Electronic Companion of "Service Oriented Considerate Routing: Data, Predictions and Robust Decisions"

Appendix A Proofs

Proof of Theorem 1. We prove the ℓ_p-based CALM metric satisfies each property in the following.

(a) Matched Assignment. Assuming $\boldsymbol{\zeta} = \boldsymbol{\xi} \cdot \mathbf{1}^\top \boldsymbol{\zeta}$, we can formulate an optimal solution where

$$s_{ij} = \begin{cases} \zeta_i, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases},$$

resulting in a zero objective value. Conversely, if $\rho(\boldsymbol{\zeta}, \boldsymbol{\xi}) = 0$, it is necessary that $s_{ij} = 0$ for all $i \neq j$, given that $d_{ij} > 0$. As a result, the first and second lines of constraints in Equation (4) simplify to $s_{ii} = \xi_i \mathbf{1}^\top \boldsymbol{\zeta},\ s_{ii} = \zeta_i,\ \forall i \in [M]$. This leads to the conclusion that $\boldsymbol{\zeta} = \boldsymbol{\xi} \cdot \mathbf{1}^\top \boldsymbol{\zeta}$.

(b) Basic Location Mismatch. This property is trivial since the unique feasible solution to $\rho(e_i, e_j)$ is $s_{tk} = 1$ when $t = i, k = j$ and 0, otherwise, which implies that $\rho(e_i, e_j) = d_{ij}$.

(c) Positive Homogeneous. For any $\alpha > 0$, we have

$$\begin{aligned} \rho(\alpha\boldsymbol{\zeta}, \boldsymbol{\xi}) = \min & \left(\sum_{i \in [M]} \sum_{j \in [M]} \left(d_{ij} s_{ij}\right)^p \right)^{1/p} \\ \text{s.t.} & \sum_{i \in [M]} s_{ij} = \alpha \xi_j \mathbf{1}^\top \boldsymbol{\zeta} && \forall j \in [M] \\ & \sum_{j \in [M]} s_{ij} = \alpha \zeta_i && \forall i \in [M] \\ & \boldsymbol{s} \geq \mathbf{0} \\ = \min & \left(\alpha^p \sum_{i \in [M]} \sum_{j \in [M]} \left(d_{ij} s_{ij}/\alpha\right)^p \right)^{1/p} \\ \text{s.t.} & \sum_{i \in [M]} s_{ij}/\alpha = \xi_j \mathbf{1}^\top \boldsymbol{\zeta} && \forall j \in [M] \\ & \sum_{j \in [M]} s_{ij}/\alpha = \zeta_i && \forall i \in [M] \\ & \boldsymbol{s} \geq \mathbf{0} \end{aligned}$$

$$
\begin{aligned}
= \quad & \min \quad \alpha\left(\sum_{i\in[M]}\sum_{j\in[M]}\left(d_{ij}s_{ij}\right)^p\right)^{1/p} \\
& \text{s.t.} \quad \sum_{i\in[M]} s_{ij} = \xi_j \mathbf{1}^\top \boldsymbol{\zeta} \qquad \forall j \in [M] \\
& \qquad \sum_{j\in[M]} s_{ij} = \zeta_i \qquad \forall i \in [M] \\
& \qquad \boldsymbol{s} \geq \mathbf{0} \\
= \ & \alpha\rho(\boldsymbol{\zeta}, \boldsymbol{\xi}).
\end{aligned}
$$

(d) Sub-additive. Suppose $\boldsymbol{s}^1, \boldsymbol{s}^2$ are the optimal solutions to $\rho(\boldsymbol{\zeta}_1, \boldsymbol{\xi})$ and $\rho(\boldsymbol{\zeta}_2, \boldsymbol{\xi})$ respectively, then $\boldsymbol{s}^1 + \boldsymbol{s}^2$ is a feasible solution corresponds to $\rho(\boldsymbol{\zeta}_1 + \boldsymbol{\zeta}_2, \boldsymbol{\xi})$. Therefore,

$$
\begin{aligned}
\rho(\boldsymbol{\zeta}_1 + \boldsymbol{\zeta}_2, \boldsymbol{\xi}) &\leq \left(\sum_{i\in[M]}\sum_{j\in[M]}\left(d_{ij}(s^1_{ij} + s^2_{ij})\right)^p\right)^{1/p} \\
&\leq \left(\sum_{i\in[M]}\sum_{j\in[M]}\left(d_{ij}s^1_{ij}\right)^p\right)^{1/p} + \left(\sum_{i\in[M]}\sum_{j\in[M]}\left(d_{ij}s^2_{ij}\right)^p\right)^{1/p} \\
&= \rho(\boldsymbol{\zeta}_1, \boldsymbol{\xi}) + \rho(\boldsymbol{\zeta}_2, \boldsymbol{\xi}),
\end{aligned}
$$

where the second inequality holds for any $p \geq 1$ by Minkowski's inequality. □

Proof of Theorem 3. First, the generic workload function can be linearized by introducing a binary vector $\boldsymbol{z}_k$ as follows:

$$
g(\hat{\boldsymbol{w}}_L, \boldsymbol{x}_k) = \min_{\boldsymbol{z}_k \in \{0,1\}^L} \left\{ \sum_{\ell\in[L]} \hat{w}_\ell z_{k\ell} \,\middle|\, \begin{array}{l} \mathbf{1}^\top \boldsymbol{x}_k = \sum_{\ell\in[L]} (\ell - 1) z_{k\ell} \\ \mathbf{1}^\top \boldsymbol{z}_k = 1 \end{array} \right\}.
$$

This linearization is valid because the workload $\mathbf{1}^\top \boldsymbol{x}_k$ can only assume values from the set $\{0, 1, \ldots, L-1\}$. Moreover, by definition of the CALM metric, we have

$$
\begin{aligned}
\rho\left(\boldsymbol{H}\boldsymbol{x}_k, \hat{\boldsymbol{\xi}}_k\right) &= \left\{ \begin{array}{ll} \min & \sum_{i,j\in[M]} d_{ij}s_{ij} \\ \text{s.t.} & \sum_{i\in[M]} s_{ij} = \hat{\xi}_{kj}\mathbf{1}^\top \boldsymbol{x}_k \ \ \forall j \in [M] \\ & \sum_{j\in[M]} s_{ij} = \boldsymbol{h}_i^\top \boldsymbol{x}_k \quad \forall i \in [M] \\ & \boldsymbol{s} \geq \mathbf{0} \end{array} \right. \\
&= \rho_k(\boldsymbol{x}_k).
\end{aligned}
$$

By utilizing the two reformulations and introducing an auxiliary variable λ, we establish the equivalence between Problem (8) and Problem (9). The second equality in Equation (10) is derived from the application of linear optimization duality.

□

Proof of Theorem 2. (i)We first prove the case when $M \geq 3$.

("⇒") Proof for the sufficient condition. Suppose $p = 1$, $d_{ii=0}$ and $d_{ij} = 2\kappa$ for $i \neq j$. Now define the following

$$\begin{aligned}\bar{\rho}(\boldsymbol{\zeta}, \boldsymbol{\xi}) = \max\ & 2\kappa \sum_{i\in[M]} s_{ii} \\ \text{s.t.}\ & \sum_{i\in[M]} s_{ij} = \xi_j \mathbf{1}^\top \boldsymbol{\zeta}\ \ \forall j \in [M] \\ & \sum_{j\in[M]} s_{ij} = \zeta_i \qquad \forall i \in [M] \\ & \boldsymbol{s} \geq \mathbf{0}.\end{aligned}$$

It is easy to verify that

$$\bar{\rho}(\boldsymbol{\zeta}, \boldsymbol{\xi}) = 2\kappa \sum_{i\in[M]} \min\{\zeta_i, \xi_i(\mathbf{1}^\top \boldsymbol{\zeta})\},$$

as the optimal solution s^*_{ii} must be obtained either at $\xi_i(\mathbf{1}^\top \boldsymbol{\zeta})$ or ζ_i, depending on which one is smaller in order to be feasible. For example, say $\zeta_i < \xi_i(\mathbf{1}^\top \boldsymbol{\zeta})$, then $s^*_{ii} = \zeta_i$ and $s^*_{ij} = 0$ for $j \neq i$.

