Stochastic dynamic job scheduling with interruptible setup and processing times: An approach based on queueing control

Dongnuan Tian
Department of Management Science, Lancaster University Management School, Lancaster University, Lancaster LA1 4YW, United Kingdom. Email: d.tian2@lancaster.ac.uk.

Rob Shone*
Department of Management Science, Lancaster University Management School, Lancaster University, Lancaster LA1 4YW, United Kingdom. Email: r.shone@lancaster.ac.uk.

Online Appendices

A Proof of Theorem 2.2.

Proof. Consider a modified MDP in which the state space is

$$\tilde{S} := \{(v, w, (x_1, ..., x_d)) \mid v \in V,\ w \in D,\ x_i \geq 0 \text{ for } i \in D\}, \tag{A.1}$$

where the extra variable w represents the most recent demand point at which the server witnessed no jobs present. More precisely, if $w = i$ then this indicates that at a certain time step t_0 in the history of the process the system was in a state with $v = i$ and $x_i = 0$, and none of the states visited in the more recent time steps (between t_0 and the present time) had the server at another demand point $j \in D \setminus \{i\}$ at which there were no jobs present. (We can set w to an arbitrary value when the process is initialized.) All other aspects of the modified MDP formulation (e.g. actions and costs) are the same as in the original version. The transitions of the process do not lose their memoryless property when w is included, since the knowledge that $w = i$ at a particular time step is sufficient to specify the probability distribution for its value at the next step; specifically, w is guaranteed to remain unchanged unless either of the following two cases applies:

1. The server is at a demand point $j \in D \setminus \{i\}$ with $x_j = 1$ and chooses action j, in which case there is a probability of μ_j that we have $w = j$ at the next time step and a probability of $1 - \mu_j$ that we still have $w = i$.

2. The server is at an intermediate stage $k \in N$ adjacent to a demand point $j \in D \setminus \{i\}$ with

*Corresponding author

$x_j = 0$ and chooses action j, in which case there is a probability of τ that we have $w = j$ at the next time step and a probability of $1 - \tau$ that we still have $w = i$.

Consider a 'polling system' policy $\theta^{[P]}$ under which the server visits the demand points in a repeating sequence $(1, 2, ..., d, 1, 2, ..., d, 1, 2, ...)$ and, upon arriving at any demand point i, remains there until the number of jobs has been reduced to zero ($x_i = 0$) before moving to the next demand point in the sequence. If we already have $x_i = 0$ when the server arrives at node i, then the server immediately moves to the next demand point. It should be noted that, since switches require an exponentially distributed amount of time, it may happen that the server decides to move away from demand point i when $x_i = 0$ but a new job arrives at i while the server is still located at i. In this case, under our proposed policy, the server continues trying to move to the next demand point rather than processing the new job. We assume that the server always chooses the shortest path (in terms of the number of intermediate stages that must be traversed) between two demand points and, in the case where two or more paths are tied for the shortest length, the path is selected according to some fixed priority ordering of the nodes. This policy can be represented as a stationary policy in our modified MDP by specifying actions according to the simple rule that if $w = i \leq d - 1$ then the server attempts to move towards demand point $i + 1$ by taking the next step on the shortest path to that node; or, if it is already at $i + 1$, then it remains there. Similarly if $w = d$ then the server attempts to move to demand point 1 (or remains there).

Under the proposed policy $\theta^{[P]}$, the system behaves as a polling system with an exhaustive polling regime (meaning that demand points are served until they are empty). Given that $\rho < 1$, Lemma 3.1 in Altman et al. (1992) implies that the system is stable and there exists a probability distribution $\{\pi_{\theta^{[P]}}(\mathbf{x})\}_{\mathbf{x} \in \tilde{S}}$ such that $\pi_{\theta^{[P]}}(\mathbf{x})$ is the long-run proportion of time spent in state $\mathbf{x} \in \tilde{S}$. In the next part of the proof we show that it is possible to use value iteration to compute an optimal policy for the modified MDP. Given that the state space S is infinite, this requires the use of the 'approximating sequences' method developed in Sennott (1999). Let Ψ denote the modified MDP with state space $\tilde{S}$ defined in (A.1) and let $(\Psi_0, \Psi_1, \Psi_2, ...)$ denote a sequence of MDPs that are defined on finite spaces, so that the MDP Ψ_m (for $m \in \mathbb{N}_0$) has state space $\tilde{S}_m$ given by

$$\tilde{S}_m := \{(v, w, (x_1, ..., x_d)) \mid v \in V,\ w \in D,\ 0 \leq x_i \leq m \text{ for } i \in D\}.$$

Thus, in the MDP Ψ_m, we do not allow the number of jobs at any node $i \in D$ to be greater than m. We do this by modifying the transition probabilities so that if $\mathbf{x}$ is a state with $x_i = m$ for some $i \in D$, then the arrival of a new job at node i is impossible and instead the 'self-transition' probability $p_{\mathbf{x},\mathbf{x}}(a)$ is increased by λ_i. Let θ_m^* be an optimal policy for the MDP Ψ_m. We can show that the sequence $(\theta_0^*, \theta_1^*, ...)$ converges to an optimal policy for the infinite-state MDP Ψ, but this requires certain conditions to be verified. Specifically, we must show that the assumptions (AC1)-(AC4) described on p. 169 of Sennott (1999) hold for the sequence $(\Psi_0, \Psi_1, \Psi_2, ...)$. (See also Sennott (1997), pp. 117-118 for an equivalent set of assumptions.)

Assumption (AC1) in Sennott (1999) states that, for the finite-state MDP Ψ_m, there exists a constant g_m and a function $h_m : \tilde{S}_m \to \mathbb{R}$ satisfying

$$g_m + h_m(\mathbf{x}) = f(\mathbf{x}) + \min_{a \in A_{\mathbf{x}}} \left\{ \sum_{\mathbf{y} \in \tilde{S}} p_{\mathbf{x},\mathbf{y}}(a) h_m(\mathbf{y}) \right\} \qquad \forall \mathbf{x} \in \tilde{S}_m. \tag{A.2}$$

We can easily show that, given any two states $\mathbf{x}, \mathbf{y} \in \tilde{S}_m$, there exists a stationary policy θ such that $\mathbf{y}$ is accessible from $\mathbf{x}$ in the Markov chain induced by θ. This can be achieved by considering arbitrary states $\mathbf{x}, \mathbf{y} \in \tilde{S}_m$ and identifying a sequence of random transitions that would cause the system to transition from $\mathbf{x}$ to $\mathbf{y}$ under policy θ (which can be specified differently for each pair of states $(\mathbf{x}, \mathbf{y})$). It then follows from Puterman (1994) (p. 478) that Ψ_m belongs to the 'communicating' class of multichain MDP models, and the constant g_m in (A.2) is the optimal long-run average cost for Ψ_m. Moreover, the values $h_m(\mathbf{x})$ for states $\mathbf{x} \in \tilde{S}_m$ can be computed using the well-known method of value iteration, in which we define $v_m^{(k)}(\mathbf{x})$ as the optimal (minimal) total cost in a finite-horizon problem with k stages initialized in state $\mathbf{x}$ and then compute $h_m(\mathbf{x}) = \lim_{k \to \infty}(v_m^{(k)}(\mathbf{x}) - v_m^{(k)}(\mathbf{z}))$, with the reference state $\mathbf{z} \in \tilde{S}_m$ chosen arbitrarily.

In order to verify assumptions (AC2)-(AC4) we will need to use several properties of the functions $v_m^{(k)}(\mathbf{x})$. Let $\mathbf{x}^{i+}$ denote a state identical to $\mathbf{x}$ except that one extra job is present at demand point $i \in D$. The required properties are:

1. $v_m^{(k)}(\mathbf{x}) \leq v_m^{(k+1)}(\mathbf{x}) \quad \forall\, m, k \in \mathbb{N}_0,\ \mathbf{x} \in \tilde{S}_m.$ (A.3)
2. $v_m^{(k)}(\mathbf{x}) \leq v_m^{(k)}(\mathbf{x}^{i+}) \quad \forall\, m, k \in \mathbb{N}_0,\ i \in D,\ \mathbf{x} \in \tilde{S}_m$ such that $x_i < m.$ (A.4)
3. $v_m^{(k)}(\mathbf{x}) \leq v_{m+1}^{(k)}(\mathbf{x}) \quad \forall\, m, k \in \mathbb{N}_0,\ \mathbf{x} \in \tilde{S}_m.$ (A.5)
4. Fix $m \in \mathbb{N}_0$ and let $\mathbf{x} = (v, w, (x_1, ..., x_d))$ and $\mathbf{x}' = (v', w', (x'_1, ..., x'_d))$ be two states in $\tilde{S}_m$ with $v = v'$ and $x_i = x'_i$ for $i = 1, ..., d,$ but $w \neq w'$. Then $v_m^{(k)}(\mathbf{x}) = v_m^{(k)}(\mathbf{x}')$ for $k \in \mathbb{N}_0$. (A.6)

All of the properties above are logical and can be proved using induction on k. We have omitted details of the induction arguments in order to avoid making this proof excessively long, but they are quite straightforward and only require some care in considering the different possible actions that might be chosen by the relevant finite-horizon optimal policies. Property (A.3) states that the optimal expected total cost is increasing with the number of stages remaining, k. Property (A.4) states that this cost is increasing with the number of jobs initially present at any demand point. Property (A.5) states that this cost is increasing with the maximum number of jobs, m, allowed to be present at any demand point. Finally, property (A.6) states that the variable w has no effect on the optimal expected total cost, which makes sense as it does not impose any constraints on the actions that may be chosen in the k remaining stages.

