1 Model Description

1.1 Genetic Algorithm

The standalone genetic algorithm heuristic utilized for this problem contains custom features intended to optimize the likelihood of finding good solutions at every iteration, with an additional diversity constraint to provide ergodicity. We will structure the description of the algorithm around its main parameters: Initialization method, Representation of solution, Parent selection, Crossover, Repair and Mutation operators, Replacement criteria and Stopping conditions.

Initialization method: The initial population of the genetic algorithm was built from randomly generated solutions and a population size of 40. The motivation for choosing a random initialization over using solutions generated from another heuristic or another initial solution method is to have a diverse initial population that prevents premature convergence and allows for most, if not all, of the solution space to be searched without relying on mutations.

Representation of solution: Each member of the population is represented by a “double strand of DNA”, where one array of length n is the traditional solution representation and holds the hub to which each node is allocated, and another array of length p holds the hubs in the same solution. In terms of information, the latter is redundant as it can be obtained by building a set of the values in the former, but as this array occupies little space in the memory and is used for crossovers, the information is stored for computational efficiency.

Parent selection: Each iteration generates 3 new couples, the members of which are selected through a tournament mechanism for each parent, with a total of 6 tournaments per generation. First, the tournament size is determined as a random number from 1 to 3, to allow, with a lower probability, the inclusion of very poor solutions as parents. Competitors are then selected from the population without repetition and the fittest member of each tournament is selected as parent.

Crossover Operator: The crossover operator acts on a two-stage process, first selecting

the offspring's hubs with the hub array and then performing a simple 1-point crossover on the solution array. Let the set of the parent's hubs be A, B, the set of the offspring's hubs be H, and C a set made from $p - N(A \cap B)$ random elements of the set $(A \cup B) - (A \cap B)$. The new hubs are generated semi-randomly such that $H = (A \cap B) UC$, that is, the set initially contains the intersection of the parent's hubs and is completed to size p with other random hubs from the parents. This is inspired in the genetic mechanism of recessive and dominant alleles and done to ensure that hubs are selected randomly unless they are part of both parents, in which case they will surely be passed on. The reasoning behind this is that, if the parents are likely good solutions since they won their tournaments and they have common hubs, then these are likely good hubs as well, therefore, it makes sense to conserve this trait. The 1-point crossover simply selects a random point to split the parents' solution arrays and forms two offspring by crossing and recombining them.

Repair Operator: If a solution generated contains nodes allocated to hubs that are not part of the new solution, that solution is unfeasible and the repair operator is called to reassign that node to either itself, if the node is a hub in the new solution, or to a random hub, if the node is a spoke in the new solution.

Mutation Operator: The mutation operator may be applied either in the spokes, where it reassigns a spoke to another hub, or in the hubs, where it swaps a random hub with a random spoke and reassigns spokes linked to the old hub to the new one. Both occur with 1% chance.

Replacement criteria: A track of the *gene pool* in the population is kept and updated at every iteration, with the number of times each node is a hub in the solutions. Any nodes that only appear as a hub once across all solutions marks the solution that has it as a hub as "endangered species". The offspring then replaces the worst solutions one at a time, updating the gene count accordingly in the process. This serves as a constraint on the population's diversity that guarantees a minimum level of differentiation in the solutions and increase ergodicity to reduce the algorithm's risk of converging and stopping at local optimums.

Stopping criteria: Two stopping criteria are used to terminate the algorithm: maximum number of generations, and 50 generations with no improvement in the best solution. Additionally, if the optimal cost is known, finding the optimal value is also included in the stopping criteria.

1.2 Tabu Search + Genetic Algorithm + Variable Neighborhood Descent

1.2.1 Overview of mathematical properties of the model

In order to develop a more efficient algorithm tailored to the hub selection problem, its mathematical properties were studied. For a given subset of the solution space, we succeeded in establishing a lower bound and determining criteria that, if not satisfied, guarantees the subset is dominated. Finally, an algorithm that capitalizes on these properties was built to ultimately reduce the search space and increase speed.