By the definition of d_{ij}, the objective in $\rho(\boldsymbol{\zeta}, \boldsymbol{\xi})$ becomes

$$2\kappa \sum_{i,j\in[M]:i\neq j} s_{ij},$$

and further notice that

$$\sum_{i\in[M]} \sum_{j\in[M]} s_{ij} = \mathbf{1}^\top \boldsymbol{\zeta},$$

this implies

$$\rho(\boldsymbol{\zeta}, \boldsymbol{\xi}) = 2\kappa \left[\mathbf{1}^\top \boldsymbol{\zeta} - \bar{\rho}(\boldsymbol{\zeta}, \boldsymbol{\xi})\right].$$

We can further rewrite it by

$$\begin{aligned}\mathbf{1}^\top \boldsymbol{\zeta} - \bar{\rho}(\boldsymbol{\zeta}, \boldsymbol{\xi}) &= \mathbf{1}^\top \boldsymbol{\zeta} - \sum_{i\in[M]} \min\{\zeta_i, \xi_i(\mathbf{1}^\top \boldsymbol{\zeta})\} \\ &= \mathbf{1}^\top \boldsymbol{\zeta} - \sum_{i\in[M]:\zeta_i<\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \zeta_i - \sum_{i\in[M]:\zeta_i\geq\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \xi_i(\mathbf{1}^\top \boldsymbol{\zeta}) \\ &= \sum_{i\in[M]:\zeta_i\geq\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \zeta_i - \sum_{i\in[M]:\zeta_i\geq\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \xi_i(\mathbf{1}^\top \boldsymbol{\zeta})\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i\in[M]:\zeta_i\geq\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \left(\zeta_i - \xi_i(\mathbf{1}^\top\boldsymbol{\zeta})\right) \\
&= \frac{1}{2}\|\boldsymbol{\zeta} - \boldsymbol{\xi}\cdot(\mathbf{1}^\top\boldsymbol{\zeta})\|_1,
\end{aligned}
$$

where the first equation holds by the representation of $\bar{\rho}(\boldsymbol{\zeta},\boldsymbol{\xi})$, the second to the fourth equations are by simple algebra, and the last equation holds since $\mathbf{1}^\top\boldsymbol{\xi} = 1$ implies

$$
\sum_{i\in[M]:\zeta_i\geq\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \left(\zeta_i - \xi_i(\mathbf{1}^\top\boldsymbol{\zeta})\right) = \sum_{i\in[M]:\zeta_i<\xi_i(\mathbf{1}^\top\boldsymbol{\zeta})} \left(\xi_i(\mathbf{1}^\top\boldsymbol{\zeta}) - \zeta_i\right).
$$

Therefore,

$$
\rho(\boldsymbol{\zeta},\boldsymbol{\xi}) = 2\kappa\left[\mathbf{1}^\top\boldsymbol{\zeta} - \bar{\rho}(\boldsymbol{\zeta},\boldsymbol{\xi})\right] = \kappa\|\zeta - \boldsymbol{\xi}\cdot(\mathbf{1}^\top\boldsymbol{\zeta})\|_1.
$$

("$\Leftarrow$") Proof for the necessary condition. Suppose $\rho(\boldsymbol{\zeta},\boldsymbol{\xi}) = \kappa\|\zeta - \boldsymbol{\xi}\cdot(\mathbf{1}^\top\boldsymbol{\zeta})\|_p$ for some $\kappa > 0$ and $p \geq 1$. Consider the case that

$$
\begin{aligned}
&\boldsymbol{\zeta} = (0,\ldots,0,1,0,\ldots,0) && \text{1 at position } i \text{ and 0 others,} \\
&\boldsymbol{\xi} = (0,\ldots,0,\xi_i,0,\ldots,0,\xi_j,0,\ldots,0,\ldots,0,\xi_k,\ldots) && \xi_i,\xi_j,\xi_k \text{ at positions } i,j,k \text{ and 0 others,}
\end{aligned}
$$

where $0 \leq \xi_i,\xi_j,\xi_k \leq 1$ and $\xi_i + \xi_j + \xi_k = 1$. Then we have

$$
\begin{aligned}
\rho(\boldsymbol{\zeta},\boldsymbol{\xi}) &= \left((d_{ii}\xi_i^p) + (d_{ij}\xi_j^p) + (d_{ik}\xi_k^p)\right)^{1/p} \\
&= \kappa\|(1-\xi_i,\xi_j,\xi_k)\|_p \\
&= \kappa\|(\xi_j+\xi_k,\xi_j,\xi_k)\|_p
\end{aligned}
\tag{17}
$$

where the first equation is by the definition of the ℓ_p-based CALM metric, the second is by the condition that $\rho(\boldsymbol{\zeta},\boldsymbol{\xi}) = \kappa\|\boldsymbol{\zeta} - \boldsymbol{\xi}\cdot(\mathbf{1}^\top\boldsymbol{\zeta})\|_p$, and the last one is by simple algebra.

- First, taking $\xi_i = 1,\ \xi_j = \xi_k = 0$, then equations (17) suggests

  $$
  (d_{ii}\xi_i^p)^{1/p} = 0,
  $$

  it follows immediately that $d_{ii} = 0$ for all $i \in [M]$.

- Next, taking only $\xi_k = 0$, then $1 - \xi_i = \xi_j$ and equations (17) suggests

  $$
  (d_{ij}\xi_j^p)^{1/p} = \kappa\|(\xi_j,\xi_j,0)\|_p,
  $$

  Note that $\kappa\|(\xi_j,\xi_j,0)\|_p = \kappa\xi_k\cdot 2^{1/p}$, thus $d_{ij} = 2^{1/p}\kappa$.

- Finally, by the previous results on d_{ij}, equations (17) suggests

$$\begin{aligned} & 2^{1/p}\kappa(\xi_j^p+\xi_k^p)^{1/p} = \kappa\|(\xi_j+\xi_k,\xi_j,\xi_k)\|_p \\ \Longleftrightarrow \quad & 2(\xi_j^p+\xi_k^p) = (\xi_j+\xi_k)^p+\xi_j^p+\xi_k^p \\ \Longleftrightarrow \quad & \xi_j^p+\xi_k^p = (\xi_j+\xi_k)^p \end{aligned}$$

  for any $\xi_j,\xi_k\in[0,1]$ and $\xi_j+\xi_k\le 1$, which holds if and only if $p=1$. In the final step we utilize the fact that $M\ge 3$ as we need at lease three nonzero components in $\boldsymbol{\xi}$, while in the first two steps we only need two nonzero components thus the results still hold even if $M=2$.

(ii) We now prove the case when $M=2$. In this case, the linear programming in the definition of the ℓ_p-based CALM metric can be simplified significantly. We can set $\boldsymbol{\zeta}=(\zeta,1-\zeta)$ and $\boldsymbol{\xi}=(\xi,1-\xi)$ for any $\zeta,\xi\in[0,1]$ since without loss of generality we can normalize $\boldsymbol{\zeta}$ by the positive homogeneous property and we can also assume $\zeta\le\xi$. It follows that the feasible solution of $\boldsymbol{s}$ can be represented by

$$\boldsymbol{s}=\begin{bmatrix} \zeta-t & , & t \\ \xi-\zeta+t & , & 1-\xi-t \end{bmatrix},$$

where t is a decision variable, thus

$$\begin{aligned} \rho(\boldsymbol{\zeta},\boldsymbol{\xi}) = \min_t\ & ((d_{11}(\zeta-t))^p+(d_{22}(1-\xi-t))^p+(d_{12}t)^p+(d_{21}(\xi-\zeta+t))^p)^{1/p} \\ \text{s.t.}\ & t\ge 0 \\ & \zeta-t\ge 0 \\ & 1-\xi-t\ge 0. \end{aligned} \tag{18}$$

("⇒") Proof for the sufficient condition. Suppose $d_{ii}=0$ and $d_{ij}=2^{1/p}\kappa$, then Problem (18) becomes

$$\begin{aligned} \rho(\boldsymbol{\zeta},\boldsymbol{\xi}) = \min_t\ & ((d_{12}t)^p+(d_{21}(\xi-\zeta+t))^p)^{1/p} \\ \text{s.t.}\ & t\ge 0 \\ & \zeta-t\ge 0 \\ & 1-\xi-t\ge 0, \end{aligned}$$

which clearly obtains the optimality when $t=0$, thus $\rho(\boldsymbol{\zeta},\boldsymbol{\xi})=d_{21}|\xi-\zeta|=2^{1/p}\kappa|\xi-\zeta|$. Further notice that $\|\boldsymbol{\zeta}-\boldsymbol{\xi}\|_p=|\xi-\zeta|\cdot 2^{1/p}$ completes the proof of this part.

