Appendix A: Genetic Algorithm Implementation

In this section, we describe the implementation of the genetic algorithm used in this paper. We start by defining the initial population generation method. Next, we explain how the crossover and mutation operators are constructed. Finally, we elaborate on the population update procedure and how short and long simulations are used for fitness evaluation. For clarity, throughout this section, we reuse the exemplary chromosome representation in Figure 16.

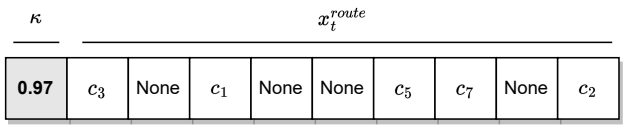

Figure 16 Chromosome representation

A.1. Initial Population Generation

The population generation process aims to create a diverse set of individuals (i.e., solutions). In this paper, we experimentally set the population size to 200 individuals. Each individual in the initial population is constructed as follows.

First, the target job-fill-rate κ (first gene in the chromosome) is randomly selected from the set $\{0.80, 0.81, 0.82, \ldots, 0.99\}$. Next, an ordered list of integers from 1 to 9 is generated, as we have nine pending requests in this example. This list is then shuffled to create a random permutation of the original list. Afterward, for each element in the list, we randomly change its value to *None* by generating two random numbers between 0 and 1 independently. If the first random number is smaller than the second, then the element's value is replaced by *None*. Finally, the two parts are combined to create a complete chromosome. Figure 16 shows one possible realization of this process. After the population is constructed, individuals are assigned fitness values as in Algorithm 1.

A.2. Crossover Operators

At the beginning of each genetic search iteration, two individuals from the population are selected as parents for crossover. In this paper, parents are selected using the tournament selection procedure (Miller, Goldberg et al. 1995). Each parent is chosen by randomly selecting 50 individuals from the population. The individual with the best fitness value among the 50 selected individuals is then designated as one of the parents. Next, the parents undergo the crossover process, as illustrated in Figure 17.

Figure 17 (A) shows the two selected parents. In the first step (Figure 17 (B)), two random crossover points are selected along the length of the chromosome. Next, the two gene segments between these two crossover points are swapped. In this second step (Figure 17 (C)), the first element of each chromosome is replaced by the average value from both, effectively crossing over the target job-fill-rate from the parents. Finally, in the last step (Figure 17 (D)), duplicated genes in each newly generated chromosome are replaced by iteratively selecting genes not present in the current chromosome from the other chromosome, from left to