One way to visualize the full space of solutions is comparing different solutions. Two solutions are equal if and only if they have the same set of hubs and the same spoke allocation. There are $\binom{n}{p}$ possible combinations for different sets of hubs, and $(n-p)^p$ different ways to allocate the spokes for a single set of hubs. The full problem then has $\binom{n}{p}(n-p)^p$ possible solutions that can be viewed as $\binom{n}{p}$ different sets of hubs, each of them with $(n-p)^p$ different allocations for the spokes. The cost of a solution s is given by (x), where $h(x)$ is a function that maps each spoke to the hub it is allocated to (and hubs to themselves):

$$C(s)=\sum_{i}^{n}\sum_{j}^{n} W_{ij}\left(\chi\, c_{ih(i)}+\alpha c_{h(i)h(j)}+\delta c_{jh(j)}\right)$$

Let $E(i)=\sum_{j}^{n} W_{ij}$ be the sum of the flow leaving from i and $I(i)=\sum_{j}^{n} W_{ji}$ be the sum of the flow arriving to i. Then, assuming the cost matrix is symmetrical (simply for a shorter notation, it has not no impact in the calculations if it's not) we can reorganize the cost function as follows:

$$C(s)=\chi\sum_{i}^{n} c_{ih(i)}\left(\sum_{j}^{n} W_{ij}\right)+\delta\sum_{i}^{n} c_{ih(i)}\left(\sum_{j}^{n} W_{ji}\right)+\sum_{i}^{n}\sum_{j}^{n} W_{ij}\left(\alpha c_{h(i)h(j)}\right)$$

$$C(s)=\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]+\alpha\sum_{i}^{n}\sum_{j}^{n} W_{ij}c_{h(i)h(j)}$$

Where $\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]$ is the Spoke-Hub cost $SHC(s)$ and $\alpha\sum_{i}^{n}\sum_{j}^{n} W_{ij}c_{h(i)h(j)}$ is the Hub-Hub cost $HC(s)$. The latter is non-negative since both flow and cost are always non-negative (and in practical situations certainly greater than zero unless the networks are isolated, or the transportation cost is zero). Therefore, as $0<\alpha<1$, then:

$$\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]<C(s)<\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]+\sum_{i}^{n}\sum_{j}^{n} W_{ij}c_{h(i)h(j)}$$

$$SHC(s)<C(s)<\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]+\sum_{i}^{n}\sum_{j}^{n} W_{ij}c_{h(i)h(j)}$$

1.2.2 Lower bound for a subspace of the solution space

We can use this inequality to find a lower bound for the cost of a solution by setting $\alpha=0$, i.e.: calculating only $SHC(s)$. Still, this isn't very useful as it's only valid for a single solution so it's more convenient to calculate $C(s)$ instead of just $SHC(s)$. However, when we set $\alpha=0$ and determine the hubs are a set of p nodes H, the decision problem is reduced only to determining to which hub will each spoke be allocated, and the optimal solution is given by $s^{\iota}=min\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]\rightarrow h(i)=min\, c_{ih(i)}$ which is simply allocating each spoke to its cheapest hub. Note that if s^{ι} is the optimal solution of SHC for H and S_H the subset of all solutions that have H as hubs, $SHC(s^{\iota})\leq SHC(s)\,\forall\, s\in S_H$. Given that $SHC(s^{\iota})$ is a lower bound for $SHC(s)\,\forall\, s\in S_H$, and $SHC(s)$ is a lower bound for $C(s)$, we conclude $SHC(s^{\iota})\leq C(s)\,\forall\, s\in S_H$. An algorithm can then calculate $SHC(s^{\iota})$ for every subspace S_H it moves to and never search it if $SHC(s^{\iota})$ is greater than the cost of the incumbent solution. this property can greatly improve search speed, especially for lower values of α, where the gap between $C(s)$ and $SHC(s)$ is smaller.