("⇐") Proof for the necessary condition. This is straightforward by directly employing the first two steps in the proof of the necessary condition (i "⇐") when $M\ge 3$.

□

Proof of Proposition 1. Notice that Problem (8) and Problem (11) share the same constraints except for an additional routing distance budget constraint $\sum_{k\in[K]}\mu(\boldsymbol{y}_k)\le\hat{Z}(1+\alpha)$ in Problem (11). Note that $\hat{Z}$ is the optimal object of Problem (8), that is, the minimum value of total routing distance. Therefore, when $\alpha\ge 0$, any optimal solution to Problem (8) must be a feasible solution to Problem (11).

When $\alpha=0$, we claim that the budget constraint $\sum_{k\in[K]}\mu(\boldsymbol{y}_k)\le\hat{Z}$ in Problem (11) must be tight. Assuming it is not tight at the optimality of to Problem (11), then its optimal solution is also feasible for Problem (8) with an objective value strictly below $\hat{Z}$, which leads to a contradiction. Therefore, the optimal solution to Problem (11) is also optimal to Problem (8) when $\alpha=0$.

□

Proof of Proposition 2. By the definition of dual norm, we have

$$\gamma_k\|\boldsymbol{w}-\hat{\boldsymbol{w}}\|=\sup_{\|z\|_*\le\gamma_k} z^\top(\boldsymbol{w}-\hat{\boldsymbol{w}}). \tag{19}$$

Notice that the first line of constraints in Problem (12) is equivalent to

$$\sup_{\boldsymbol{w}\in\mathbb{R}^{L+2}}\{g(\boldsymbol{w}_L,\boldsymbol{x}_k)+w_{L+1}\mu(\boldsymbol{y}_k)+w_{L+2}\rho_k(\boldsymbol{x}_k)-\gamma_k\|\boldsymbol{w}-\hat{\boldsymbol{w}}\|\}\le T_k\ \ \forall k\in[K].$$

This, together with Equation (19), further indicates that

$$\begin{aligned}
&\sup_{\boldsymbol{w}\in\mathbb{R}^{L+2}}\{g(\boldsymbol{w}_L,\boldsymbol{x}_k)+w_{L+1}\mu(\boldsymbol{y}_k)+w_{L+2}\rho_k(\boldsymbol{x}_k)-\gamma_k\|\boldsymbol{w}-\hat{\boldsymbol{w}}\|\}\\
=&\sup_{\boldsymbol{w}\in\mathbb{R}^{L+2}}\inf_{\|z\|_*\le\gamma_k}\left\{g(\boldsymbol{w}_L,\boldsymbol{x}_k)+w_{L+1}\mu(\boldsymbol{y}_k)+w_{L+2}\rho_k(\boldsymbol{x}_k)-z^\top(\boldsymbol{w}-\hat{\boldsymbol{w}})\right\}\\
=&\sup_{\boldsymbol{w}\in\mathbb{R}^{L+2}}\inf_{\|z\|_*\le\gamma_k}\left\{\sum_{\ell\in[L]}w_\ell(\delta_\ell(\boldsymbol{x}_k)-z_\ell)+w_{L+1}(\mu(\boldsymbol{y}_k)-z_{L+1})+w_{L+2}(\rho_k(\boldsymbol{x}_k)-z_{L+2})+z^\top\hat{\boldsymbol{w}}\right\}\\
=&\inf_{\|z\|_*\le\gamma_k}\sup_{\boldsymbol{w}\in\mathbb{R}^{L+2}}\left\{\sum_{\ell\in[L]}w_\ell(\delta_\ell(\boldsymbol{x}_k)-z_\ell)+w_{L+1}(\mu(\boldsymbol{y}_k)-z_{L+1})+w_{L+2}(\rho_k(\boldsymbol{x}_k)-z_{L+2})+z^\top\hat{\boldsymbol{w}}\right\},\\
=&\begin{cases}\inf\limits_{\|z\|_*\le\gamma_k} z^\top\hat{\boldsymbol{w}}\ , & \text{if } (\delta_1(\boldsymbol{x}_k),\cdots,\delta_\ell(\boldsymbol{x}_k))=\boldsymbol{z},\mu(\boldsymbol{y}_k)=z_{L+1},\rho_k(\boldsymbol{x}_k)=z_{L+2}\\ \infty\ , & \text{otherwise}\end{cases}\\
=&\begin{cases}g(\hat{\boldsymbol{w}}_L,\boldsymbol{x}_k)+\hat{w}_{L+1}\mu(\boldsymbol{y}_k)+\hat{w}_{L+2}\rho_k(\boldsymbol{x}_k)\ , & \text{if } ||(\delta_1(\boldsymbol{x}_k),\cdots,\delta_L(\boldsymbol{x}_k),\mu(\boldsymbol{y}_k),\rho_k(\boldsymbol{x}_k))||_*\le\gamma_k\\ \infty\ , & \text{otherwise,}\end{cases}
\end{aligned}$$

where the third equality follows from the Sion's max-min theory of convex-concave functions (Sion 1958). Therefore, the first line of constraints in Problem (12) is equivalent to:

$$\begin{cases}g(\hat{\boldsymbol{w}}_L,\boldsymbol{x}_k)+\hat{w}_{L+1}\mu(\boldsymbol{y}_k)+\hat{w}_{L+2}\rho_k(\boldsymbol{x}_k)\le T_k & \forall k\in[K]\\ ||(\delta_1(\boldsymbol{x}_k),\cdots,\delta_L(\boldsymbol{x}_k),\mu(\boldsymbol{y}_k),\rho_k(\boldsymbol{x}_k))||_*\le\gamma_k & \forall k\in[K].\end{cases}$$

This gives the exact reformulation (13) of Problem (12).

□

Proof of Theorem 4. Notice $\boldsymbol{\delta}(\boldsymbol{x}_k), \mu(\boldsymbol{y}_k), \rho_k(\boldsymbol{x}_k)$ are non-negative and $\sum_{\ell\in[L]} \delta_\ell(\boldsymbol{x}_k) = 1$, for all $\forall k \in [K]$, and further note that the dual norm of ℓ_∞-norm is just ℓ_1-norm, thus the second line of constraints in Problem (12) can be written as

$$\begin{cases} g(\hat{\boldsymbol{w}}_L, \boldsymbol{x}_k) + \hat{w}_{L+1}\mu(\boldsymbol{y}_k) + \hat{w}_{L+2}\rho_k(\boldsymbol{x}_k) \le T_k & \forall k \in [K] \\ 1 + \mu(\boldsymbol{y}_k) + \rho_k(\boldsymbol{x}_k) \le \gamma_k & \forall k \in [K]. \end{cases}$$

By invoking Proposition 2 directly, we can derive the following equivalence of Problem (12):

$$\begin{aligned} \Gamma_\alpha = \min \ & \sum_{k\in[K]} (1 + \mu(\boldsymbol{y}_k) + \rho_k(\boldsymbol{x}_k)) \\ \text{s.t.} \ & g(\hat{\boldsymbol{w}}_L, \boldsymbol{x}_k) + \hat{w}_{L+1}\mu(\boldsymbol{y}_k) + \hat{w}_{L+2}\rho_k(\boldsymbol{x}_k) \le T_k \quad \forall k \in [K] \\ & \sum_{k\in[K]} \mu(\boldsymbol{y}_k) \le \hat{Z}(1+\alpha) \\ & \boldsymbol{x} \in \mathcal{X}. \end{aligned} \tag{20}$$