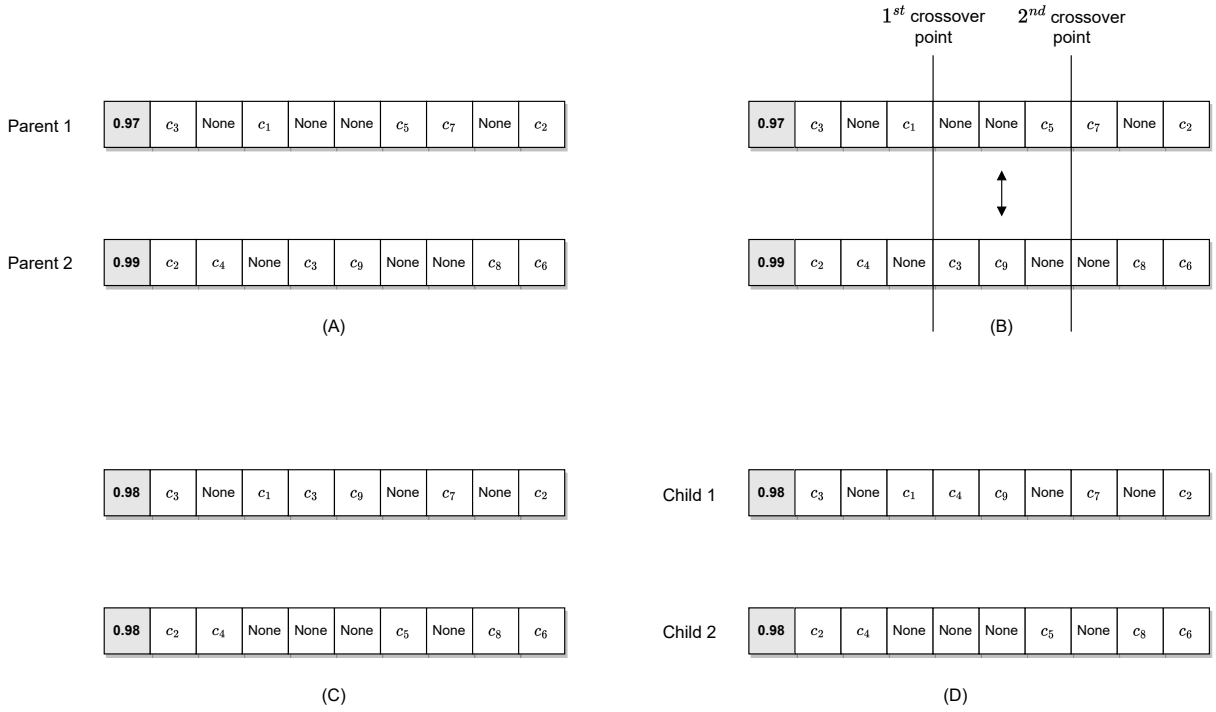

Figure 17 Crossover operator

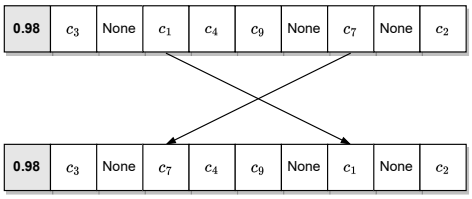

Figure 18 Swap mutation

right. If no such gene can be found for replacement, then the *None* value is inserted instead. In this example, the fifth element from the first chromosome switches from 3 to 4. At the end of the process, we have two newly created chromosomes, *child 1* and *child 2*.

A.3. Mutation Operators

Mutations are randomly applied to each newly generated child to maintain diversity in the population and prevent the search from prematurely converging to a local optimum. We use the following six mutation operators. The application order of the mutation operators is shuffled in each iteration. The default probability of the application of each operator is 0.2 and increases to 0.6 if the total fitness value of the top 10 solutions does not change in the last 50 iterations (hereafter referred to as nonimproving iterations). The only exception is the cheapest insertion mutation, where we experimentally set the mutation rate to a constant value of 0.8.

- **Swap mutation.** The swap mutation is illustrated in Figure 18, where two random genes from the second part of the chromosome swap values.

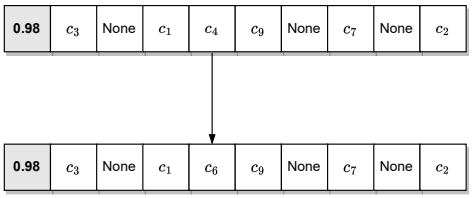

Figure 19 Exchange mutation

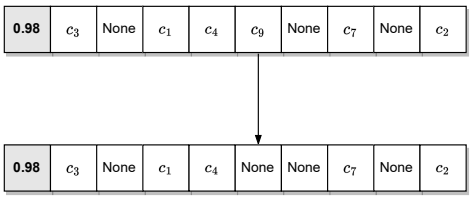

Figure 20 Omission mutation

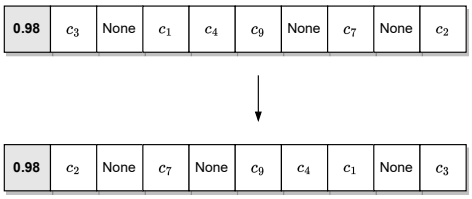

Figure 21 Reverse mutation

• **Exchange mutation.** In the exchange mutation, a single gene from the second part (which is not *None*) is randomly selected, and its value is mutated to one of the values not on the chromosome. For example, the fifth gene changes its value from 4 to 6 in Figure 19.

• **Omission mutation.** This mutation randomly selects a gene (which is not *None*) from the second part and mutates its value to *None*, essentially omitting a customer from the route. An example is shown in Figure 20.

• **Reverse mutation.** This mutation reverses the order of the genes in the second part of the chromosome, as shown in Figure 21.

• **Cheapest insertion mutation.** The cheapest insertion mutation removes all genes that are not *None* from the chromosome and re-inserts them one by one such that the increase in travel distance is the smallest.

• **Job-fill-rate mutation.** This mutation slightly alters the value of the first part of the chromosome by adding a random quantity from the set $\{-0.02, -0.01, 0.01, 0.02\}$ to the first gene. The resulting value is bounded to ensure that it ranges from 0.80 to 0.99.

A.4. Population Update and Stopping Conditions

After the crossover and mutation procedures are completed, the newly created chromosomes are added to the population. Next, the population is updated by removing the two individuals with the worst fitness value, maintaining a population of a fixed size.

The whole genetic search procedure stops when the search reaches 300 nonimproving iterations. The best 50 feasible solutions then undergo a more extensive simulation, where the number of sampling iterations

increases from 1 (during the search) to 100. Finally, the individual with the best fitness value is chosen as the best solution.

Appendix B: Mixed-integer programming formulation for obtaining results of π^{CLA}

Consider a graph $G=(V,E)$, where $V=v_0,v_1,\ldots,v_K$. Vertex v_0 denotes the depot, while the rest of the vertices denote the set of K customer locations. E represents the set of edges between these locations in V. Let $\varphi \in \mathbb{R}$ represent the penalty cost for each late period associated with a repair request. Define c_i^{arrival} as the arrival date for a repair request at customer i, c_i^{deadline} as the deadline for the repair request at customer i, d_{ij} as the cost to traverse edge (i,j), and t_{ij} as the time required to travel on edge (i,j). Furthermore, let s denote the constant service time for each request, and let $T^{\max}$ be the maximum working time for a technician in a single day.