1.2.3 Dominated solutions in spoke allocation

When we break the hub selection problem into two a two-part problem of first choosing hubs and then allocating spokes to them, the latter remains an NP-hard problem with $(n-p)^p$ solutions. At first one might think this is counterintuitive and expect the solution to be the same solution s^i for $C(s)$ when $\alpha=0$, but it is easy to see this is not the case as the complexity arises from the *tradeoff* between SHC and HC, that is, assigning a spoke to its closest hub may be sub-optimal because another allocation, while increasing the solution's SHC, might reduce the HC more and thus improve the solution overall. Still, motivated by the intuition that if a hub is sufficiently far from a spoke, then it must certainly not be a good allocation for that spoke, we try to verify if this intuition holds mathematically, and if so, determine the threshold for "sufficiently far". Figure 1 illustrates a small network with hubs A,B, C and spokes D,E,F,G,H,I and a spoke's allocation possibilities in dotted lines.

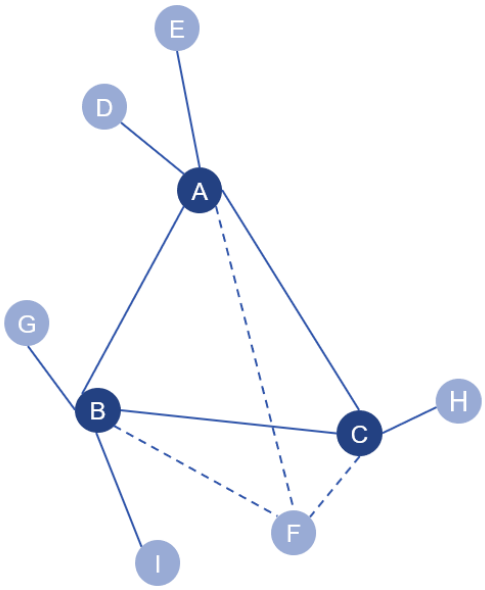

Figure 1: Illustration of a single allocation spoke-hub network

In the situation above, we highlight the decision on $h(F)$. This decision will impact the total cost of the network, specifically by changing the costs associated both to the flow starting from F and to the flow going to F. Looking at the cost function in the shape:

$$C(s)=\sum_{i}^{n} c_{ih(i)}\left[\chi E(i)+\delta I(i)\right]+\alpha\sum_{i}^{n}\sum_{j}^{n} W_{ij}c_{h(i)h(j)}$$

Let $Flow(x)=\chi E(x)+\delta I(x)$. Then for the purpose of analyzing the allocation of F, we are interested only in a few terms of these sums, namely:

$Flow(F)c_{Fh(F)}+\alpha\sum_{i}^{n}W_{iF}c_{h(i)h(F)}+\alpha\sum_{j}^{n}W_{Fj}c_{h(F)h(j)}$. Table 1 summarizes these costs per unit of flow to each hub depending on the allocation.

	Allocated to A	**Allocated to B**	**Allocated to C**
Cost to A	$c_{FA}+\alpha c_{AA}$	$c_{FB}+ac_{BA}$	$c_{FC}+ac_{CA}$
Cost to B	$c_{FA}+\alpha c_{AB}$	$c_{FB}+\alpha c_{BB}$	$c_{FC}+ac_{CB}$
Cost to C	$c_{FA}+ac_{AC}$	$c_{FB}+ac_{BC}$	$c_{FC}+\alpha c_{CC}$

Table 1:

Each column in the table is an array containing the unitary cost of getting to each of the hubs, made by a common term, the cost of moving to its assigned hub, plus a term referent to the cost of moving from that hub to the destination hub. This is also valid for when these hubs are the same, we need only define $c_{ii}=0\,\forall i$. In a comparison of the possible hubs for assignment, there are p candidates, each with a p -dimensional array of costs, and usually comparing two arrays gives no clear absolute winner: we can't compare a unified metric of the sum or average of the costs as the importance of each is determined by the flow in the final arrangement. If, however, a hub i is outperformed by another hub j in all p -dimensions for spoke k , then independently of the final weighted flow, $i\prec j$ for k . In a spoke, should the former be too high compared to other hubs, all terms in the array will be high as well, thus making it a poor assignment for that spoke, confirming our intuition that if a hub is far enough from a spoke, that allocation is strictly inferior to others.