We proceed to verify (AC2)-(AC4). Assumption (AC2) states that $\limsup_{m \to \infty} h_m(\mathbf{x}) < \infty$ for each $\mathbf{x} \in \tilde{S}$. Consider the polling-type policy $\theta^{[P]}$ described earlier. It is clear that the Markov chain induced by $\theta^{[P]}$ has a unichain structure on the state space $\tilde{S}$, since the state $\mathbf{z} = (1, 1, (0, 0, ..., 0))$ is accessible from any other state under this policy (this can be seen from the fact that, in between consecutive visits to demand point 1, it is always possible for no new jobs to arrive at any demand points). Let $J_{\theta^{[P]}}(\mathbf{x},\mathbf{z})$ denote the expected total cost incurred until the system enters state $\mathbf{z}$, given that it is initialized in state $\mathbf{x}$ and follows policy $\theta^{[P]}$. Since $\mathbf{z}$ is positive recurrent, it follows from standard theory (see Sennott (1999), pp. 298-302) that $J_{\theta^{[P]}}(\mathbf{x},\mathbf{z}) < \infty$ for all $\mathbf{x} \in \tilde{S}$. Next, for $m \in \mathbb{N}_0$, let $J_{m,\theta^{[P]}}(\mathbf{x},\mathbf{z})$ be defined in an analogous way to $J_{\theta^{[P]}}(\mathbf{x},\mathbf{z})$ except that we consider applying the policy $\theta^{[P]}$ to the finite-state MDP Ψ_m instead of the infinite-state MDP Ψ. It can easily be shown that $J_{m,\theta^{[P]}}(\mathbf{x},\mathbf{z}) < J_{\theta^{[P]}}(\mathbf{x},\mathbf{z})$ for all $m \in \mathbb{N}$, since the amount of time that the server spends at any demand node (and, hence, the cost incurred) is stochastically smaller under Ψ_m than under Ψ. We now follow similar arguments to those in the proof of Proposition 8.2.1, p. 171 in Sennott (1999). By using property (A.3) and the fact that $v_m^{(k)}(\mathbf{x})$ is defined as the expected k-stage cost under an optimal policy, we have

$$v_m^{(k)}(\mathbf{x}) \le v_m^{(p)}(\mathbf{x}) \le v_{m,\theta}^{(p)}(\mathbf{x}) \quad \forall \mathbf{x} \in \tilde{S},\ m \in \mathbb{N}_0,\ p \ge k, \tag{A.7}$$

where $v_{m,\theta}^{(k)}$ is the expected k-stage cost under an arbitrary policy θ. Let θ be defined to mimic policy $\theta^{[P]}$ until the system reaches state $\mathbf{z}$ and then follow an optimal finite-horizon policy for k steps. Then, from (A.7) it follows that $v_m^{(k)}(\mathbf{x}) \le J_{m,\theta^{[P]}}(\mathbf{x},\mathbf{z}) + v_m^{(k)}(\mathbf{z})$. Hence, using the previous arguments, we have

$$v_m^{(k)}(\mathbf{x}) - v_m^{(k)}(\mathbf{z}) \le J_{m,\theta^{[P]}}(\mathbf{x},\mathbf{z}) \le J_{\theta^{[P]}}(\mathbf{x},\mathbf{z}) \quad \forall m \in \mathbb{N}_0,\ \mathbf{x} \in \tilde{S}.$$

Since $h_m(\mathbf{x}) = \lim_{k\to\infty}(v_m^{(k)}(\mathbf{x}) - v_m^{(k)}(\mathbf{z}))$, this establishes that the sequence $\{h_m(\mathbf{x})\}_{m\in\mathbb{N}}$ is bounded above and hence $\limsup_{m\to\infty} h_m(\mathbf{x}) < \infty$ as required.

Assumption (AC3) states that there exists a constant $Q \ge 0$ such that $-Q \le \liminf_{m\to\infty} h_m(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$. By using property (A.4) and taking limits as $k \to \infty$, we obtain

$$h_m(\mathbf{x}) \le h_m(\mathbf{x}^{i+}) \quad \forall i \in D,\ \mathbf{x} \in \tilde{S}_m \text{ such that } x_i < m. \tag{A.8}$$

This shows that the function h_m attains a minimum on the subset of states with no jobs present, which we denote as U. That is,

$$\arg\min_{\mathbf{x}\in\tilde{S}_m} h_m(\mathbf{x}) \in \{(v, w, (x_1, ..., x_d)) \mid v \in V,\ w \in D,\ x_i = 0 \text{ for all } i \in D\} =: U.$$

Let $\mathbf{u}^*$ be a state that attains the minimum above. We will assume that $\mathbf{u}^*$ is positive recurrent under $\theta^{[P]}$. However, this requires some justification. Recall that the server visits the demand points according to the sequence $(1, 2, ..., d, 1, 2, ..., d, 1, 2, ...)$ under policy $\theta^{[P]}$. Therefore, at any given time, the server's current node v must lie somewhere on the path between demand points w and $w+1$ (if $w \le d-1$) or d and 1 (if $w = d$). Using property (A.6), we can assume that the variables v and w associated with state $\mathbf{u}^*$ do indeed satisfy these constraints, as it is always possible to change the value of w without making any difference to the optimal finite-stage expected cost. Also, if $\mathbf{u}^*$ is a state with $v \notin D$ (i.e. the server is at an intermediate stage rather than a demand point), it is reasonable to assume that node v is visited during the server's cyclic route under policy $\theta^{[P]}$, since if this is not the case, we can always modify the policy $\theta^{[P]}$ slightly so that node v is visited at some point during the server's route.

Using similar arguments to those given for (AC2), we then have that $\mathbf{u}^*$ is positive recurrent under $\theta^{[P]}$ and $J_{m,\theta^{[P]}}(\mathbf{x},\mathbf{u}^*) \leq J_{\theta^{[P]}}(\mathbf{x},\mathbf{u}^*) < \infty$ for any $\mathbf{x} \in \tilde{S}_m$. In particular let us consider the state $\mathbf{z} = (1, 1, (0, 0, ..., 0))$. Repeating previous arguments, we have $v_m^{(k)}(\mathbf{z}) \leq J_{\theta^{[P]}}(\mathbf{z},\mathbf{u}^*) + v_m^{(k)}(\mathbf{u}^*)$. Taking limits as $k \to \infty$, we obtain $h_m(\mathbf{u}) \geq h_m(\mathbf{u}^*) \geq -J_{\theta^{[P]}}(\mathbf{z},\mathbf{u}^*)$ for all $\mathbf{u} \in U$. This establishes the required lower bound for states in U, and it follows from (A.8) that the same lower bound also works for all $\mathbf{x} \in \tilde{S}_m$. Since this argument can be repeated for each $m \in \mathbb{N}$ (establishing a uniform lower bound independent of m), we have $-J_{\theta^{[P]}}(\mathbf{z},\mathbf{u}^*) \leq \liminf_{m\to\infty} h_m(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$ as required.

Assumption (AC4) states that $\limsup_{m\to\infty} g_m =: g^* < \infty$ and $g^* \leq g(\mathbf{x})$ for $\mathbf{x} \in \tilde{S}$, where g_m is the constant that appears in (A.2) and the notation $g(\mathbf{x})$ allows for a possible dependence of the long-run average cost on the initial state $\mathbf{x}$. By Proposition 8.2.1, Step 3(i) in Sennott (1999), it is sufficient to show that

$$v_m^{(k)}(\mathbf{x}) \leq \lim_{p\to\infty} v_p^{(k)}(\mathbf{x}) \quad \forall\, m, k \in \mathbb{N}_0,\ \mathbf{x} \in \tilde{S}_m, \tag{A.9}$$

which follows immediately from property (A.5).

Having verified that assumptions (AC1)-(AC4) hold for the modified MDP Ψ, we can use the results in Sennott (1999) to conclude that any limit point of a sequence of stationary optimal policies for the finite-state MDPs $(\Psi_0, \Psi_1, \Psi_2, ...)$ is optimal for Ψ (Theorem 8.1.1) and furthermore a limit point is guaranteed to exist (Proposition B.5). By the previous arguments, we can compute an optimal policy for any finite-state MDP Ψ_m using value iteration. During the process of value iteration, the functions $v_m^{(k)}(\mathbf{x})$ for $\mathbf{x} \in \tilde{S}_m$ are computed using the rule

$$v_m^{(k+1)}(\mathbf{x}) = f(\mathbf{x}) + \min_{a\in A_\mathbf{x}} \left\{ \sum_{\mathbf{y}\in\tilde{S}_m} p_{\mathbf{x},\mathbf{y}}(a) v_m^{(k)}(\mathbf{y}) \right\}, \quad k \in \mathbb{N}_0.$$

Property (A.6) implies that if $\mathbf{x}$ and $\mathbf{x}'$ are two states in $\tilde{S}_m$ that differ from each other only in the variable w, then any action $a \in A_\mathbf{x}$ that attains the minimum in the equation above for state $\mathbf{x}$ is also a feasible action that attains the minimum in the corresponding equation for state $\mathbf{x}'$. Essentially, this means that it is possible to find an optimal stationary policy for Ψ_m that chooses actions independently of the variable w. By the previous arguments, the same property also applies to an optimal stationary policy for the infinite-state MDP Ψ (obtained as a limit of the optimal finite-state policies). However, if we have an optimal stationary policy that chooses actions independently of w, then the same policy must also be admissible for the MDP formulated in Section 2 with state space S. From (AC4) we also know that the long-run average cost under such a policy is finite, implying stability. This completes the proof. □

B DVO Heuristic

In this appendix we describe the steps of the DVO heuristic as presented in Duenyas and Van Oyen (1996). Note that, where appropriate, we adapt the authors' notation so that it is consistent with the notation used in our paper.