Consequently, we have

$$\Gamma_\alpha = \Gamma + \sum_{k\in[K]} \mu(\boldsymbol{y}_k^*) + \sum_{k\in[K]} \rho_k(\boldsymbol{x}_k^*).$$

Since $(\boldsymbol{x}_k^\dagger, \boldsymbol{y}_k^\dagger)_{k\in[K]}$ is feasible to Problem (12), then $\Gamma_\alpha \le \bar{\Gamma}_\alpha$. We also have

$$\begin{aligned} \bar{\Gamma}_\alpha &= K + \sum_{k\in[K]} \mu(\boldsymbol{y}_k^\dagger) + \sum_{k\in[K]} \rho_k(\boldsymbol{x}_k^\dagger) \\ &\le K + \sum_{k\in[K]} \mu(\boldsymbol{y}_k^\dagger) + \sum_{k\in[K]} \rho_k(\boldsymbol{x}_k^*) \\ &= \Gamma_\alpha + \sum_{k\in[K]} \mu(\boldsymbol{y}_k^\dagger) - \sum_{k\in[K]} \mu(\boldsymbol{y}_k^*) \\ &\le \Gamma_\alpha + \left(\hat{Z}(1+\alpha) - \sum_{k\in[K]} \mu(\boldsymbol{y}_k^*) \right). \end{aligned}$$

When the routing distance budget constraint is tight, employing the above results we have

$$\bar{\Gamma} = \Gamma,$$

which indicates $(\boldsymbol{x}_k^\dagger, \boldsymbol{y}_k^\dagger)_{k\in[K]}$ is also an optimal solution to Problem (12), thus Problem (11) solves Problem (12) whenever the routing distance budget constraint is tight. □

Appendix B Connection Between CALM and the Optimal Transport Problem

The ℓ_p-based CALM is closely related to the Optimal Transport (OT) problem in its Kantorovich formulation (Kantorovich and Rubinshtein 1958). The OT problem seeks the most cost-efficient way to transform one probability distribution into another while considering a predefined cost function. Specifically, given two distributions μ and ν over a space $\mathcal{X}$, and a cost function $d(\boldsymbol{x}, \boldsymbol{y})$ representing the cost of transporting one unit of mass from $\boldsymbol{x}$ to $\boldsymbol{y}$, the OT problem is formulated as:

$$\min_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d(\boldsymbol{x}, \boldsymbol{y})\, d\pi(\boldsymbol{x}, \boldsymbol{y}),$$

where $\Pi(\mu, \nu)$ is the set of all joint distributions of μ and ν, defined as:

$$\Pi(\mu, \nu) = \left\{ \pi \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) : \int_{\mathcal{X}} d\pi(\boldsymbol{x}, \boldsymbol{y}) = d\mu(x),\ \int_{\mathcal{X}} d\pi(\boldsymbol{x}, \boldsymbol{y}) = d\nu(\boldsymbol{y}) \right\}.$$

In the context of the CALM metric, the normalized assigned workload vector $\boldsymbol{\zeta}/(\mathbf{1}^\top \boldsymbol{\zeta})$ is analogous to μ, and the prior assigned location vector $\boldsymbol{\xi}$ to ν, with both treated as discrete probability distributions. The CALM metric, when normalized by dividing by $\mathbf{1}^\top \boldsymbol{\zeta}$, functions as a simplified version of the OT problem, modeling the redistribution of mass from one discrete distribution to another.

The connection to the OT problem provides valuable insights. First, CALM leverages key concepts and mathematical properties of OT while significantly providing a computational efficient option. Second, by customizing d_{ij} to reflect specific costs in different settings, CALM can be adapted to a wide range of real-world logistics scenarios. This connection provides a deeper theoretical foundation for CALM while broadening its applicability to a range of real-world scenarios. It is worthy mentioning that the applications of OT span various domains, including computer vision (Rubner et al. 2000), generative artificial intelligence (Arjovsky et al. 2017), and data-driven optimization (Mohajerin Esfahani and Kuhn 2018).

Appendix C Descriptions of Algorithms

In this section, we provide the details of the exact and heuristics algorithms, that is, the Benders decomposition and Tabu Search algorithms.

Benders Decomposition

Benders decomposition (Rahmaniani et al. 2017) solves an optimization problem by dividing it into a first-stage master problem and a second-stage subproblem. The master problem involves fewer variables and constraints compared to the original problem, and is inherently simpler to solve. Initially, the master problem is solved, and following that, a subproblem is formulated. This subproblem is derived by fixing the variables from the original problem based on the solution of the master problem and is then solved. Based on the dual optimal solution of the subproblem, a Benders cut is generated and incorporated into the master problem, which is then resolved. This iterative process continues until an optimal solution is reached. Our algorithm adheres to the standard Benders decomposition framework but involves a set of problem-specific Benders inequalities stemming from the deadline constraints, which are also the key distinctions between our problem and the classic routing problems. Since Problems (16), (8) and (11) are analogous, with Problem (11) being the most complex, we use Problem (11) as the representative case to demonstrate our Benders decomposition, which is detailed in Algorithm 1.

The master problem for Problem (11) is as follows:

$$\begin{aligned}
\min \ & \sum_{k\in[K]} \lambda_k \\
\text{s.t.} \ & g(\hat{\boldsymbol{w}}, \boldsymbol{x}_k) + \hat{w}_{L+1}\mu_k(\boldsymbol{x}_k) + \hat{w}_{L+2}\lambda_k \le T_k \quad \forall k \in [K] \\
& \mu(\boldsymbol{y}_k) \le \hat{Z}(1+\alpha) \\
& \lambda_k \ge 0 \qquad\qquad\qquad\qquad\qquad\qquad \forall k \in [K] \\
& (\boldsymbol{x}_k, \boldsymbol{y}_k)_{k\in[K]} \in \mathcal{X}^\dagger,
\end{aligned} \tag{21}$$

where λ_k is a new non-negative continuous variable to replace optimization problem $\rho_k(\boldsymbol{x}_k)$ and

$$\mathcal{X}^\dagger = \left\{ (\boldsymbol{x}_k, \boldsymbol{y}_k)_{k\in[K]} \;\middle|\; \begin{array}{ll}
\displaystyle\sum_{k\in[K]} x_{kn} = 1 & \forall n \in [N] \\
\displaystyle\sum_{a\in\mathcal{A}^+(\{0\})} y_{ka} = \sum_{a\in\mathcal{A}^-(\{0\})} y_{ka} = 1, & \forall k \in [K] \\
\displaystyle\sum_{a\in\mathcal{A}^+(\{n\})} y_{ka} = \sum_{a\in\mathcal{A}^-(\{n\})} y_{ka} = x_{kn} & \forall n \in [N], k \in [K] \\
\displaystyle\sum_{n\in[N]} q_n x_{kn} \le Q & \forall k \in [K] \\
\boldsymbol{x}_k \in \{0,1\}^N, \boldsymbol{y}_k \in \{0,1\}^{|\mathcal{A}|} & \forall k \in [K]
\end{array} \right\}.$$

Hence, the set $\mathcal{X}^\dagger$ excludes the exponential subtour elimination constraints from the set $\mathcal{X}$. With the removal, Problem (21) is compact and can be directed solved by general MIP solver such as CPLEX and Gurobi.