Additionally, for a spoke x , if the cost of going to each of the hubs when x is assigned to hub y is higher than the equivalent cost when x is assigned to hub z , we can assert the latter is objectively superior than the former for that spoke, that is, for spoke x , hub y is strictly dominated by hub z . In a formal notation, for a spoke x with a set of hubs H , y is an inefficient allocation for x if the following is true:

$$\exists h\in H\vee c_{xh}+\alpha c_{hi}<c_{xy}+\alpha c_{yi}\,\forall i\in H$$

1.2.4 Algorithm

Aiming to best capitalize on these properties, a hybrid algorithm was developed with a combination of Tabu Search, Memetic Algorithm and Variable Neighborhood Descent. Given the additional computational effort of calculating a set of hubs' lower bound and especially each spoke's dominated hub allocations for that configuration, it made sense to structure the search space in a way the algorithm's movement is efficient in three

aspects: First, it compares the incumbent solution with a region's lower bound or another, more rigorous condition to avoid regions known or likely to be dominated. Second, it ranks candidate search regions from most to least promising and moves through them in this order. Third, it only generates non-dominated solutions for spoke allocations.

We will structure the description of the algorithm around its main components: Initialization, Tabu Search in the Hub Neighborhood Structure, selection of candidate solutions for search in the spoke subspace, Memetic Algorithm for spoke allocation, VND for spoke allocation, and Stopping conditions.

Initialization: A warm start method is used to obtain the initial solution fed into the Tabu Search. This is done by generating 100 (300 if 55 nodes) random solutions with the xxxxx method and using the one with the lowest cost. There was a noticeable improvement in the algorithm's time to using this method compared to using only a random solution. A larger number of solutions is generated in the runs with 55 nodes to make up for the increased search space, and the additional time to generate them is usually justified by the time saved for having a better initial solution in the algorithm.

Tabu Search in the Hub Neighborhood Structure: The Tabu Search uses a Neighborhood Structure that allows for any spoke to be swapped with any hub and reallocates all spokes to their lowest-cost hub. Alone, this structure allows for a total of $\binom{n}{p}$ solutions with $p(n-p)$ neighbors to any given point. A percentage (xx%) of these neighbors is then randomly generated as candidates for both movement in the Hub Neighborhood Structure and in-depth search with VND/MA. All generated neighbors are stored in a list with information on their solution, cost, lower bound ($SHC(s^{¿})¿$ and whether they have been previously analyzed.

Two Tabu Lists vetoes the candidates: one, of length $p-2$, holds the nodes that recently entered the solution's set of hubs H and vetoes candidates in which H don't include its members, while the other, of length $\frac{n}{5}$, holds the nodes that recently left H and vetoes candidates in which H include its members. In other words, they intend to prevent both premature exit and reentrance in H, respectively. The

aspiration condition for both is a candidate whose cost is better than the current incumbent, and the Tabu Search will also move to the best candidate to satisfy this condition (if any). When no solution satisfies the aspiration condition, the algorithm's movement is determined by a Move Rule function that can select a candidate randomly, or the candidate with the lowest cost.

At the start of the search, the candidates in a neighborhood are randomly generated with equal probability, but these odds are updated after every $(n-p)$ iterations as the search progresses. Starting from an equal baseline, the nodes that are hubs for the current incumbent solution become 50% more likely to be included in H than the average node, and after ranking nodes by the % of solutions in which they feature that are dominated, all of them have their odds changed by up to $+¿-50$, according to their rank.

A reset mechanism is also implemented in case the algorithm stays too long on a region with no improvement. If the incumbent hasn't been updated for 20 iterations, the Tabu Lists are cleared, and the algorithm is moved randomly to one of the best 3 solutions found so far. Before resuming the Tabu Search, this solution is analyzed with the MA even if it was investigated before, in case the optimal solution is in that region but was missed previously. The rationale for this is that, if you haven't yet found the optimal (or simply a better) solution, this can be either because the region where that solution is hasn't been searched yet, or because that region has already been searched but this point was missed. The longer you search without any improvement, the more likely is the latter compared to the former, so it might pay off to search a promising region one more time.