1. If the server has just finished processing a job at some demand point $i \in D$ then there are two possible cases: either (a) there are still some jobs remaining at i ($x_i > 0$) or (b) there are no jobs remaining at i ($x_i = 0$).

   (a) In the first case, the server can either continue processing jobs at i or switch to some other demand point $j \neq i$. We carry out the following steps:

      i. Initialize an empty set $\sigma = \emptyset$.

      ii. For each demand point j with $c_j\mu_j \geq c_i\mu_i$, calculate the reward rate ψ_j that would be earned by switching to node j, serving it exhaustively, then switching back to node i, using

      $$\psi_j = c_j\mu_j \frac{x_j + \lambda_j\delta(i,j)/\tau}{x_j + \mu_j\delta(i,j)/\tau + (\mu_j - \lambda_j)\delta(j,i)/\tau}.$$

      If $\psi_j \geq c_j\mu_j\rho + c_i\mu_i(1-\rho)$, then add j to the set σ.

      iii. If σ is non-empty, then switch to the demand point j with the highest index ψ_j (with ties broken arbitrarily). Otherwise, process one more job at node i.

   (b) In the second case, the server can either remain idle at i or switch to some other demand point $j \neq i$. We carry out the following steps:

      i. Initialize three empty sets: $\sigma_1 = \emptyset$, $\sigma_2 = \emptyset$ and $\sigma = \emptyset$.

      ii. For each demand point $j \neq i$, calculate the reward rate ϕ_j in a similar way to part (a) but without including the time taken to switch back from j to i, using

      $$\phi_j = c_j\mu_j \frac{x_j + \lambda_j\delta(i,j)/\tau}{x_j + \mu_j\delta(i,j)/\tau}.$$

      If $\phi_j > c_j\mu_j\rho$, then add j to σ_1; otherwise, add j to σ_2.

      iii. If σ_1 is non-empty, let $\sigma = \sigma_1$. Otherwise, let $\sigma = \sigma_2$.

      iv. Let j^* denote the demand point in σ with the highest reward rate ϕ_{j^*}, with ties broken arbitrarily. If $x_{j^*} > \lambda_{j^*}\delta(j^*, i)/\tau$, then switch to j^*. Otherwise, remain idle at i.

2. If the server has just arrived at a demand point then it immediately begins processing jobs there if there is at least one job waiting. This ensures that, after switching to a new demand point, it must process at least one job there before switching somewhere else. If there are no jobs waiting, then the rule for idling described in step 1(b) is used.

3. If the server is idle at a demand point and a new job arrives in the system, then the rule for idling described in step 1(b) is used.

As mentioned in Section 3, there is no need to specify the rule used by the DVO heuristic when the server is at an intermediate stage $i \in N$, since the server is required to continue moving towards a particular demand point (chosen at the previous decision epoch) in this case. Similarly, if the server is at a non-empty demand point $i \in D$ and has *not* just finished processing a job, then this indicates that a service is in progress and the server is required to remain at node i.

C Proof of Theorem 3.3.

Proof. Let $\mathbf{x} = (v, (x_1, ..., x_d))$ be the current state of the system, where $v \in N$. We will use T_{arr} to denote the random amount of time until the next job arrives in the system, and T_{switch} to denote the amount of time until the server reaches a demand point. According to the rules of the K-stop heuristic, a sequence s can only be added to the set σ if it satisfies the condition $\frac{\partial}{\partial t}\psi(\mathbf{x}, s, t)\big|_{t=0} \leq 0$. We begin by showing that there always exists at least one sequence that satisfies this condition, and therefore σ_2 (and, hence, σ) must be non-empty. Indeed, the set $\mathcal{S}$ is defined to include all sequences of length m, for each $1 \leq m \leq K$. Therefore it includes the sequences of length one, (j), for each $j \in D$. Let $s = (j)$, where $j \in D$ is arbitrary. Then, using (11), we have

$$\begin{aligned}\psi(\mathbf{x}, (j), t) &= c_j\mu_j \left\{ \frac{T_1(\mathbf{x}, (j), t)}{t + \delta(v, j)/\tau + T_1(\mathbf{x}, (j), t)} \right\} \\ &= c_j\mu_j \left\{ \frac{[x_j + \lambda_j(t + \delta(v, j)/\tau)]/(\mu_j - \lambda_j)}{t + \delta(v, j)/\tau + [x_j + \lambda_j(t + \delta(v, j)/\tau)]/(\mu_j - \lambda_j)} \right\} \\ &= c_j\mu_j \left\{ \frac{x_j + \lambda_j(t + \delta(v, j)/\tau)}{x_j + \mu_j(t + \delta(v, j)/\tau)} \right\}.\end{aligned}$$

After differentiating, we obtain

$$\frac{\partial}{\partial t}\psi(\mathbf{x}, (j), t) = -c_j\mu_j \left\{ \frac{x_j(\mu_j - \lambda_j)}{[x_j + \mu_j(t + \delta(v, j)/\tau)]^2} \right\}, \tag{C.1}$$

which equals zero if $x_j = 0$, and is negative otherwise. Hence, the sequence $s = (j)$ satisfies the condition $\frac{\partial}{\partial t}\psi(\mathbf{x}, s, t)\big|_{t=0} \leq 0$. We can therefore be sure that σ_2 is non-empty, but in order to proceed we must consider two possible subcases: either (a) the set σ_1 is non-empty and we choose the sequence in σ_1 with the highest value of $\psi(\mathbf{x}, s, 0)$, or (b) the set σ_1 is empty, but σ_2 is non-empty and we choose the sequence in σ_2 with the highest value of $\psi(\mathbf{x}, s, 0)$. In the remainder of this proof we consider these two subcases separately.

Subcase (a): σ_1 is non-empty

For convenience, we will use $\xi(\mathbf{x}, s, t)$ to denote the proportion of time spent processing jobs (as opposed to switching between nodes or idling) while following sequence s. That is:

$$\xi(\mathbf{x}, s, t) := \frac{\sum_{j=1}^{|s|} T_j(\mathbf{x}, s, t)}{t + \sum_{j=1}^{|s|} [\delta(s_{j-1}, s_j)/\tau + T_j(\mathbf{x}, s, t)]}, \quad t \geq 0. \tag{C.2}$$

Given that $T_j(\mathbf{x}, s, t) \equiv R_j(\mathbf{x}, s, t)/(c_{s_j}\mu_{s_j})$ for each $j = 1, ..., |s|$, we can use identical arguments to those in the proof of Lemma 3.2 to show that $\xi(\mathbf{x}, s, t)$ is a monotonic function of t. Observe that the condition $\psi(\mathbf{x},s,0) \geq \gamma(\mathbf{x},s,0)$ is equivalent to

$$\xi(\mathbf{x},s,0) \geq \rho. \tag{C.3}$$

Let s^* denote the sequence in σ_1 that maximizes $\psi(\mathbf{x},s,0)$. In this case, according to the rules of the heuristic, the server attempts to take one step along a shortest path to demand point s_1^*, where s_1^* is the first element of s^*. Let $(i_0, i_1, ..., i_k)$ denote a shortest path from v to s_1^*, where $k := \delta(v, s_1^*)$, $i_0 := v$ and $i_k := s_1^*$. Also let $\mathbf{x}_j$ denote a state identical to $\mathbf{x}$ except that the server is located at node i_j, for $j = 1, 2, ..., k$. It is useful to note that

$$\xi(\mathbf{x}_j, s^*, 0) = \xi(\mathbf{x}_k, s^*, (k-j)/\tau) \quad \forall j \in \{1, 2, ..., k-1\}. \tag{C.4}$$