Given an optimal solution of the master problem, denoted by $(\boldsymbol{x}_k^\dagger, \boldsymbol{y}_k^\dagger, \boldsymbol{\lambda}_k^\dagger)_{k\in[K]}$, we first check whether there exist any subtours in this optimal solution. Suppose there exists a subtour for the nodes in $S^\dagger$, the following subtour elimination constraints are added to the master problem:

$$\sum_{a\in\mathcal{A}^+(S^\dagger)} y_{ka} \geq x_{kn} \quad \forall n \in S^\dagger, k \in [K]. \tag{22}$$

Once there exist no subtours in the master problem's solution, problem $\rho_k(\boldsymbol{x}_k^\dagger)$ is solved to obtain its dual optimal solution, denoted by $(\boldsymbol{u}_k^\dagger, \boldsymbol{v}_k^\dagger)$. If $\lambda_k^\dagger < \rho_k(\boldsymbol{x}_k^\dagger)$, by using the dual problem of ρ_k, the following Benders cut is incorporated into the master problem:

$$\lambda_k \geq \sum_{i\in[M]} u_i^\dagger \boldsymbol{h}_k^\top \boldsymbol{x}_k + \sum_{j\in[M]} v_j^\dagger \hat{\xi}_{kj} \mathbf{1}^\top \boldsymbol{x}_k. \tag{23}$$

To address Problem (8), the only requirement is to substitute the objective from Problem (11) with that of Problem (8), while simultaneously omitting the budget constraint $\sum_{k\in[K]} \mu(\boldsymbol{y}_k) \leq \hat{Z}(1+\alpha)$. In a similar vein, solving Problem (16) necessitates altering the objective as per Problem (16) and disregarding Benders cut (23).

Algorithm 1 *Benders Decomposition*

Initialize $\boldsymbol{x}^* \leftarrow \emptyset$ as an optimal solution;
while True **do**
  Solve master problem (21) and obtain an optimal solution $(\boldsymbol{x}^\dagger, \boldsymbol{y}^\dagger, \boldsymbol{\lambda}^\dagger)$;
  if there exists a subtour in solution $\bar{\boldsymbol{x}}$ **then**
    Add inequalities (22) to master problem (21);
  else
    Solve subproblem $\rho_k(\boldsymbol{x}_k^\dagger)$ for each $k \in [K]$;
    if there exists a courier $k \in [K]$ such that $\lambda_k^\dagger < \rho_k(\boldsymbol{x}_k^\dagger)$ **then**
      Add inequalities (23) to master problem (21);
    else
      Set $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^\dagger$;
      Break;
    end if
  end if
end while
Return $\boldsymbol{x}^*$;

Tabu Search

Tabu Search is an algorithm that incrementally improves an initial solution using various local search operators. In each iteration, the algorithm explores the neighborhoods defined by the local search operators, searching for the best solution to move on. A neighborhood of an operator is defined as the set of feasible solutions that can be generated from the incumbent solution though an operation imposed by the operator. In order to escape from local optimum, the algorithm may accept a solution worse than the incumbent solution during the search. However, to prevent the search process from cycling, after an operation is performed, the inverse operation is prohibited for a series of consecutive iterations unless this inverse operation yields a new best solution. A solution in the neighborhood is called admissible if it is a new best solution or not generated by the forbidden operations. Additionally, if the current best solution cannot be improved over a certain number of iterations, the algorithm invokes a *shake* procedure to randomly modify the incumbent solution for a predetermined number of iterations. The Tabu Search stops when it reaches a predefined limit, either in terms of the number of iterations or the elapsed execution time. Cordeau et al. (2001) provide a detailed description of Tabu Search.

Our Tabu Search procedure is outlined in Algorithm 2, where $\boldsymbol{s}$ is the initial solution, $\boldsymbol{s}^*$ is the best solution, $c(\boldsymbol{s})$ is the distance of solution $\boldsymbol{s}$, *maxIter* and *nonImproveIter* are two control parameters which determine the maximum number of iterations for the algorithm and the tenure to invoke the *shake* procedure, respectively. In our implementation, we incorporate two simple and most widely-used local search operators in the the literature, namely *relocation* and *exchange*, to explore solution's neighborhood. The *relocation* operator moves a customer from its incumbent location to a new location. The *exchange* operator selects two customers and swaps their locations. In our approach, we have chosen not to implement more complex local search operators because verifying the solution feasibility and computing the solution distance in Problem (11) are time-consuming. Consequently, complex operators will demand considerable computational time for neighborhood search.

The Tabu Search necessitates an initial solution to start the search process. However, as per our preliminary computational experiments, classical greedy algorithms, such as Clarke and Wright savings algorithm (Clarke and Wright 1964), easily fail in generating feasible initial solutions when the deadline constraints are stringent. Therefore, we design a specific algorithm, detailed in Algorithm 3, to generate initial solutions. First, K empty routes is constructed as the incumbent solution. Then a single customer or a pair of customers is repeatedly inserted into the incumbent

Algorithm 2 *Tabu Search($\boldsymbol{s}$)*

Set $\boldsymbol{s}^* \leftarrow \boldsymbol{s}$, $i \leftarrow 0$, and $j \leftarrow 0$
while $i <$ maxIter **do**
  Search for the best admissible solution $\boldsymbol{s}'$ and set $\boldsymbol{s} \leftarrow \boldsymbol{s}'$;
  if $c(\boldsymbol{s}) < c(\boldsymbol{s}^*)$ **then**
    Set $\boldsymbol{s}^* \leftarrow \boldsymbol{s}$;
  else
    Set $j \leftarrow j + 1$;
  end if
  if $j =$ nonImproveIter **then**
    Set $\boldsymbol{s} \leftarrow shake(\boldsymbol{s})$ and $j \leftarrow 0$;
  end if
  Set $i \leftarrow i + 1$;
end while
Return $\boldsymbol{s}^*$;

solution. The selection of customers and their insertion places is guided by the criterion of minimal distance increment to the incumbent solution. This insertion process continues until all customers are accommodated or no further feasible insertions can be made. In scenarios where insertions become infeasible, the incumbent solution is optimized by the Tabu Search mentioned above. If the optimization creates new feasible insertion places for the customers who have not yet been served, the insertion process resumes. Otherwise, dynamic programming (24) is invoked to determined the number of customers for each courier $k \in [K]$, since the number of customers served by a courier has a significant impact on the delivery time. Let $\hat{\ell}_k(\boldsymbol{s})$ be the number of customers served by courier $k \in K$ in solution $\boldsymbol{s}$, $\hat{T}_k(\boldsymbol{s}) = T_k - \hat{w}_{L+1}\mu_k(\boldsymbol{s}_k) - \hat{w}_{L+2}\rho_k(\boldsymbol{s}_k)$, and $\phi_k(\ell)$ be the minimum summation of the workload indices' coefficients for the first $k = 1, \ldots, K$ couriers while the total number of customers is equal to $\ell = 1, \ldots, N$. Dynamic programming (24) aims to minimize the summation of the workload indices' coefficients by properly allocating the customers who have not been served yet to each courier. Let $\boldsymbol{\ell}^*(\boldsymbol{s})$ be an optimal solution of dynamic programming (24), *i.e.*, $\ell_k^*(\boldsymbol{s})$ is the number of customers assigned to courier k in the optimal solution. Then, $\delta_{\ell_k^*(\boldsymbol{s})}$ is fixed to 1, even though the real number of customers served by courier $k \in [K]$ in the incumbent solution may be still smaller than $\ell_k^*(\boldsymbol{s})$. If a route is infeasible after $\delta_{\ell_k^*(\boldsymbol{s})}$ is fixed, the last customer in this route is repeatedly removed until feasibility is restored. The incumbent solution undergoes another round of optimization through Tabu Search. Depending on the emergence of new feasible insertion

places post-optimization, the algorithm either continues the insertion process or concludes with an infeasible solution if no such places exist.

$$\phi_k(\ell) = \begin{cases} \hat{w}_\ell, & k = 1, \ell \geq \hat{\ell}_k(\boldsymbol{s}), \hat{w}_\ell \leq \hat{T}_k(\boldsymbol{s}) \\ \min_{h \geq \hat{\ell}_k(\boldsymbol{s}), \hat{w}_h \leq \hat{T}_k(\boldsymbol{s})} \{\phi_{k-1}(\ell - h) + \hat{w}_h\}, & k = 2, \ldots, K \\ \infty, & \text{otherwise.} \end{cases} \tag{24}$$