Decision variables:

- x_{ijt}: equals 1 if the technician goes from customer i to customer j in period t, and 0 otherwise.
- y_{it}: equals 1 if the technician visits customer i in period t, and 0 otherwise.
- u_{it}: auxiliary variable to eliminate subtours.
- w_{it}: equals 1 if a repair request at customer i is completed by period t, and 0 otherwise.

$$\textbf{Minimize} \quad \sum_{t=1}^{T}\sum_{(i,j)\in E} d_{ij}x_{ijt} + \varphi \sum_{t=1}^{T}\sum_{i\in V\setminus v_0}(1-w_{it})\mathbb{1}_{t\geq c_i^{\text{deadline}}}$$

Subject to:

Flow conservation constraints:

$$\sum_{i\in V, i\neq j} x_{ijt} - \sum_{i\in V, i\neq j} x_{jit} = 0 \quad \forall j\in V, t\in 1,\ldots,T$$

Constraints to guarantee that a customer can only be visited in period t if there is an incoming edge during the same period:

$$y_{it} \leq \sum_{j\in V, i\neq j} x_{jit} \quad \forall i\in V, t\in 1,\ldots,T$$

Constraints to ensure that each customer is visited at most once:

$$\sum_{t\in T} y_{it} \leq 1 \quad \forall i\in V\setminus v_0$$

Subtour elimination constraints:

$$u_{it} - u_{jt} + |V|x_{ijt} \leq |V| - 1 \quad \forall (i,j)\in E, i\neq j, t\in 1,\ldots,T$$

Constraints to ensure a customer cannot be visited before the repair request arrives:

$$y_{it} \leq \mathbb{1}_{t\geq c_i^{\text{arrival}}} \quad \forall i\in V\setminus v_0, t\in 1,\ldots,T$$

Constraints to ensure the total travel time and service time of the technician in a day cannot be larger than a constant T^{max}:

$$\sum_{(i,j)\in E} t_{ij}x_{ijt} + s\sum_{i\in V\setminus v_0} y_{it} \leq T^{\max} \quad \forall t \in 1,\ldots,T$$

Constraints to ensure that a repair request at customer i is marked as completed at period t if the technician has visited i in any period $t' \leq t$:

$$w_{it} \leq \sum_{t'\leq t} y_{it'} \quad \forall i \in V \setminus v_0, t \in 1,\ldots,T$$

Domain constraints:

$$y_{it}, x_{ijt}, w_{it} \in 0,1 \quad \forall i \in V, (i,j) \in E, t \in 1,\ldots,T$$

$$u_{it} \in \mathbb{R} \quad \forall i \in V, t \in 1,\ldots,T$$

Appendix C: Implementation of SPSA in π^{CFA}

Simultaneous perturbation stochastic approximation (Spall et al. 1992) is an optimization method that is well-suited to optimizing systems with a large number of parameters or where each evaluation of the system is expensive or noisy. It is a type of stochastic approximation method that uses only two function measurements per iteration. The pseudo-code for the SPSA based on the implementation in Spall (1998) is given in Algorithm 2. Below is a simplified description of how the algorithm works:

- At the beginning of each iteration, we introduce a "perturbation" to the present approximation of θ^{CFA} in both directions. The extent of this perturbation is influenced by the current iteration number and the hyperparameters A, α^{CFA}, and c.
- Next, the performance of the two newly produced parameters is evaluated by rolling out the π^{CFA} using these estimates.
- Then, the numerical gradient $\hat{g}$ is calculated using the gathered performance value, which is the average daily total cost.
- Finally, we use gradient estimate $\hat{g}$ to update the current value of θ^{CFA} in the direction of the gradient.

The hyperparameters used in the SPSA follow the guidelines set out in Spall (1998), with some additional adjustments to ensure satisfactory convergence. The five hyperparameters (α^{CFA}, γ^{CFA}, A, a, and c), as detailed in Spall (1998), have been assigned values of 0.602, 0.101, 100, 0.75, and 2, respectively. On the basis of experimental trials, the roll-out horizon has been fixed at 20 days, and the training is stopped after 10,000 iterations. While it is possible to reduce the training time of π^{CFA} by shortening the roll-out horizon H (we also tried $H = 10$), this might lead to poor convergence due to the high noise level during gradient estimation. These settings have demonstrated satisfactory convergence for the parameter π^{CFA}.