This is because $\xi(\mathbf{x}_j, s^*, 0)$ represents the proportion of time spent processing jobs given that the server begins at node i_j and immediately begins traveling along a path of length $(k-j)$ in order to reach node s_1^*, while $\xi(\mathbf{x}_k, s^*, (k-j)/\tau)$ is the corresponding proportion given that the server begins at node $i_k = s_1^*$ but waits for $(k-j)/\tau$ time units before beginning to process jobs there. In terms of the proportion of time spent processing jobs while following sequence s^*, these two quantities are the same. In the case $|s^*| = 1$ (that is, $s^* = (j)$ for some $j \in D$) we infer from (C.1) that

$$\frac{\partial}{\partial t}\xi(\mathbf{x}, (j), t) = \frac{1}{c_j \mu_j}\frac{\partial}{\partial t}\psi(\mathbf{x}, (j), t) \leq 0, \tag{C.5}$$

and hence, using (C.4), we can be assured that the condition $\xi(\mathbf{x}, s^*, 0) \geq \rho$ (equivalent to $\psi(\mathbf{x}, s^*, 0) \geq \gamma(\mathbf{x}, s^*, 0)$) remains satisfied as the server moves towards s_1^*. Therefore, according to the rules of the heuristic, s^* remains included in σ_1 at all stages while the server moves from v to s_1^*. On the other hand, consider the case $|s^*| \geq 2$. In this case the rules of the heuristic imply that the extra condition $\xi(\mathbf{y}, s^*, 0) \geq \rho$ must be satisfied, where $\mathbf{y}$ is equivalent to $\mathbf{x}_k$ in the notation of this proof. By the same reasoning used to derive (C.4), we have

$$\xi(\mathbf{x}, s^*, 0) = \xi(\mathbf{x}_k, s^*, k/\tau). \tag{C.6}$$

Since $\xi(\mathbf{x}_k, s^*, t)$ is a monotonic function of t, we can infer from (C.4) and (C.6) that

$$\min\{\xi(\mathbf{x}, s^*, 0),\ \xi(\mathbf{x}_k, s^*, 0)\} \leq \xi(\mathbf{x}_j, s^*, 0) \leq \max\{\xi(\mathbf{x}, s^*, 0),\ \xi(\mathbf{x}_k, s^*, 0)\} \quad \forall j \in \{1, 2, ..., k\}. \tag{C.7}$$

Given that $s^* \in \sigma_1$, the rules of the heuristic imply that $\xi(\mathbf{x}, s^*, 0) \geq \rho$ and $\xi(\mathbf{x}_k, s^*, 0) \geq \rho$, so from (C.7) it follows that $\xi(\mathbf{x}_j, s^*, 0) \geq \rho$ for each $j = 1, 2, ..., k$. Also, given that $s^* \in \sigma_1$, the heuristic rules imply $\frac{\partial}{\partial t}\psi(\mathbf{x}, s^*, t)\big|_{t=0} \leq 0$. Applying the same reasoning to the function ψ that we used for ξ, we have

$$\psi(\mathbf{x}_j, s^*, 0) = \psi(\mathbf{x}_k, s^*, (k-j)/\tau) \quad \forall j \in \{1, 2, ..., k\}, \tag{C.8}$$

$$\psi(\mathbf{x}, s^*, 0) = \psi(\mathbf{x}_k, s^*, k/\tau). \tag{C.9}$$

From the proof of Lemma 3.2 we know that $\frac{\partial}{\partial t}\psi(\mathbf{x}, s^*, t)$ has the same sign for all $t \geq 0$. Hence, from (C.9) it follows that $\frac{\partial}{\partial t}\psi(\mathbf{x}_k, s^*, t)\big|_{t=0} \leq 0$ and then from (C.8) it follows that

$\frac{\partial}{\partial t}\psi(\mathbf{x}_j, s^*, t)\big|_{t=0} \leq 0$ for each $j = 1, 2, ..., k$. By collating the arguments given thus far, we can say that as the server travels from v to s_1^* along the shortest path $(i_0, i_1, ..., i_k)$, the sequence s^* always satisfies the necessary conditions to be included in σ_1.

Now suppose that the server arrives at node i_1 before time T_{arr}. If $i_1 = s_1^*$ (i.e. the shortest path from v to s_1^* is of length 1), then there is nothing further to prove, as the server has completed its journey to demand point s_1^* via the shortest possible path. On the other hand, if $i_1 \neq s_1^*$ (i.e. the shortest path from v to s_1^* has length greater than 1), then we must continue. Given that $\frac{\partial}{\partial t}\psi(\mathbf{x}, s^*, t)\big|_{t=0} \leq 0$ and using Lemma 3.2 again, it must be the case that $\psi(\mathbf{x}, s^*, 0) \leq \psi(\mathbf{x}_1, s^*, 0)$; in other words, the average reward does not decrease after the server takes a step along the shortest path to s_1^*. By the previous arguments, s^* is still included in σ_1 when the server is located at i_1. However, we cannot be sure that s^* is chosen by the heuristic at state $\mathbf{x}_1$.

If s^* is chosen by the heuristic at $\mathbf{x}_1$, then the server continues attempting to move along the shortest path to node s_1^*. On the other hand, suppose the heuristic chooses an alternative sequence $\tilde{s} \in \sigma_1$ at $\mathbf{x}_1$, and let $\tilde{s}_1$ be the first demand point in sequence $\tilde{s}$. If $\tilde{s}_1 = s_1^*$, then (once again) we have no difficulties, as the server continues attempting to move along the shortest path to s_1^*. In the rest of this subcase we assume a non-trivial case where $\tilde{s}_1 \neq s_1^*$.

We can show that although $\tilde{s}_1 \neq s_1^*$, the server's movement from node v to i_1 still qualifies as a step along a shortest path from v to $\tilde{s}_1$. In other words, even if the server now prefers to move towards a different demand point, the step from v to i_1 was still a step in the right direction. To see this, first note that

$$\psi(\mathbf{x}, \tilde{s}, 0) \leq \psi(\mathbf{x}, s^*, 0) \leq \psi(\mathbf{x}_1, s^*, 0) \leq \psi(\mathbf{x}_1, \tilde{s}, 0), \tag{C.10}$$

where the first inequality is due to the fact that s^* is preferred to $\tilde{s}$ at state $\mathbf{x}$, the second inequality is due to the fact that $\frac{\partial}{\partial t}\psi(\mathbf{x}, s^*, t)\big|_{t=0} \leq 0$ (as stated earlier) and the third inequality is due to the fact that $\tilde{s}$ is preferred to s^* under the new state $\mathbf{x}_1$. The rules of the heuristic state that if the average rewards for two sequences are equal, then a sequence is chosen according to a fixed priority ordering of the demand points in D. Therefore either the first inequality or the third inequality in (C.10) must be strict, as if they both hold with equality then the same sequence (either s^* or $\tilde{s}$) must be chosen at both $\mathbf{x}$ and $\mathbf{x}_1$. We conclude that

$$\psi(\mathbf{x}, \tilde{s}, 0) < \psi(\mathbf{x}_1, \tilde{s}, 0), \tag{C.11}$$

implying that the average reward for sequence $\tilde{s}$ has increased after moving from v to i_1. Let $\tilde{\mathbf{x}}$ denote a state identical to $\mathbf{x}$ except that the server is located at node $\tilde{s}_1$ instead of v. By definition, we have

$$\psi(\mathbf{x}, \tilde{s}, 0) = \psi(\tilde{\mathbf{x}}, \tilde{s}, \delta(v, \tilde{s}_1)/\tau), \tag{C.12}$$

$$\psi(\mathbf{x}_1, \tilde{s}, 0) = \psi(\tilde{\mathbf{x}}, \tilde{s}, \delta(i_1, \tilde{s}_1)/\tau). \tag{C.13}$$

Suppose (for a contradiction) that $\delta(i_1, \tilde{s}_1) \geq \delta(v, \tilde{s}_1)$. If these two distances are equal then from (C.12)-(C.13) we have $\psi(\mathbf{x}, \tilde{s}, 0) = \psi(\mathbf{x}_1, \tilde{s}, 0)$, giving a contradiction with (C.11). On the other hand, if the inequality is strict (that is, $\delta(i_1, \tilde{s}_1) > \delta(v, \tilde{s}_1)$) then using Lemma 3.2 we infer that $\frac{\partial}{\partial t}\psi(\mathbf{x}_1, \tilde{s}, t)\big|_{t=0} > 0$. Hence, according to the rules of the heuristic, the sequence $\tilde{s}$ could not have been chosen under state $\mathbf{x}_1$. We conclude that $\delta(i_1, \tilde{s}_1) < \delta(v, \tilde{s}_1)$ and therefore the server's step from v to i_1 represents a step along a shortest path from v to $\tilde{s}_1$.

Subcase (b): σ_1 is empty

From the arguments at the beginning of the proof we know that even if σ_1 is empty, σ_2 must be non-empty. As in case (a), let s^* denote the sequence chosen by the heuristic under state $\mathbf{x}$, and let $(i_0, i_1, ..., i_k)$ denote a shortest path from v to s_1^*, where $k = \delta(v, s_1^*)$, $i_0 = v$ and $i_k = s_1^*$. Also let $\mathbf{x}_j$ denote a state identical to $\mathbf{x}$ except that the server is located at node i_j, for $j = 1, 2, ..., k$. After the server moves from node v to i_1, there are two possible scenarios: either σ_1 remains empty, or it becomes non-empty. In the first scenario, we can simply repeat the relevant arguments used in subcase (a) to show that, even if the heuristic chooses some alternative sequence $\tilde{s} \in \sigma_2$ under state $\mathbf{x}_1$, it must be the case that $\delta(i_1, \tilde{s}_1) < \delta(v, \tilde{s}_1)$, and therefore the server's movement from v to i_1 qualifies as a step along a shortest path from v to $\tilde{s}_1$. This is due to the fact that $\frac{\partial}{\partial t}\psi(\mathbf{x}_1, \tilde{s}, t)\big|_{t=0}$ must be non-positive in order for $\tilde{s}$ to be chosen. In the rest of this part we consider the second scenario, where σ_1 becomes non-empty.