Algorithm 3 *Initial Solution*

Construct a solution s with K empty routes;
Set $U \leftarrow N$ and state $\leftarrow 0$;
while $U \neq \emptyset$ **do**
  Search for a feasible insertion which minimizes the distance increment to s;
  if such an insertion exits **then**
    Insert the selected customers to $\boldsymbol{s}$ and remove them from U;
    Set state $\leftarrow 0$;
  else if state = 0 **then**
    Set $\boldsymbol{s} \leftarrow TabuSearch(\boldsymbol{s})$ and state $\leftarrow 1$;
  else if state = 1 **then**
    Invoke dynamic programming (24);
    Set $\delta_{\ell_k^*(\boldsymbol{s})} \leftarrow 1 \; \forall k \in [K]$;
    for $k \in [K]$ **do**
      while the route of courier k is infeasible **do**
        Remove the last customer in the route and add it to U;
      end while
    end for
    Set $\boldsymbol{s} \leftarrow TabuSearch(\boldsymbol{s})$ and state $\leftarrow 2$;
  else
    Break;
  end if
end while
Return $\boldsymbol{s}$;

We finally present the hyperparameters used in our Tabu Search algorithm, which are given in Table 8. *maxIter* is the maximum number of iterations for the Tabu Search. *nonImproveIter* is the number of iterations that the Tabu Search fails to find a better solution so that the Tabu Search invokes the *shake* procedure. *shake tenure* is the tenure for the Tabu Search to invoke the *shake* procedure. *shake size* is the number of iterations that the *shake* procedure takes to perturb a solution.

	Parameter
maxIter	500
nonImproveIter	20
shake tenure	10
shake size	100
tenure reduction rate	0.99

Table 8 Parameters in Tabu Search algorithm.

tenure reduction rate controls the decreasing speed of the Tabu tenure after the *shake* procedure. Each time the *shake* procedure finishes, the Tabu tenure is multiplied by this *tenure reduction rate*.

Appendix D Additional Information for Our Data and Analysis

We provide more information of our data and analysis here. The map of the 35 municipal districts in Dongguan city is illustrated in Figure 5 and the summary statistics of the key variables of our data set are provided in Table 9.

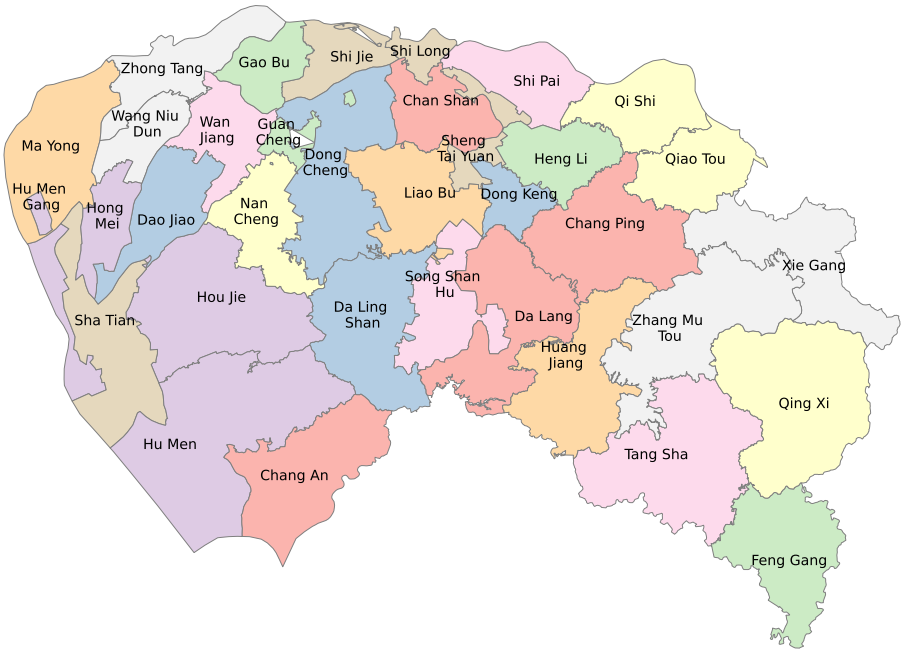

Figure 5 Municipal Districts of Dongguan City.

	Delivery Time (hr)	Customers	Distance (km)	Weights (kg)
count	23,074	23,074	23,074	23,074
mean	2.74	18.40	28.99	542.75
std	1.11	8.86	16.28	304.15
min.	0.02	2	0.06	7.99
25%	1.98	12	19.10	328.12
50%	2.76	17	26.82	498.50
75%	3.47	24	35.73	711.81
max.	10.69	90	349.61	5802.00

Note: std, min., and max. denote standard deviation, minimum, and maximum, respectively.

Table 9 Summary Statistics of Key Variables.

In Section 2, we only show the plot between workload and delivery time. Here, we explore how both workload and distance influence delivery time. To elucidate their relationship, we have included a plot in Figure 6 that examines how delivery time per unit of workload and distance varies with changes in workload. The plot reveals that as workload increases, the delivery time per unit of

workload and distance decreases. This suggests that couriers potentially become more efficient as their workload increases, indicating an improvement in delivery efficiency with higher workloads.

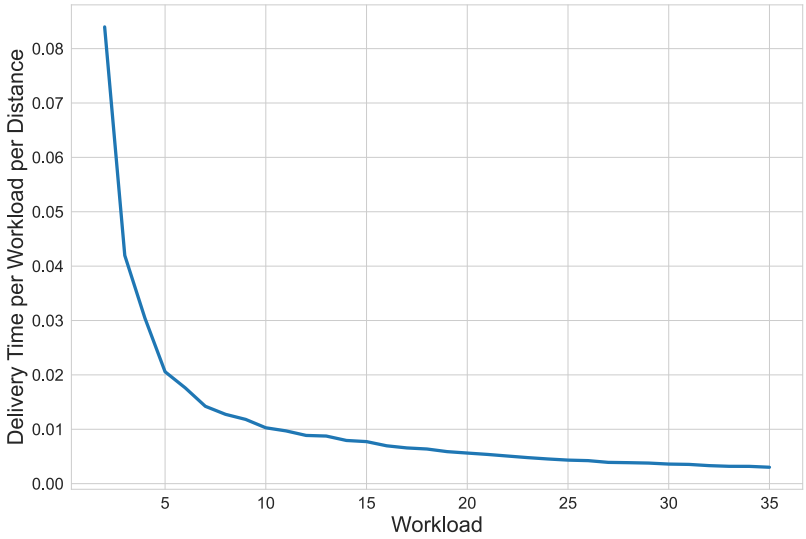

Figure 6 Workload Vs. Delivery time per workload per distance.

Additionally, we present the regression results for Model (3) with the inclusion of CALM in Table 10. The results indicate that CALM remains statistically significant in the model and slightly increases the R^2 value, though the improvement is marginal. This is likely due to the courier fixed effects (FE) already accounting for individual-specific factors in the model.

	Dependent variable: Delivery Time	
	Model (3)	Model (3) including CALM
Const.	-2.291*(0.144)	-2.144*(0.147)
Distance	0.010*(0.002)	0.010*(0.002)
Workload	-0.039*(0.009)	-0.035*(0.009)
Workload$^{1/2}$	1.131* (0.076)	1.063*(0.078)
CALM		0.022*(0.005)
Couriers' FE	✓	✓
Time FE	✓	✓
Observations	23,074	23,074
R^2	0.714	0.718
Adjusted R^2	0.712	0.716
Residual Std. Error	0.597	0.593
df.; #coef.	23,927; 146	23,926; 147

Note: *p<0.001. CALM, FE, df., and #coef. denote courier assigned location mismatch metric, fixed effects, degrees of freedom, and number of coefficients, respectively. Robust standard errors clustered at the level of the courier in parentheses.

Table 10 Regression Results of Model (3) with/without CALM.