Algorithm 2 Simultaneous Perturbation Stochastic Approximation - π^{CFA}

Initialization:
$\theta^{CFA} \leftarrow 0 \quad \alpha^{CFA} \leftarrow 0.602 \quad \gamma^{CFA} \leftarrow 0.101 \quad A \leftarrow 100 \quad a \leftarrow 0.75 \quad c \leftarrow 2$
for $k \leftarrow 1, 2, \ldots$ number of iterations **do**
  Generate θ^+ and θ^-:
    $a_k \leftarrow a/(k+A)^{\alpha^{CFA}}$
    $c_k \leftarrow c/k^{\gamma^{CFA}}$
    $\Delta \leftarrow random.choice(-1, 1)$
    $\theta^+ \leftarrow \theta^{CFA} + c_k \times \Delta$
    $\theta^- \leftarrow \theta^{CFA} - c_k \times \Delta$
  Estimate gradient
    $y^+ \leftarrow evaluate(\theta^+)$
    $y^- \leftarrow evaluate(\theta^-)$
    $\hat{g} \leftarrow (y^+ - y^-)/(2 \times c_k \times \Delta)$
  Update estimation
    $\theta^{CFA} \leftarrow \theta^{CFA} - a_k \times \hat{g}$
end for

Appendix D: Additional Tables and Figures

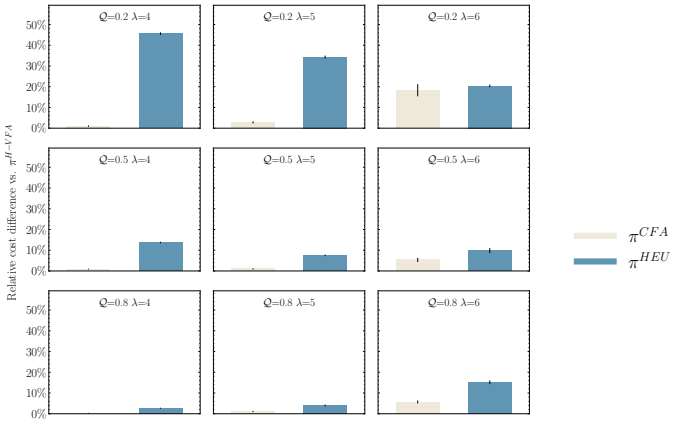