Suppose $\tilde{s} \in \sigma_1$ and the heuristic chooses sequence $\tilde{s}$ under state $\mathbf{x}_1$. It may be the case that $\tilde{s}_1 = s_1^*$, in which case the server simply keeps following the same path. However, if $\tilde{s}_1 \neq s_1^*$, we can again show that $\delta(i_1, \tilde{s}_1) < \delta(v, \tilde{s}_1)$. To see this, let $\tilde{\mathbf{x}}$ denote the state identical to $\mathbf{x}$ except that the server is located at node $\tilde{s}_1$. Due to the rules of the heuristic, we must have

$$\xi(\mathbf{x}_1, \tilde{s}, 0) \geq \rho \tag{C.14}$$

and additionally, if $|\tilde{s}| \geq 2$, then

$$\xi(\tilde{\mathbf{x}}, \tilde{s}, 0) \geq \rho. \tag{C.15}$$

Note that $\xi(\mathbf{x}_1, \tilde{s}, 0) = \xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(i_1, \tilde{s}_1)/\tau)$. Suppose (for a contradiction) that $\delta(i_1, \tilde{s}_1) \geq \delta(v, \tilde{s}_1)$. Given that sequence $\tilde{s}$ was not included in σ_1 when the server was at state $\mathbf{x}$, at least one of the conditions $\frac{\partial}{\partial t}\psi(\mathbf{x}, \tilde{s}, t)\big|_{t=0} \leq 0$, $\xi(\mathbf{x}, \tilde{s}, 0) \geq \rho$ and $\xi(\tilde{\mathbf{x}}, \tilde{s}, 0) \geq \rho$ (where the latter only applies if $|\tilde{s}| \geq 2$) must fail to hold. However, given that $\tilde{s}$ is chosen at $\mathbf{x}_1$ and $\psi(\mathbf{x}_1, \tilde{s}, 0) \equiv \psi(\mathbf{x}, \tilde{s}, 1/\tau)$, it must be the case (using Lemma 3.2) that $\frac{\partial}{\partial t}\psi(\mathbf{x}_1, \tilde{s}, t)\big|_{t=0}$ has the same sign as $\frac{\partial}{\partial t}\psi(\mathbf{x}, \tilde{s}, t)\big|_{t=0}$, so the derivative condition holds. We also have $\xi(\tilde{\mathbf{x}}, \tilde{s}, 0) \geq \rho$ when $|\tilde{s}| \geq 2$ from (C.15), so we can proceed to assume that $\xi(\mathbf{x}, \tilde{s}, 0) < \rho$, which is equivalent to $\xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(v, \tilde{s}_1)/\tau) < \rho$. Hence, we have

$$\xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(v, \tilde{s}_1)/\tau) < \rho \leq \xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(i_1, \tilde{s}_1)/\tau) \tag{C.16}$$

but also (by assumption)

$$0 < \delta(v, \tilde{s}_1) \leq \delta(i_1, \tilde{s}_1), \tag{C.17}$$

implying that $\xi(\tilde{\mathbf{x}}, \tilde{s}, t)$ is increasing with t. If $|\tilde{s}| = 1$ then this gives a contradiction, since it was shown in (C.5) that $\xi(\cdot)$ is non-increasing with t for sequences of length one. On the other hand, if $|\tilde{s}| \geq 2$, we modify the right-hand side of (C.16) and combine (C.14)-(C.15) to give

$$\xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(v, \tilde{s}_1)/\tau) < \rho \leq \min\{\xi(\tilde{\mathbf{x}}, \tilde{s}, 0),\ \xi(\tilde{\mathbf{x}}, \tilde{s}, \delta(i_1, \tilde{s}_1)/\tau)\},$$

which, along with (C.17), contradicts the fact that $\xi(\tilde{\mathbf{x}}, \tilde{s}, t)$ is a monotonic function of t. We conclude that $\delta(i_1, \tilde{s}_1) < \delta(v, \tilde{s}_1)$ as required.

We can repeat the arguments given in subcases (a) and (b) to show that any movement of the server from one intermediate stage to another (prior to $\min\{T_{\text{arr}},\ T_{\text{switch}}\}$) qualifies as a step along a shortest path from v to some particular demand point j^*. The key point is that even though the sequences chosen at the various intermediate stages may not always begin with node j^*, the distance from the currently-occupied node to j^* is always reduced after each step, so the complete path is indeed a shortest path from v to j^*. Given that the number of nodes in the network is finite, the demand point j^* must eventually be reached if no new jobs arrive in the meantime. This completes the proof. □

D Proof of Corollary 3.4.

Proof. We assume that the server is initially located at some intermediate stage $v \in N$ and use $M = \max_{\{i\in N,\, j\in D\}} \delta(i,j)$ to denote the maximum distance between an intermediate stage and a demand point. Let α denote the number of new jobs that arrive in the system before the server reaches a demand point, given that the K-stop heuristic is followed. By conditioning on α, we have

$$\mathbb{E}[T_{\text{switch}}] = \sum_{k=0}^{\infty} \mathbb{E}[T_{\text{switch}} \mid \alpha = k]\ \mathbb{P}(\alpha = k). \tag{D.1}$$

Due to Theorem 3.3 we know that if the server is at an intermediate stage, then until the next new job arrives it attempts to move along a shortest path to some particular demand point j^*. Since (prior to T_{switch}) the server is always attempting to move, the expected amount of time until the next event (either a switch or the arrival of a new job in the system) is always $(\Lambda + \tau)^{-1}$, where $\Lambda = \sum_{i\in D} \lambda_i$. In the worst case, the number of nodes that must be traversed in order to reach j^* is M. Hence, we can form an upper bound for $\mathbb{E}[T_{\text{switch}} \mid \alpha = 0]$:

$$\mathbb{E}[T_{\text{switch}} \mid \alpha = 0] \le \frac{M}{\Lambda + \tau}.$$

Extending this argument, we can obtain an upper bound for $\mathbb{E}[T_{\text{switch}} \mid \alpha = k]$ (for $k \ge 1$) by supposing that every time a new job arrives in the system, the server changes direction and attempts to move to a demand point M nodes away, and manages to complete $(M-1)$ of these switches before a new job arrives in the system and forces it to change direction again. Suppose this pattern continues until k arrivals have occurred, at which point it manages to complete M switches without interruption and reaches a demand point. In this scenario, the total number of switches made is $k(M-1) + M$ and the total number of new jobs arriving is k, so the total number of system events is $(k+1)M$. Hence:

$$\mathbb{E}[T_{\text{switch}} \mid \alpha = k] \le \frac{(k+1)M}{\Lambda + \tau}, \quad k \ge 0.$$

Next, consider the probabilities $\mathbb{P}(\alpha = k)$. Each time a system event occurs (prior to T_{switch}), there is a probability of $\tau/(\Lambda+\tau)$ that this is a switch rather than the arrival of a new job. We can obtain an upper bound for $\mathbb{P}(\alpha = 0)$ by supposing that the server only needs to complete one switch in order to reach its intended demand point. Hence:

$$\mathbb{P}(\alpha = 0) \leq \frac{\tau}{\Lambda+\tau}.$$

On the other hand, the largest possible probability of an arrival occurring before the server reaches a demand point is $1-(\tau/(\Lambda+\tau))^M$, since this represents the case where the server must complete M switches in order to reach its intended demand point. Putting these arguments together, we have the following upper bound:

$$\mathbb{P}(\alpha = k) \leq \left[1-\left(\frac{\tau}{\Lambda+\tau}\right)^M\right]^k \left(\frac{\tau}{\Lambda+\tau}\right), \quad k \geq 0.$$

Let $p := 1-(\tau/(\Lambda+\tau))^M$ for notational convenience. Then, using (D.1), we have

$$\begin{aligned}
\mathbb{E}[T_{\text{switch}}] &\leq \sum_{k=0}^{\infty} \frac{(k+1)M}{\Lambda+\tau}\, p^k \left(\frac{\tau}{\Lambda+\tau}\right) \\
&= \frac{\tau M}{(\Lambda+\tau)^2} \sum_{k=0}^{\infty} (k+1)p^k \\
&= \frac{\tau M}{(\Lambda+\tau)^2} \cdot \frac{1}{(1-p)^2} \\
&= \frac{\tau M}{(\Lambda+\tau)^2} \left(\frac{\Lambda+\tau}{\tau}\right)^{2M} \\
&= \frac{M}{\tau} \left(\frac{\Lambda+\tau}{\tau}\right)^{2(M-1)}. \qquad \text{(D.2)}
\end{aligned}$$

This completes the proof. We also note that the bound holds with equality if and only if $M = 1$. $\square$

E Proof of Theorem 3.5.