Selection of candidate solutions for search in the spoke subspace: The candidates generated in the Tabu Search are evaluated by different criteria to define whether the node will be expanded for optimization of the spoke subspace, in which order they will be evaluated, and by which algorithm (VND or MA). Inspired on a Best-first search algorithm, for every candidate s, a heuristic evaluation function $f(s)$ is applied to evaluate how promising is expanding that node. If $f(s)<C(s_i)$, where s_i is the current incumbent solution, then s is selected for expansion, and added to a priority queue ordered by $f(s)$. The choice of f has a lot of impact on the model, usually in the form of a *tradeoff* between optimality and execution time, as increasing f will

select less candidates for node expansion, which reduces the time spent exploring these regions but risks overlooking the optimal solution's region as well. For instance, the maximum value we found for f to guarantee the optimal solution's region will always pass the cutoff independently of the data is $f(s)=SHC(s)$. For high values of α, using this function will approve most of the candidates for node expansion, increasing the algorithm's runtime for each iteration severalfold. Different formats for $f(s)$ were tried, but ultimately the most successful of them was $f(s)=\frac{C(s)}{1.03}$, as the local optimums of most regions didn't tend to improve a lot over their respective $C(s)$.

After each iteration of the Tabu Search, if the priority queue isn't empty, i.e.: if some candidates were approved for analysis in the spoke subspace, the model decides for each of them whether they will go through the VND or MA for further optimization. A function determines the dominated spoke-hub allocations in the subspace S of all solutions with the same hubs as s, and from the result, the number of non-dominated solutions in the region $N(S)$ is calculated with $N(S)=\prod_{i}^{n} N(G_i)$ where G_i is the set of non-dominated feasible hub allocations of node i based on the hubs of solution s (and hubs can be seen as spokes that must be allocated to themselves). Since the VND algorithm is much faster than MA but the latter does well on larger combinatorial problems, the model uses $N(S)$ to direct the largest subspaces to the MA algorithm, and the remaining is searched with VND. The exact threshold is based on an array that holds $N(S)$ for the last $n\times p$, equal to the value of the (90/85/75) quantile for $\alpha=(0.2,0.4,0.8)$. Additionally, though a rare occurrence, there is a chance that for a solution v and its corresponding region V, $N(V)=1$, that is, there is only one non-dominated solution in V which is v itself, therefore v is the local optimum in V and no further search is necessary.

Memetic Algorithm for spoke allocations: This algorithm is based on the GA model, so to avoid repetition, we will focus on the significant differences compared to the original. These can be summarized in the Initialization method, Representation of solution, Crossover operator, Mutation operator, Offspring Local Search and Replacement criteria. The concept of *mutable nodes* is also defined to help crossover and mutation

operators: as both spokes with only one non-dominated hub allocation and hubs are the same across all feasible, non-dominated solutions, only the spokes with more than one non-dominated hub allocation are of interest in generating new solutions – these are referred as mutable nodes.

Initialization method: The initial population of the memetic algorithm is composed by the candidate solution s and 39 randomly generated non-dominated solutions in S to a population size of 40.

Representation of solution: As we're interested only in generating solutions in S, the entire population has the same set of hubs H, therefore reducing to a single array of length n in the solution representation. This can be further reduced to an array containing only mutable nodes, but this change would only provide an imperceptible computational advantage.

Crossover operator: This performs a 1-point crossover along the solution whose cut is generated randomly between the mutable nodes. This is made to prevent the creation of clones of the parents, which would be common especially whenever sequences of immutable nodes appear at the start or end of the array.

Mutation operator: Mutations occur only in mutable nodes, with 1% chance, and when a spoke is mutated it is reassigned randomly to another non-dominated hub for that spoke.

Offspring Local Search: Each of the offspring goes through a Classical Local Search with Steepest Descent in the Neighborhood Structure "type c" after any possible mutation and before joining the rest of the population. A modified cost function was developed specifically for this Neighborhood Structure, which, by calculating the difference in cost of a step instead of the cost function, reduces the time for each iteration by an order of approximately n.

Replacement criteria: The "gene pool" concept is modified to keep track of the amount of times each hub appears in the mutable nodes, and whenever a hub only appears once in a mutable node, the solution that contains that specific assignment is not eligible for replacement. The offspring then replaces the worst solutions in an identical manner to

the Genetic Algorithm.