Figure 22 **Relative performance π^{H-VFA} - π^{CFA} - π^{HEU} — 2 Clusters**

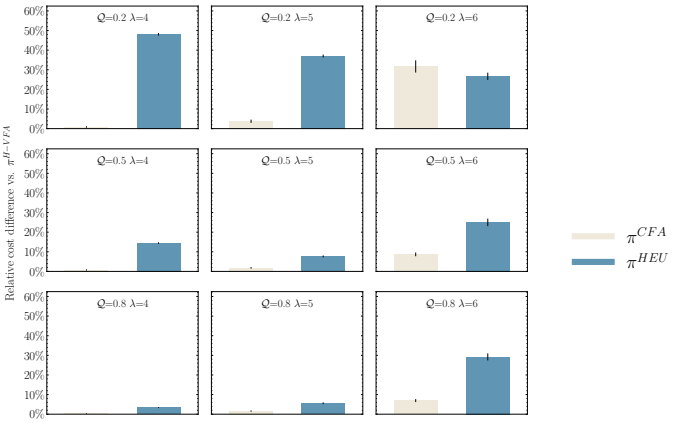

Figure 23 **Relative performance π^{H-VFA} - π^{CFA} - π^{HEU} — 3 Clusters**

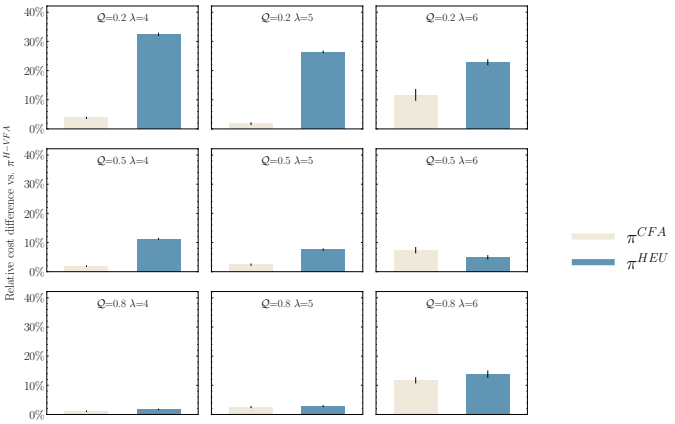

Figure 24 **Relative performance π^{H-VFA} - π^{CFA} - π^{HEU} — Center**

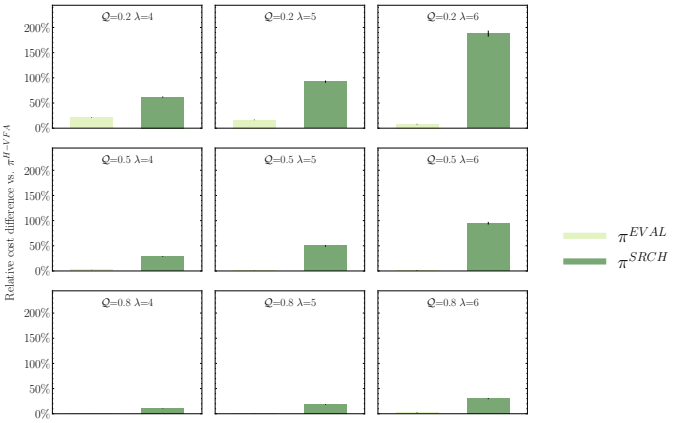

Figure 25 **Relative performance π^{H-VFA} - π^{EVAL} - π^{SRCH} — 2 Clusters**

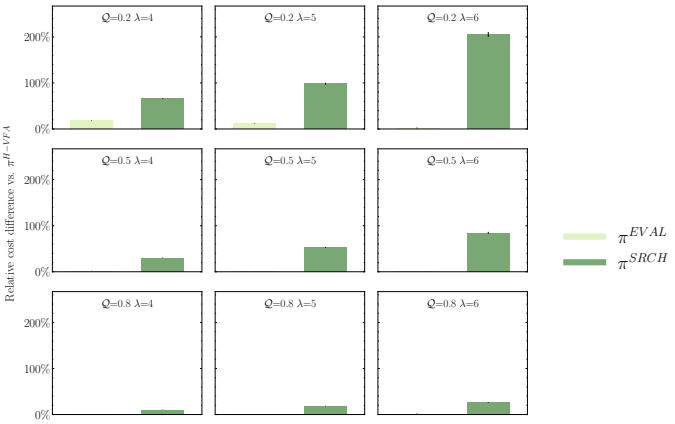

Figure 26 **Relative performance π^{H-VFA} - π^{EVAL} - π^{SRCH} — 3 Clusters**

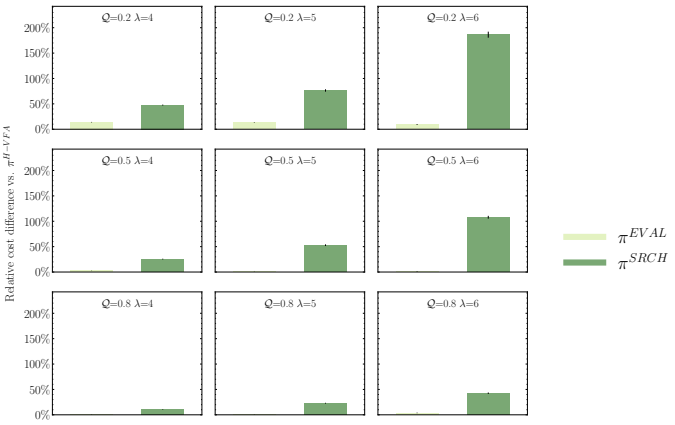

Figure 27 **Relative performance π^{H-VFA} - π^{EVAL} - π^{SRCH} — Center**

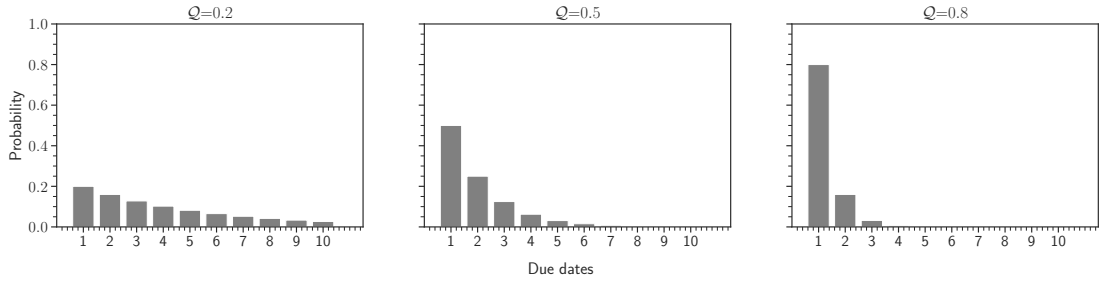

Figure 28 **Probability mass function of due dates**

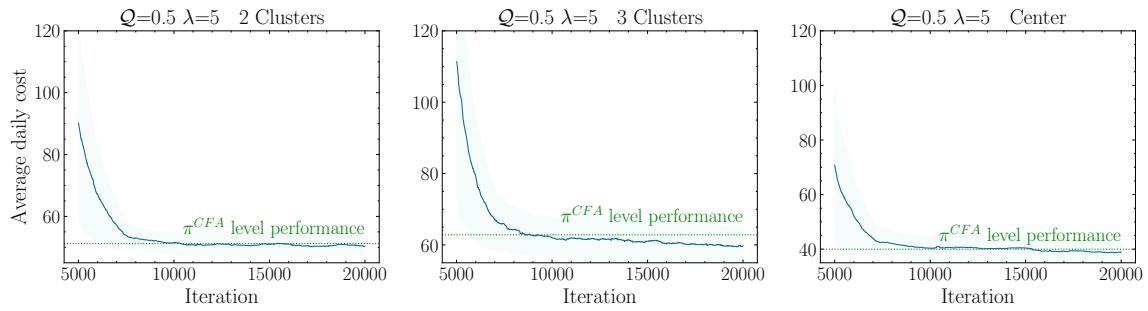

Figure 29 **Cost evolution during training - π^{H-VFA} (5000-point moving average across four exemplary runs)**