Proof. We will use λ, μ and c to denote the common arrival rate, service rate and holding cost (respectively) for all job types. Firstly, let $\mathbf{x}$ be a state under which the server is located at a demand point $i \in D$ with $x_i \geq 1$. Recall from step 1(a) of the K-stop heuristic algorithm that $\mathcal{S}$ is the set of sequences of the form $s = (s_1, s_2, ..., s_m)$, where $1 \leq m \leq K$, $s_j \in D$ for each $j \in \{1, 2, ..., m\}$, $s_1 \neq v$ and $s_i \neq s_j$ for any pair of elements $s_i, s_j \in s$ with $i \neq j$. For any sequence $s \in \mathcal{S}$, $t \geq 0$ and $j =\in \{1, ..., |s|\}$ it is clear from (12) that

$$\phi_j(\mathbf{x}, s, t) < \frac{\sum_{k=1}^{j} R_k(\mathbf{x}, s, t)}{\sum_{k=1}^{j} T_k(\mathbf{x}, s, t)} = \frac{c\mu \sum_{k=1}^{j} T_k(\mathbf{x}, s, t)}{\sum_{k=1}^{j} T_k(\mathbf{x}, s, t)} = c\mu.$$

On the other hand, the quantity $\beta_j(\mathbf{x}, s, t)$ in (13) is equal to $c\mu$. Therefore the condition $\phi_j(\mathbf{x}, s, t) \geq \beta(\mathbf{x}, s, t)$ is not satisfied for any $s \in \mathcal{S}$, and according to the rules of the heuristic the server should remain at node i. The intuitive explanation for this is that the server earns rewards at the maximum possible rate $c\mu$ by remaining at its current node, so there is no reason to switch to another node. It follows that, under the K-stop policy, the server always remains at a demand point until all jobs have been processed.

We can also show that the server visits all demand points infinitely often under the K-stop policy. Indeed, suppose (for a contradiction) that there exists some non-empty subset $D_0 \subset D$ such that demand points in D_0 are visited only finitely many times. We also define $D_1 := D \setminus D_0$ as the subset of demand points that are visited infinitely often. The subset D_1 must be non-empty because, as noted in the proof of Theorem 3.3, sequences s of length one always satisfy the condition $\frac{\partial}{\partial t}\psi(\mathbf{x}, s, t)\big|_{t=0} \leq 0$, and therefore the set σ constructed in step 2 of the heuristic algorithm must be non-empty, which implies that if the server is at an intermediate stage then it will move towards a demand point. Each time the server selects a demand point in $i \in D_1$ to move to, there is a positive probability that no new jobs arrive at any of the other demand points $j \in D_1 \setminus \{i\}$ while it switches to i and processes jobs there. Therefore the system must eventually reach a state in which there are no jobs at any of the demand points in D_1. Let $\mathbf{x}$ be a state with $x_j = 0$ for all $j \in D_1$ and let $\mathcal{S}_1$ denote the set of all sequences in $\mathcal{S}$ that involve visiting only demand points in D_1. Also, define

$$\psi_1^{\max} := \max_{s \in \mathcal{S}_1} \psi(\mathbf{x}, s, 0).$$

It is clear from the definition of $\psi(\mathbf{x}, s, 0)$ in (11) that $\psi_1^{\max}$ is strictly smaller than $c\mu$. Next, consider an arbitrary demand point $j \in D_0$ and consider the sequence of length one, $s = (j)$. We have assumed that demand point j is visited only finitely many times and therefore, in the long run, x_j tends to infinity. From (11) it follows that the index $\psi(\mathbf{x}, (j), 0)$ tends towards $R_1(\mathbf{x}, (j), 0)/T_1(\mathbf{x}, (j), 0) = c\mu$. Hence, at some point $\psi(\mathbf{x}, (j), 0)$ must exceed $\psi_1^{\max}$. Similar reasoning can be used to show that any sequence s that begins by visiting a demand point in D_1 must eventually become inferior (in terms of index value) to the sequence (j), where $j \in D_0$. The rules of the heuristic then imply that the server selects a sequence which begins by visiting one of the demand points in D_0, which yields a contradiction. We conclude that all demand points must be visited infinitely often under the K-stop policy. Standard arguments based on stability in polling systems (see the comments following the statement of Theorem 2.2) then imply that the system is stable, which proves the first statement in the theorem.

Next, we make the additional assumption that V is a complete graph, which implies that the server can move between any two demand points $i, j \in D$ by traversing a single edge of the network, without having to pass through any intermediate stages. We will show that, with this extra assumption (in addition to the homogeneity assumptions made already), the K-stop policy causes the server to act in the following way:

(i) If the server is at a non-empty demand point $i \in D$, then it remains there.

(ii) If the server is at an empty demand point $i \in D$, then it switches to the demand point $j \neq i$ with the largest number of jobs present. (If there is a tie, an arbitrary choice of j can be made.)

Note that the rules (i)-(ii) are sufficient to completely specify the actions chosen by the K-stop policy, since it cannot ever visit an intermediate stage. Property (i) was already established in the first part of the proof (the additional assumption that V is a complete graph does not alter the argument given previously). To establish property (ii), we again recall the previous argument and note that the set σ constructed in step 2 of the heuristic algorithm is non-empty, because it includes sequences of length one. It follows that, given some state $\mathbf{x}$ under which the server is at an empty demand point, it selects the sequence s that maximizes $\psi(\mathbf{x}, s, 0)$. In general, the sequence selected by the K-stop policy could be of any length between 1 and K. However, we can show that if a sequence of length greater than one is selected, then the demand points in the sequence must be visited in descending order of x_i; that is, the server prioritizes the demand point with the greatest number of jobs. To see this, note that for a sequence s of length m (where $1 \leq m \leq K$), the index $\psi(\mathbf{x}, s, 0)$ can be expressed as

$$\psi(\mathbf{x}, s, 0) = \frac{cu \sum_{k=1}^{m} T_k(\mathbf{x}, s, 0)}{m/\tau + \sum_{k=1}^{m} T_k(\mathbf{x}, s, 0)}, \tag{E.1}$$

where the index k corresponds to the position of demand point s_k in the sequence, so $T_1(\mathbf{x}, s, 0)$ is the amount of time spent at the first demand point in the sequence (under the fluid model), etc. Suppose we have a sequence in which the demand points are *not* ordered in descending order of x_i. This implies that there must be some $k \in \{1, ..., m-1\}$ such that $x_{s_k} < x_{s_{k+1}}$. We will show that the index in (E.1) would increase if we swapped the positions of demand points s_k and s_{k+1} in the sequence. Indeed, let $y = x_{s_{k+1}} - x_{s_k} > 0$. After performing the swap, $T_k(\mathbf{x}, s, 0)$ increases by $y/(\mu - \lambda)$ since, under the fluid model, the amount of time taken to process a single job is $1/(\mu - \lambda)$. On the other hand, $T_{k+1}(\mathbf{x}, s, 0)$ increases by $\lambda y/((\mu - \lambda)^2) - y/(\mu - \lambda)$, where the first term $\lambda y/((\mu - \lambda)^2)$ is due to the extra arrivals that occur during the extra $y/(\mu - \lambda)$ time units spent at node s_k and the second term $-y/(\mu - \lambda)$ is due to the fact that there are y fewer jobs at s_{k+1} after performing the swap. Hence, the overall increase in $T_k(\mathbf{x}, s, 0) + T_{k+1}(\mathbf{x}, s, 0)$ is $\lambda y/((\mu - \lambda)^2)$, which is non-negative. Furthermore, $T_l(\mathbf{x}, s, 0)$ increases for all $l \geq k+2$ following the swap, due to the fact that $T_k(\mathbf{x}, s, 0) + T_{k+1}(\mathbf{x}, s, 0)$ increases and therefore extra arrivals occur at these nodes before the server arrives. The result is that the sum $\sum_{k=1}^{m} T_k(\mathbf{x}, s, 0)$ increases following the swap, and therefore the index in (E.1) also increases. By extending this argument, we can claim that if s is a sequence in which the demand points are *not* visited in descending order of x_i, then it is beneficial to perform a sequence of swaps until they are in descending order. It follows that the sequence that maximizes (E.1) must be a sequence in which the first node visited has the greatest number of jobs.