We have also included prediction and regression analyses using map-based paths to compute travel distances in Table 11 and Table 12. Compared to the results in Table 1 through Table 4, we find that when distance is the only covariate, models using map-based distances outperform those using Euclidean distances. However, once additional covariates such as workload and the CALM metric are included, the performance difference becomes minimal (approximately 0.001 in terms of R^2 scores). This indicates that map-based distances provide noticeable improvements only in simpler models. In practice, couriers may not strictly follow map-based routes between locations. Moreover, obtaining map-based distances incurs significant time and monetary costs, especially when routing involves a quadratic number of location pairs via API calls. Therefore, we find Euclidean distances to be a far more cost-effective alternative. Moreover, the downstream robust optimization can further help mitigate the uncertainty introduced by such distance proxies and lead to better routing decisions.

	Dependent variable: Delivery Time			
	Model (1)	Model (2)	Model (3)	Model (6)
Const.	✓	✓	✓	✓
Distance (map-based)	✓	✓	✓	✓
Workload		✓	✓	✓
Workload$^{1/2}$		✓	✓	✓
Couriers' FE			✓	
Time FE	✓	✓	✓	✓
CALM				✓
Train R^2	0.344	0.609	0.698	0.621
Train MSE	0.727	0.430	0.330	0.416
AIC	33,232	26,229	22,789	25,801
BIC	33,372	26,384	23,545	25,964
Test R^2	0.082	0.518	0.627	0.541
Test MSE	0.973	0.514	0.396	0.488

Note: FE, MSE, AIC, and BIC denote fixed effects, mean squared error, Akaike information criterion, and Bayesian information criterion, respectively.

Table 11 Prediction Results of Models using map-based paths.

	Dependent variable: Delivery Time			
	Model (1)	Model (2)	Model (3)	Model (6)
Const	1.561*(0.082)	-2.003*(0.135)	-2.240*(0.140)	-1.848*(0.140)
Distance (map-based)	0.0198*(0.001)	0.006*(0.001)	0.006*(0.001)	0.006*(0.001)
Workload		-0.084*(0.010)	-0.044*(0.009)	-0.071*(0.010)
Workload$^{1/2}$		1.380*(0.079)	1.141* (0.069)	1.242*(0.086)
Couriers' FE			✓	
Time FE	✓	✓	✓	✓
CALM				0.036*(0.006)
Observations	23,074	23,074	23,074	23,074
R^2	0.329	0.613	0.713	0.632
Adjusted R^2	0.329	0.613	0.711	0.632
Residual Std. Err.	0.912	0.693	0.598	0.676
df.; #coef.	23,055; 18	23,053; 20	22,927; 146	23,054; 19

Note: *p<0.001. FE, df., and #coef. denote fixed effects, degrees of freedom, and number of coefficients, respectively. Standard errors in parentheses are robust and clustered at the courier level.

Table 12 Regression Results of Models using map-based paths.

Appendix E Additional Results and Details of the Experiments

We preset additional results and details of the experiments in this section.

Hyperparameter Calibration of Robust Satisficing

In each experimental run, we divide our dataset, which comprises 23,074 routes, into a training set and a testing set in an 80:20 ratio, respectively. During the training phase, we identify covariates linked to actual delivery times and employ Model (6) to estimate the respective coefficients $\hat{w}_1, \cdots, \hat{w}_{L+2}$ with $L = 41$. For the testing phase, we specify a delivery deadline of $T = 2.5$ hours and designate $R = 23,074 \times 0.2 = 4,615$ routes for evaluation. The number of routes selected, S, is determined by the formula:

$$S = \left\lfloor \text{ratio} \times \frac{\textit{number of training routes with delivery time} \leq T}{\textit{total number of training routes}} \times R \right\rfloor,$$

where the ratio is set at 0.5.

Here, we explore how this ratio influences the calibration process. Table 13 presents CPU times for solving the model under various parameter settings, demonstrating that these are negligible. Figures 7 through 9 illustrate that despite variations in the ratio, the optimal choice of α consistently falls between 0.2 and 0.3. Note that Problem (15) is a mixed-integer linear program that minimizes the CALM metric subject to a travel distance budget controlled by α. Although it involves binary decisions, its behavior is analogous to that of its linear programming relaxation, whose optimal objective is convex in α. As the number of delayed routes is closely tied to the CALM metric, this further explains the convex-like trend observed across the figures.

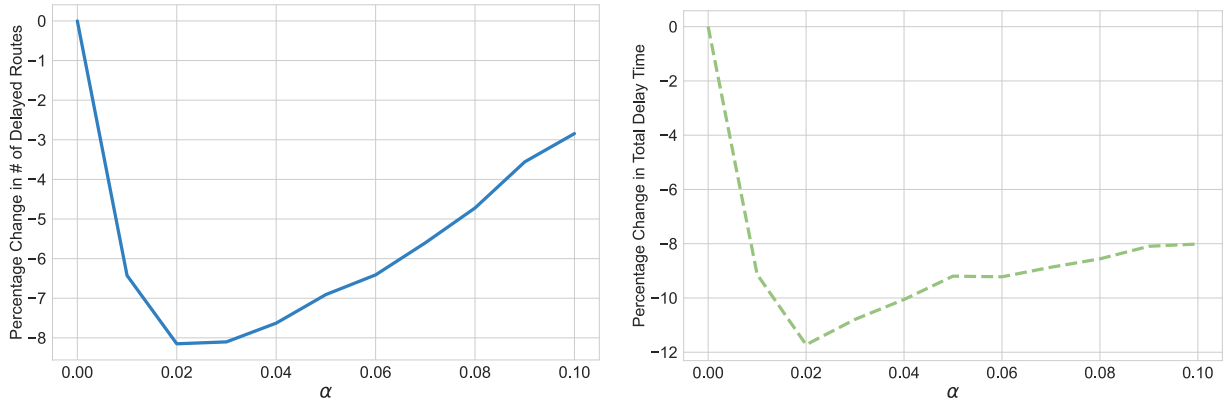

Figure 7 Percentage Change of SOCR Compared to SOR in Two Metrics: the Number of Delayed Routes (left) and the Total Delay Time (right). The ratio is 0.5.

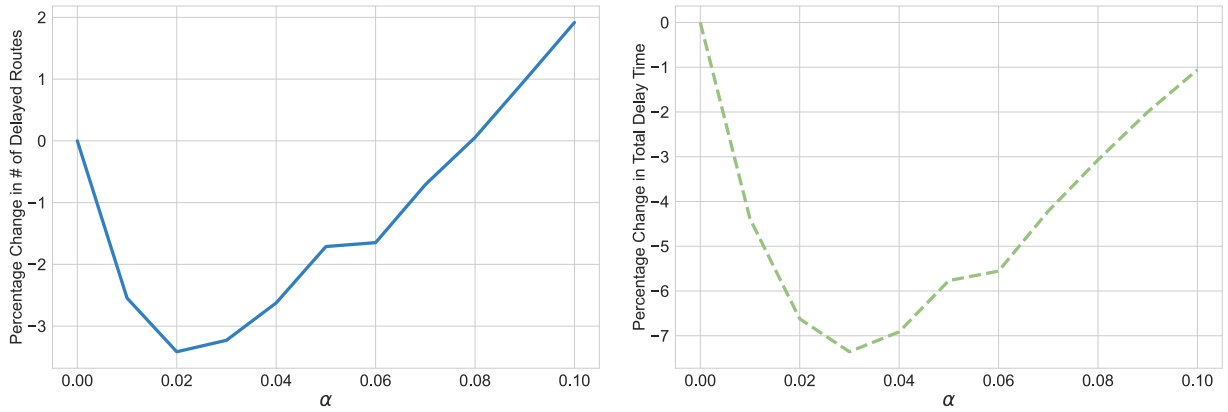

Figure 8 Percentage Change of SOCR Compared to SOR in Two Metrics: the Number of Delayed Routes (left) and the Total Delay Time (right). The ratio is 0.6.