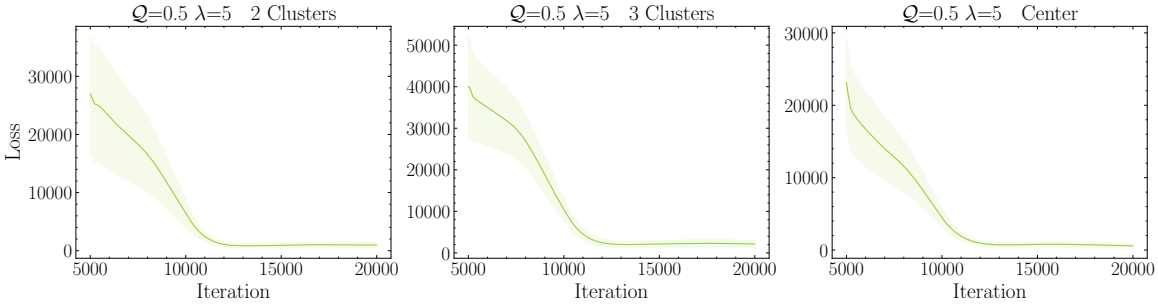

Figure 30 **Loss evolution during training - π^{H-VFA} (5000-point moving average across four exemplary runs)**

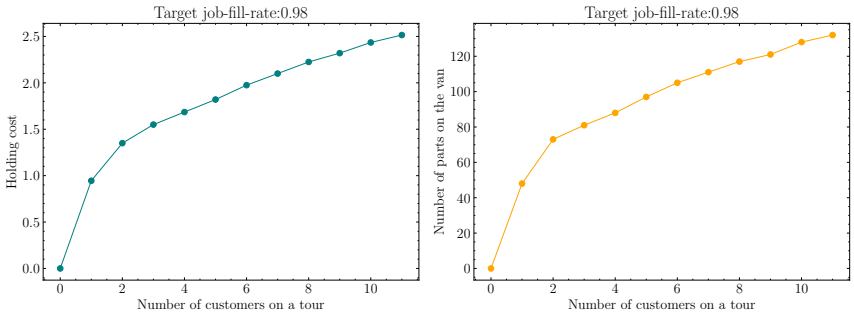

Figure 31 **The concave relation between the holding costs, number of spare parts stocked in the van, and the number of customers on a tour.**

Notations and symbols	
$\mathcal{C}_t$	the set of pending requests at the end of day t.
c_i	a request.
c_i^{loc}	the location of request c_i.
c_i^{due}	the number of days until the due date of request c_i.
$c_i^{visited}$	the visiting status of request c_i.
$\mathcal{C}_t^{failed}$	the set of failed requests on route x_t^{route}.
$\mathcal{C}_t^{success}$	the set of successfully completed requests on route ξ_t.
$C(S_t, x_t)$	the expected immediate cost incurred by being in state S_t and making decision x_t.
$\mathcal{D}$	the depot.
D_{t+1}	the random variable representing spare parts required by customers in the route plan x_t^{route} whose $c_i^{visited} = 0$.
d_{t+1}	a realization of D_{t+1}.
e_{mn}	an element of matrix D_{t+1} to indicate if spare part n is required by customer m.
M_t	the number of customers on route plan x_t^{route} whose $c_i^{visited} = 0$.
N	the number of spare part types.
n_{jt}	the number of spare part of types j at the end of day t before decision x_t has been made.
n_{jt}^x	the number of spare parts of type j at the end of day t after decision x_t has been made.
p_j	the probability that part type j is needed for a repair task.
$\mathbb{P}(S_{t+1} = S' \mid S_t, x_t)$	the probability of transitioning to pre-decision state S' given pre-decision state S_t and the decision x_t.
$\mathcal{R}_{t+1}^{new}$	the set of newly arrived requests during day $t+1$.
r_{t+1}^{new}	a realization of $\mathcal{R}_{t+1}^{new}$.
S_t	the pre-decision state variable at the end of day t.
S_t^x	the post-decision state variable at the end of day t.
$\mathcal{S}$	the set of all possible pre-decision states.
$\mathcal{S}^x$	the set of all possible post-decision states.
$S^M(S_t, x_t)$	the transition function from pre-decision state S_t to post-decision state S_t^x.
$S^{Mx}(S_t^x, W_{t+1})$	the transition function from post-decision state S_t^x to pre-decision state S_{t+1}.
T^{max}	the maximum daily working duration of the technician.
$V(S_t)$	the discounted value of the state S_t.
W_{t+1}	the exogenous information variable on day $t+1$.
x_t	the decision made at the end of day t.
x_t^{route}	the route plan to be executed on day $t+1$.
x_t^{parts}	the list of parts to be added to the repair kit at the end of day t.
$\mathcal{X}_t$	the set of all feasible decisions at the end of day t.
x_{jt}^{parts}	the number of units of spare part type j to be added to the repair kit at the end of day t.
ξ_t	the repair kit at the end of day t.
ξ_t^x	the repair kit at the end of day t after decision x_t has been made.
ω_{t+1}	a realization of W_{t+1}.
$\Upsilon_t^{travel}(x_t^{route})$	the travel cost incurred from route plan x_t^{route}.
$\Upsilon_t^{det-delay}(\mathcal{C}_t, x_t^{route})$	the deterministic service delay cost of executing route plan x_t^{route}.
$\Upsilon_t^{holding}(\xi_t^x)$	the holding cost incurred from updated repair kit ξ_t^x.
$\Upsilon_t^{failed}(S_t^x, W_{t+1})$	the expected fail-to-fix cost incurred from decision x_t.
$\Upsilon_t^{exo-delay}(S_t^x, W_{t+1})$	the expected service delay cost incurred from decision x_t.
π	a policy.
π^*	the optimal policy.
Π	the set of all policies.
ψ_{it}^x	the expected job-fill-rate at customer c_i, given decision x_t.
γ	the discount factor.

