(1 - y_{ijk}) \qquad i \neq j \in \mathcal{S},\ 1 \leq u_{i_k}, u_{j_k} \leq |V_k|\ k \in \mathcal{K},$$

$$1 \leq u_{i_k} \leq |V_k| \qquad i \in \mathcal{S},\ 1 \leq u_{i_k} \leq |V_k|\ k \in \mathcal{K},$$

- Binary and nonnegative constraints for decision variables:

$$\begin{aligned} y_{ijk} &\in \{0,1\} \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}, \\ x_{sp} &\in \{0,1\} \qquad \forall s \in S_0, p \in \mathcal{P}. \end{aligned}$$

Objective Function

The objective function includes the distance travelled by the packer to collect each order starting and ending at the door for each order. Thus, the objective in this problem is to minimize total distance travelled by Tartan Trade and is thus given by

$$\min \quad \sum_{k \in \mathcal{K}} \sum_{i \in S} \sum_{\substack{j \in S \\ i \neq j}} w_{ij} y_{ijk}.$$