Having established that properties (i) and (ii) hold under the K-stop policy, the next task is to show that these properties also hold under an optimal policy, and thus the K-stop policy is optimal. We can show this using a sample path argument. To establish (i), suppose the system is in state $\mathbf{x}$ with $v(\mathbf{x}) = i \in D$, $x_i \geq 1$. Consider two continuous-time processes, referred to as P_1 and P_2 for convenience, both initialized in state $\mathbf{x}$. In the continuous-time setting, an action $a(t) \in D$ must be specified by the server for each time point $t \in \mathbb{R}_{\geq 0}$. In P_1 we suppose that the server follows a policy whereby the action at $t = 0$ is to switch to demand point $j \neq i$. In P_2, on the other hand, the server remains at node i until a service completion occurs there, then copies the same sequence of actions that P_1 chose starting from $t = 0$, except that it does not choose i again (skipping these actions as necessary) until P_1 has completed its first service at i. Using standard stochastic coupling arguments, we assume that new job arrivals occur at the same times under both processes, and furthermore the times required for service completions and switch completions are the same under both processes, although (due to the difference in server behaviors) a particular service or switch that begins at time $t \geq 0$ in P_1 may begin at a different time $t' \neq t$ under P_2. As an example, suppose the trajectory of (action, duration, completion) pairs under P_1 evolves as follows:

$$(j^{\text{switch}}, T_0, \checkmark), (j^{\text{serv}}, T_1, \times), (i^{\text{switch}}, T_2, \checkmark), (i^{\text{serv}}, T_3, \times), (k^{\text{switch}}, T_4, \checkmark), ..., (i^{\text{serv}}, T_m, \checkmark), ...$$

which indicates that the server begins by switching to j for T_0 time units and this action is completed (denoted by '$\checkmark$'), then it serves a job at node j for T_1 time units but this service is not completed (denoted by '$\times$') because it interrupts the service to switch to node i, and the switch to i takes T_2 units and is successfully completed, etc. We use m to represent the sequence position of the first completed service at i. Then the corresponding trajectory under P_2 would begin with

$$(i^{\text{serv}}, T_m, \checkmark), (j^{\text{switch}}, T_0, \checkmark), (j^{\text{serv}}, T_1, \times), (k^{\text{switch}}, T_4, \checkmark), ...$$

Let $\hat{T}$ denote the total amount of time that the server spends switching to demand point i (whether the switches are completed or not) in P_1 before it eventually completes a service there. In P_2 we assume that, after the server has finished copying the non-i actions of the server in P_1 prior to $T_0 + ... + T_{m-1}$, it then chooses action i for a further $\hat{T}$ time units. This ensures that both processes are in the same state at time $T_0 + ... + T_m$, and they continue to evolve identically from that point onwards.

We can compare the total costs incurred by P_1 and P_2 under the above coupling construction. We note that:

(a) Any service completion that occurs in P_1 at time $t \geq 0$, except for the first job at demand point i, occurs with a delay of no more than T_m time units in P_2. Therefore P_2 incurs extra holding costs of (at most) $c \times R \times T_m$ due to the delayed processing of these jobs, where R is the total number of jobs processed in P_1 before processing the first job at node i.

(b) The first job at demand point i is processed at time T_m in P_2, but is not processed until time $T_0 + ... + T_m$ in P_1. Therefore P_1 incurs an extra holding cost of $c \times (T_0 + ... + T_{m-1})$ due to the delayed processing of this job.

Note that the set $\{T_0, ..., T_{m-1}\}$ includes the times needed to process R jobs at other demand points before the first job is processed at i under P_1. Hence, given that the service times of all jobs are independent and identically distributed, we have $\mathbb{E}[T_0 + ... + T_{m-1}] \geq \mathbb{E}[R \times T_m]$, so the extra costs incurred in P_1 are indeed higher than those in P_2. Therefore, switching away from a non-empty demand point is a suboptimal action in the context of minimizing total costs over a long time horizon. Intuitively, the suboptimality occurs because the server is forced to perform extra switching actions in P_1.

Next, we move on to property (ii) and aim to show that if the server is at an empty demand point then it is optimal to switch to the demand point with the largest number of jobs. It is sufficient to show that for any state $\mathbf{x} = (i, (x_1, ..., x_d))$ with $x_i < x_j$ for some $j \in D$, the minimal expected total cost (over a long time horizon) starting from state $\mathbf{x}$ is greater than it would be if x_i and x_j were swapped. We will again use a sample path argument. Let P_1 and P_2 be two continuous-time processes initialized at in states $\mathbf{x} = (i, (x_1, ..., x_d))$ and $\mathbf{x}' = (i, (x'_1, ..., x'_d))$ respectively, where $\mathbf{x}'$ is identical to $\mathbf{x}$ except that $x'_i = x_j$ and $x'_j = x_i$, and we assume $x_i < x_j$. The two processes are coupled in the same way as in (i), so that new job arrivals occur at the same times under both processes and the times needed for successful service and switch completions are also equivalent.

Suppose that in P_1 the server remains at node i until $y \geq 0$ jobs have been processed before attempting to switch to a different demand point. In P_2, the server begins by remaining at node i until $y + 1$ services have been completed before switching to another demand point, so we force it to process an extra job before switching. It then copies the same actions as the server in P_1, but 'skips' any actions j chosen in P_1 (recall that P_1 has at least one extra job at j) until eventually P_1 completes its first service at j. For example, if $y = 1$ then the trajectories under P_1 and P_2 could be as follows:

$$\begin{aligned} P_1 &: \ (i^{\text{serv}}, T_0, \checkmark), (j^{\text{switch}}, T_1, \times), (k^{\text{switch}}, T_2, \checkmark), ..., (j^{\text{serv}}, T_m, \checkmark), ... \\ P_2 &: \ (i^{\text{serv}}, T_0, \checkmark), (i^{\text{serv}}, T_m, \checkmark), (k^{\text{switch}}, T_2, \checkmark), ... \end{aligned}$$

for some $m \geq 0$. The important thing in this case is to make a comparison between the first job processed at j in P_1 and the $(y+1)^{\text{th}}$ job processed at i in P_2, and our coupling construction assumes these jobs have the same service time (we can think of them as the same job, which begins at j in P_1 but begins at i in P_2). Similarly to (i), we also let $\hat{T}$ denote the total amount of time spent by the server switching to demand point j in P_1 before it completes a service there and specify that in P_2, the server should spend $\hat{T}$ time units choosing action j after it finishes copying the non-j actions of the server in P_1 prior to $T_0 + ... + T_{m-1}$. This ensures that both processes are in the same state at time $T_0 + ... + T_m$.

The comparison between the costs incurred by the two processes then works in a similar way to the comparison in (i). In P_2, there is a certain number of jobs that have their services delayed by (at most) T_m time units compared to P_1, but the extra holding costs for these are collectively smaller than the extra holding cost incurred by P_1 as a result of processing the first job at j later than the processing of the $(y + 1)^{\text{th}}$ job at i in P_2.

We note that the argument given above becomes most meaningful in the case where the server in P_1 remains at the initial node i until it is empty (which it must do under an optimal policy, according to our previous argument). In this case the server in P_2 can still process an extra job at i, but the server in P_1 has to switch at least once before processing the corresponding extra job at node j. We conclude that it is always beneficial for the server to be at the demand point with the largest number of jobs present, and it follows that if the server is at an empty demand point then it should attempt to switch to the demand point with the largest number of jobs. We have shown that, in all cases, the actions of the K-stop policy are identical to those selected by an optimal policy. □

F Methods for generating the parameters for the numerical experiments in Section 4

For each of the 11,250 instances considered in Section 4.1, the system parameters are randomly generated as follows:

- The size of the left-hand cluster, d_1, is sampled unbiasedly from the set $\{1, 2, 3, 4\}$.
- The size of the right-hand cluster, d_2, is sampled unbiasedly from the set $\{1, 2, 3, 4\}$.
- The number of intermediate stages, n, is sampled unbiasedly from the set $\{1, 2, 3, 4, 5, 6\}$.
- The overall traffic intensity, ρ, is sampled from a continuous uniform distribution between 0.1 and 0.9 in 10,000 experiments, and sampled from a continuous uniform distribution between 0.9 and 1 in the remaining 1,250 experiments. Subsequently, the job arrival rates λ_i and processing rates μ_i for $i \in D$ are generated as follows:
  - Each processing rate μ_i is initially sampled from a continuous uniform distribution between 0.1 and 0.9.
  - For each demand point $i \in D$, an initial value for the job arrival rate λ'_i is sampled from a continuous uniform distribution between $0.1\mu_i$ and μ_i.
  - For each demand point $i \in D$, the actual traffic intensity ρ_i is obtained by re-scaling the initial traffic intensity λ'_i/μ_i, as follows:

    $$\rho_i := \frac{\lambda'_i/\mu_i}{\sum_{i\in D}\lambda'_i/\mu_i}\rho.$$

    This ensures that $\sum_{i\in D}\rho_i = \rho$.
  - For each demand point $i \in D$, the actual job arrival rate λ_i is obtained as follows:

    $$\lambda_i := \rho_i\mu_i.$$

  - All of the λ_i, μ_i and τ values are re-scaled in order to ensure that $\sum_{i\in D}\lambda_i + \max\{\mu_1, ..., \mu_d, \tau\} = 1$ (as assumed in Section 2) and then rounded to 2 significant figures.

- For each demand point $i \in D$, the holding cost c_i is sampled from a continuous uniform distribution between 0.1 and 0.9.

- In order to generate the switching rate τ, we first define $\eta := \tau/(\sum_{i \in D} \lambda_i)$ and generate the value of η as follows:

  – Sample a value p from a continuous uniform distribution between 0 and 1.

  – If $p < 0.5$, sample η from a continuous uniform distribution between 0.1 and 1.

  – If $p \geq 0.5$, sample η from a continuous uniform distribution between 1 and 10.

  We then define $\tau := \eta \sum_{i \in D} \lambda_i$.

G Simulation methods for the numerical experiments in Section 4

In each of the problem instances considered in Sections 4.1 and 4.2, we first generate a set of random system parameters as described in Appendix F. Since the overall traffic intensity ρ may approach 1, potentially leading to very large average queue sizes or instability under some heuristics considered in this paper, we treat two cases separately: $\rho < 0.9$ and $\rho \geq 0.9$.