Table 2 List of notations and symbols

Appendix E: List of Modeling Notations

The list of notations and symbols used in the modeling section is summarized in Table 2.

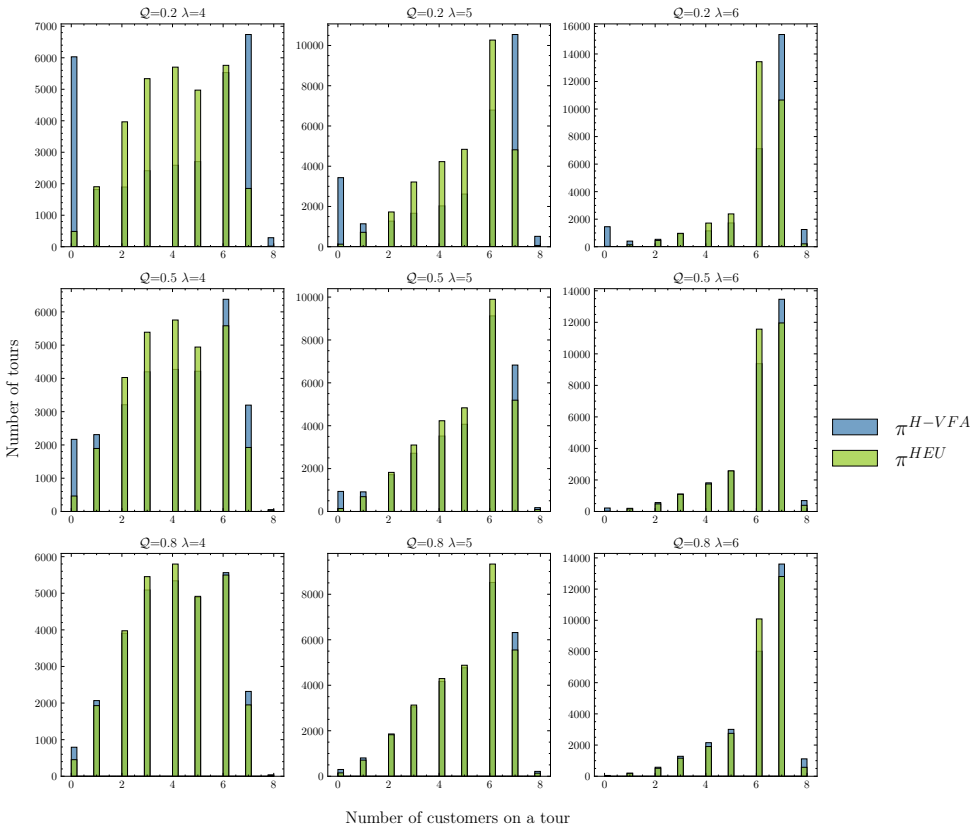

Figure 32 Tour lengths

Appendix F: Analysis of the Level of Consolidation in Relation to the Repair Kit

In the course of our experiment, we observed that the assortment of spare parts in the service vehicle remains unchanged despite an increase in $\mathcal{Q}$. This may initially seem counterintuitive, as the instinctive expectation would be to decrease the inventory of parts in a low-urgency scenario to lower the holding costs, especially considering that there is more leeway for return visits before the deadline in the event of failed repairs. However, as depicted in Figure 32, it is advantageous to retain the same repair kit while favoring increased consolidation in low-urgency situations, evidenced by the larger number of tours with a high customer count under π^{H-VFA}. In the most extreme scenario, characterized by high flexibility and a low arrival rate (that is, $\mathcal{Q} = 0.2$ and $\lambda = 4$), π^{H-VFA} results in a substantial number of days where the technician was not dispatched, yet there exist days where the tour includes seven or even eight requests. This strategy enables most service requests to be resolved on the first visit while keeping the holding costs reasonably low, further capitalizing on the concave relation mentioned earlier due to the high number of customers on each tour. An added advantage of having days with no dispatches is that the technician can use this downtime for other productive activities, such as furthering their training, maintaining equipment, or attending to administrative responsibilities. Contrasted with a strategy such as π^{HEU}, which overlooks consolidation, the additional benefits are the infrequent need to modify the composition of the repair kit and the reduction in the number of times the service vehicle has to be restocked.