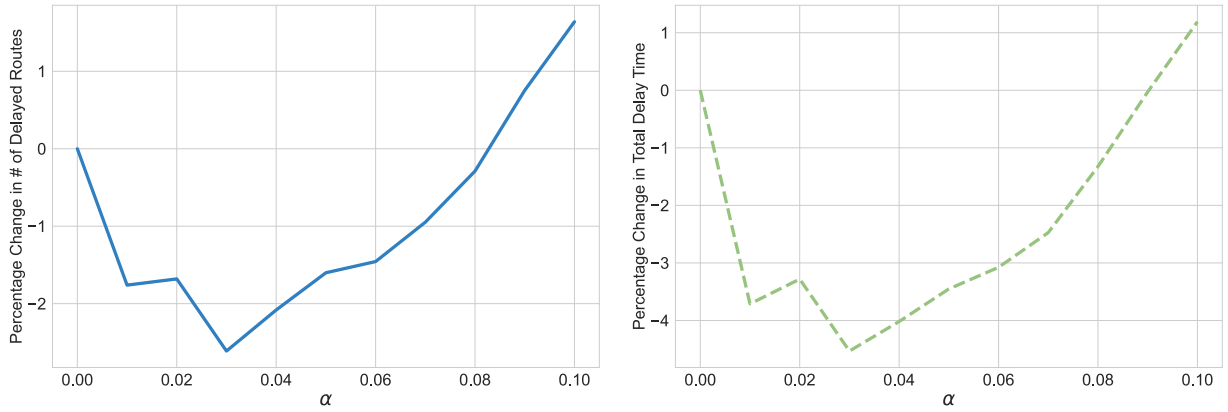

Figure 9 Percentage Change of SOCR Compared to SOR in Two Metrics: the Number of Delayed Routes (left) and the Total Delay Time (right). The ratio is 0.7.

	ratio = 0.5	ratio = 0.6	ratio = 0.7
SOR	0.0071	0.0084	0.0101
SOCR	0.0176	0.0203	0.0236

Table 13 Average CPU time in seconds for solving the SOR and SOCR models, $\alpha = 0.02$.

Instances Sets

Utilizing our real-world dataset, we have categorized two distinct sets of instances as described below. (1)*Medium-size instances:* This category comprises 130 instances, with each instance containing 20-40 nodes (including the depot) served by 2-3 couriers, where every courier is assigned a specific route. (2)*Large-size instances:* In this category, there are 87 instances, each featuring 80-100 nodes (including the depot) and necessitating the collaboration of 4-7 couriers to manage the deliveries.

We have detailed the summary statistics of these instances in Table 14 for medium-size instances and Table 15 for large-size instances. In these tables, "Nodes" represents the number of total nodes including customers and the depot; "Couriers" means the number of couriers in an instance; "Distance" indicates the total travel distance (km) for all routes within an instance; "Time" denotes the total delivery time (hr) for all routes; and "Time Per Route" calculates the average delivery time (hr) per route across all routes in an instance.

	Nodes	Couriers	Distance (km)	Time (hr)	Time Per Route
mean	32.26	2.15	87.32	6.55	3.06
std	4.88	0.36	46.34	1.13	0.37
min	21	2	25.65	5.03	2.51
25%	29	2	59.16	5.75	2.80
50%	33	2	69.90	6.21	3.01
75%	36	2	107.54	6.97	3.22
max	40	3	327.84	10.19	4.97

Table 14 The Summary Statistics of Medium-size Instances.

	Nodes	Couriers	Distance (km)	Time (hr)	Time Per Route
mean	89.70	4.97	185.78	15.88	3.21
std	6.24	0.72	57.42	2.12	0.27
min	80	4	69.50	11.35	2.77
25%	84	4.5	148.90	14.36	3.01
50%	89	5	170.47	15.83	3.20
75%	95	5	205.70	17.22	3.36
max	100	7	353.91	20.70	4.01

Table 15 The summary Statistics of Large-Size Instances.

In each instance, the distance between nodes is calculated using the Euclidean distance between their respective longitudes and latitudes. Due to the absence of specific start and end times at the

depot in our dataset, we employ a standard VRP study workaround by setting the distance between the depot and any node to zero. This approach allows us to focus on the distance among customer nodes while simplifying the initial and final leg of the routes from and to the depot.

Evaluation Process

The detailed evaluation process includes three steps applied to each instance.

1. *Training:* To effectively solve the models, it is essential first to obtain the necessary coefficients. For the VRPD model outlined in Problem (16), we employ Model (1) for coefficient estimation. Similarly, for the SOR and SOCR models described in Problem (8) and Problem (11), coefficients are derived using Model (6) to determine the coefficient vector $\hat{\boldsymbol{w}}$. Throughout the estimation process, we exclude time fixed effects to streamline the number of decision variables, considering their marginal impact on enhancing prediction accuracy. The coefficient estimation for each instance leverages data from the 100 most recent instances prior to the instance's date, ensuring relevance in our analytical approach.
2. *Solving:* Once the coefficients are determined, we proceed to solve the models employing both exact and heuristic algorithms. Off-the-shelf exact solvers like CPLEX prove effective for the majority of medium-size instances, adeptly managing their computational demands. However, exact solution methods encounter difficulties with large-size instances due to their increased complexity. To overcome this, we have crafted heuristic algorithms specifically designed for these challenging larger instances, facilitating their resolution within minutes. For each instance solved, we define $q_n = 1$ and the capacity Q as the lesser value between $L - 1 = 40$ (the upper limit in the generic workload function) and the total number of nodes present in the instance, ensuring a tailored approach to each scenario's unique constraints.
3. *Testing:* The effectiveness of routing decisions made by solving each model is evaluated using various criteria, such as delay rate and delay time. However, accurately predicting a courier's arrival time at each customer node—which is essential for calculating delay rate and delay time—poses a challenge. To address this, we approximate these arrival times by assuming an equal distribution of time across the route. This method offers a more reliable approximation than proportionally dividing delivery time based on travel distance, especially since our previous analysis has shown workload to be a more effective covariate than travel distance for predicting delivery outcomes.

Given the challenge of determining the delivery time for new routes using real-world data, we deploy an 'underlying generation model' to compute the delivery time. This comprehensive linear

regression model integrates couriers' fixed effects, travel distance, a generic workload dependency function (including constant, first-order, second-order, and square root terms of workload), and the CALM metric. This model is more intricate than the one employed in the training phase, thereby addressing potential model misspecification in real-world applications.

For coefficient estimation within this model, we apply a bootstrapping approach, a robust statistical method involving repeated dataset resampling to evaluate the estimator's distribution and confirm the reliability of model parameters (detailed discussions on bootstrapping are available in Freedman 1981). In each bootstrapping iteration, the dataset is randomly split into an 80% training set and a 20% testing set, with the training set treated as a bootstrap sample. The model is trained on this sample and tested for performance on the testing set, with this process being replicated across 200 iterations to build a solid foundation of statistical insights. The underlying generation model showcases a training R^2 of 0.717 and a training mean square error of residuals of 0.59. In testing, it achieves an R^2 of 0.712, underscoring its predictive strength. We utilize this model to generate training samples, including those in the instances sets, by adding a random Gaussian noise with a mean of zero and a standard error of 0.59. When evaluating new solutions derived from the models in testing stage, we apply the model directly without introducing noise, ensuring a clear evaluation of its predictive accuracy.

Hence, our evaluation process closely mirrors real-world scenarios, allowing us not only to test the effectiveness of our solutions but also to directly contrast them with the daily operational choices currently made by the company. Such a comparative analysis provides valuable insights into the potential improvements our models can offer in practical applications.

References

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. *International Conference on Machine Learning*, 214–223 (PMLR).

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12(4):568–581.

Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* 52(8):928–936.

Freedman DA (1981) Bootstrapping regression models. *The Annals of Statistics* 9(6):1218–1228.

Kantorovich LV, Rubinshtein S (1958) On a space of totally additive functions. *Vestnik of the St. Petersburg University: Mathematics* 13(7):52–59.

Mohajerin Esfahani P, Kuhn D (2018) Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *Mathematical Programming* 171(1-2):115–166.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259(3):801–817.

Rubner Y, Tomasi C, Guibas LJ (2000) The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision* 40:99–121.

Sion M (1958) On general minimax theorems. *Pacific J. Math.* 8(4):171–176.