Case (a): Moderate traffic intensities ($\rho < 0.9$)

For moderate traffic intensities, the performances of the K-stop and (K from L)-stop heuristics are estimated by simulating the discrete-time evolution of the uniformized MDP described in Section 2. We also use a 'common random numbers' method to ensure that job arrivals occur at the same times under each of these policies. More specifically, the steps of the simulation procedure are as follows:

1. Set $N_0 := 10,000$ as the length of the warm-up period and $N_1 := 1,000,000$ as the length of the main simulation.

2. Generate a list Z of length $N_0 + N_1 = 1,010,000$, consisting of uniformly-distributed random numbers between 0 and 1.

3. Consider each of the K-stop and (K from L)-stop heuristics in turn. For each one, set $\mathbf{x}_0 := (1, (0, 0, ..., 0))$ as the initial state and use the first N_0 random numbers in Z to simulate events in the first N_0 time steps (this is done by sampling from the transition probability distribution described in (2). This is the 'warm-up period'. Let $\mathbf{y}$ denote the state reached at the end of the warm-up period. Then, use the remaining N_1 random numbers in Z to simulate events during the next N_1 time steps, with the system beginning in state $\mathbf{y}$, and use the statistics collected during these time steps to quantify the heuristic's performance.

The method for simulating the DVO heuristic is different, because (as discussed in Section 3) this heuristic is not directly compatible with the MDP formulation in the paper. To simulate the DVO heuristic, we simulate in continuous time as follows:

1. Use the same set of randomly-generated system parameters used for the other heuristics.
2. Let $T_0 = 10,000\Delta$ and $T_1 = 1,000,000\Delta$ as the warm-up period length and the main simulation length, respectively, where Δ is defined in (1).
3. Set $\mathbf{x}_0 := (1, (0, 0, ..., 0))$ as the initial state. Note that, under the DVO heuristic, the set of decision epochs is not the same as under the other heuristics (see Section 3.1). At each decision epoch we make a decision using the rules of the DVO heuristic and then simulate the time until the next decision epoch, which requires simulating from an exponential distribution (if the server processes a job or idles at an empty demand point) or an Erlang distribution (if the server switches to another demand point). Costs are accumulated based on the job counts x_i at the various demand points in between decision epochs. This process continues until the total time elapsed exceeds T_0, at which point the warm-up period ends. Let $\mathbf{y}$ denote the state reached at the end of the warm-up period. We then carry out the main simulation in the same way, starting from state $\mathbf{y}$ and continuing for a further T_1 time units, at which point the process ends and the performance of the DVO heuristic is estimated by dividing the total costs incurred during the main simulation by the total time elapsed.

Case (b): High traffic intensities ($\rho \geq 0.9$)

For high traffic intensities, the length of the warm-up period under any heuristic considered in this paper is determined through repeated application of the Mann–Kendall Test with the Hamed–Rao correction (see Hamed and Rao (1998), Gocic and Trajkovic (2013)), a nonparametric method for detecting monotonic trends in time series while accounting for autocorrelation. This assessment is performed at regular intervals during the warm-up period, up to a ceiling length of $N_0 =$ 1,000,000 time steps (or $T_0 =$ 1,000,000Δ for the DVO heuristic), ensuring that the heuristic's performance is to be evaluated only once the system has reached a stable regime. The procedure is as follows:

1. **Sample collection:** During the warm-up simulation, one new observation is collected every 1000 time steps (or every 1000Δ time units for the DVO heuristic) by computing the cumulative average cost up to that point. Once the number of observations exceeds m (where $m = 100$), the latest m samples are used for trend testing.
2. **Trend detection using the Mann–Kendall Test:** Let $\{g_1, \ldots, g_m\}$ denote the latest m observations. The Mann–Kendall S statistic is computed as

$$S = \sum_{k=1}^{m-1} \sum_{j=k+1}^{m} \operatorname{sgn}(g_j - g_k), \quad \operatorname{sgn}(y) = \begin{cases} 1 & y > 0, \\ 0 & y = 0, \\ -1 & y < 0. \end{cases}$$

The variance of S is adjusted to account for tied values. Let t_i denote the number of tied values in the ith tied group, and suppose there are p such groups. Then

$$\mathrm{Var}(S) = \frac{m(m-1)(2m+5) - \sum_{i=1}^{p} t_i(t_i-1)(2t_i+5)}{18}.$$

3. **Hamed–Rao correction for autocorrelation:** Let $\tilde{g}_k$ denote the rank of the kth observation of the latest m observations, and let $\bar{g}$ be the mean rank. The sample autocorrelation at lag i is

$$r_i = \frac{\sum_{k=1}^{m-i}(\tilde{g}_k - \bar{g})(\tilde{g}_{k+i} - \bar{g})}{\sum_{k=1}^{m}(\tilde{g}_k - \bar{g})^2}, \quad i = 1, \dots, m-1.$$

Significant lags are identified when $|r_i| > 1.96/\sqrt{m}$, corresponding to a 95% confidence level. The effective sample size accounting for significant autocorrelations is

$$m^* = \frac{m}{1 + \frac{2}{m(m-1)(m-2)} \sum_i (m-i)(m-i-1)(m-i-2)\, r_i},$$

where the summation is taken over all significant lags. The corrected variance of S is

$$\mathrm{Var}^*(S) = \mathrm{Var}(S)\,\frac{m}{m^*},$$

and the standardized test statistic is

$$Z = \begin{cases} \dfrac{S-1}{\sqrt{\mathrm{Var}^*(S)}} & S > 0, \\ 0, & S = 0, \\ \dfrac{S+1}{\sqrt{\mathrm{Var}^*(S)}} & S < 0. \end{cases}$$

4. **Decision rule:** The test compares $|Z|$ with the critical value $Z_{\text{critical}} = 1.96$, corresponding to a 95% confidence level (two-tailed). If $|Z| < 1.96$, no significant trend is detected, and the system is considered to have stabilized; the current state $\mathbf{y}$ is recorded, and the main simulation proceeds as in Case (a), starting from $\mathbf{y}$ without performing an additional warm-up period. On the other hand, if a significant trend is detected, the simulation continues, collecting new samples every 1000 time steps (or 1000Δ time units), and the latest m samples are tested again. This process repeats until either stability is observed or the ceiling length N_0 (or T_0 for DVO) is reached. If the system remains unstable at the ceiling length, evaluation proceeds to the next heuristic or instance.

In our experiments, we found that the DVO, K-stop and (K from L)-stop heuristics were able to attain system stability before reaching the time limit in the vast majority of cases. Specifically, each of these heuristics met the stability criterion within the time limit in between 95-98% of instances with $\rho \geq 0.9$. In the minority of cases where the stability criterion was not met within the time limit, the average ρ value was about 0.985.

H Method for testing the feasibility of dynamic programming (DP) for the numerical experiments in Section 4

As explained in Section 4, we aim to use DP (specifically, relative value iteration) to compute the optimal average cost g^* whenever it is computationally feasible to do so. For a particular problem instance, we carry out the following steps in order to classify it as either 'feasible' or 'infeasible' and (for the feasible instances only) estimate the optimal average cost:

1. If $d \geq 4$, classify this instance as infeasible.
2. Otherwise (if $d \leq 3$), carry out the following steps:
   (a) Set $m = 10$, $t = 0$ and $g^* = 0$.
   (b) Consider a finite-state MDP in which the number of jobs present at any demand point is not allowed to exceed m, as described in the proof of Theorem 2.2. Let M denote the size of the state space, given by
   $$M := (d + n)(m + 1)^d.$$
   (Recall that d and n are the numbers of demand points and intermediate stages in the network, respectively.) If $M \geq 1,000,000$, go to step (d). Otherwise, solve the finite-state MDP using DP, let t denote the time taken (in seconds) and let g^* denote the optimal average cost.
   (c) If $t < 600$, increase m by 10 and return to step (b). Otherwise, continue to step (d).
   (d) If either (i) $g^* = 0$, or (ii) the latest value of g^* exceeds the previous value by more than ϵ (where we set $\epsilon = 0.001$), then classify this instance as infeasible. Otherwise, classify it as feasible and let the latest value of g^* be an approximation for the optimal average cost in the infinite-state MDP.

References

Altman, E., Konstantopoulos, P., and Liu, Z. (1992). Stability, monotonicity and invariant quantities in general polling systems. *Queueing Systems*, 11:35–57.

Duenyas, I. and Van Oyen, M. P. (1996). Heuristic scheduling of parallel heterogeneous queues with set-ups. *Management Science*, 42(6):814–829.

Gocic, M. and Trajkovic, S. (2013). Analysis of changes in meteorological variables using Mann-Kendall and Sen's slope estimator statistical tests in Serbia. *Global and Planetary Change*, 100:172–182.

Hamed, K. H. and Rao, A. R. (1998). A modified Mann-Kendall trend test for autocorrelated data. *Journal of Hydrology*, 204(1-4):182–196.

Puterman, M. (1994). *Markov Decision Processes - Discrete Stochastic Dynamic Programming.* Wiley & Sons, New York.

Sennott, L. I. (1997). The computation of average optimal policies in denumerable state Markov decision chains. *Advances in Applied Probability*, 29(1):114–137.

Sennott, L. I. (1999). *Stochastic dynamic programming and the control of queueing systems.* John Wiley & Sons.