Appendix G: Performance Comparison with Only Travel and Holding Costs

It is essential to note that the loss-of-goodwill costs in this study, specifically the *service delay* and *fail-to-fix* costs, serve as reasonable estimates to encapsulate customer inconvenience due to prolonged waiting times or the necessity of a second technician visit. In the absence of these two costs, two of the three strategies outlined in this study, namely, π^{CFA} and π^{H-VFA}, would effectively cease to function. The reason is that under an infinite horizon setting with no obligation to serve all customers, the most cost-efficient strategy would be to avoid customer visits entirely, thereby maintaining zero inventory in the van and incurring zero travel and holding costs. The strategy π^{HEU} remains "functional" as it greedily inserts customers into the tour regardless of these costs.

To further explore the robustness of our proposed method under slight changes in the problem setting, we conducted additional experiments without considering these loss-of-goodwill costs. This experimental setting assumes that customers whose deadlines have expired are attended to by an "emergency" technician making a direct round trip from the depot, with the associated travel costs added to the total travel costs for the day. Such direct routes are typically more expensive due to the principle of the triangle inequality.

Upon retraining both π^{H-VFA} and π^{CFA} under this updated cost setting and optimizing the κ parameter for π^{HEU}, we reassessed their performance using the same experimental set, excluding loss-of-goodwill cost calculations. The results are shown in Figure 33.

The results indicate that our primary approach π^{H-VFA} remains the top-performing strategy, underscoring the adaptability of our learning methodology to minor variations in settings. However, we observed that the performance gap between π^{H-VFA} and the other benchmark policies has somewhat narrowed (compared with the results in Figure 9). A closer analysis shows that the modified setting prevents the additive effect of a large number of customers waiting for an extended period by immediately deploying an emergency technician to handle overdue requests, effectively removing them from the system once their deadlines have passed. Consequently, the benefit of proactive planning in π^{H-VFA} for circumventing such situations using anticipation is less pronounced compared to the original setting.

While this modification offers insights into the algorithm's performance, we argue that the loss-of-goodwill costs are indeed pertinent in real-world scenarios. Companies risk either gradually losing customers due to inadequate service or facing financial obligations to compensate customers due to missed deadlines. Moreover, emergency technician deployments generally incur more than mere travel costs, somewhat resembling the penalty terms initially included in our cost considerations.

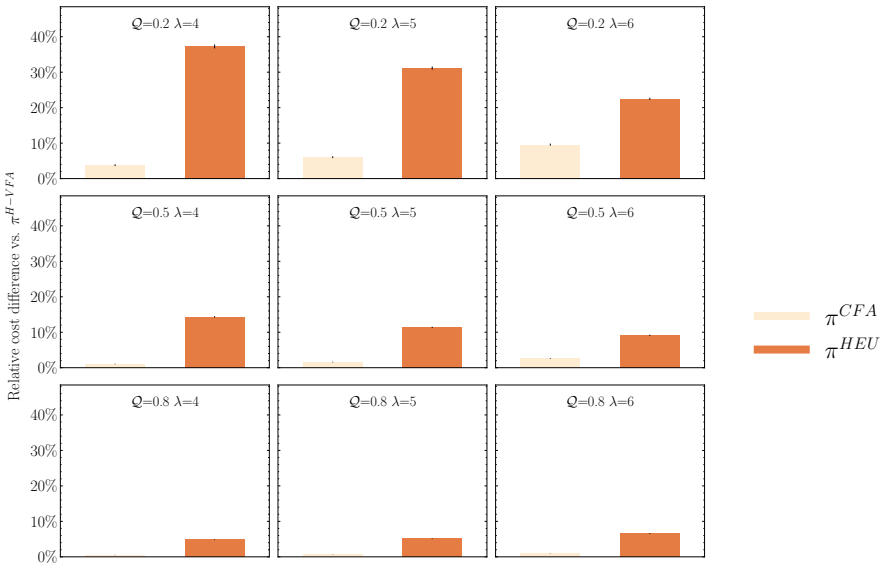

Figure 33 Relative performance π^{H-VFA} - π^{CFA} - π^{HEU} — without loss-of-goodwill costs

References

Miller BL, Goldberg DE, et al., 1995 *Genetic algorithms, tournament selection, and the effects of noise. Complex Cystems* 9(3):193–212.

Spall JC, 1998 *Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Transactions on Aerospace and Electronic Systems* 34(3):817–823.

Spall JC, et al., 1992 *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control* 37(3):332–341.