To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–19

A heuristic algorithm for optimal fleet composition with vehicle routing considerations

E. Mardaneh, Q. Lin and R. Loxton*

Department of Mathematics and Statistics, Curtin University, Perth, Australia

(*Received 30 August 2014*)

This paper proposes a fast heuristic algorithm for solving a combined optimal fleet composition and multi-period vehicle routing problem. The aim of the problem is to determine an optimal fleet mix, together with the corresponding vehicle routes, to minimize total cost subject to various customer delivery requirements and vehicle capacity constraints. The total cost includes not only the fixed, variable, and transportation costs associated with operating the fleet, but also the hiring costs incurred whenever vehicle requirements exceed fleet capacity. Although the problem under consideration can be formulated as a mixed-integer linear program (MILP), the MILP formulation for realistic problem instances is too large to solve using standard commercial solvers such as CPLEX. Our proposed heuristic decomposes the problem into two tractable stages: in the first (outer) stage, the vehicle routes are optimized using cross entropy; in the second (inner) stage, the optimal fleet mix corresponding to a fixed set of routes is determined using dynamic programming and golden section search. Numerical results show that this heuristic approach generates high-quality solutions and significantly outperforms CPLEX in terms of computational speed.

Keywords: fleet composition; vehicle routing; cross entropy; dynamic programming; golden section method

AMS Subject Classification: 90B06; 90B10; 90C11; 90C39

1. Introduction

Purchasing a new vehicle fleet is a major commercial expense faced by many companies and organizations. The size and composition of such a fleet must be carefully considered. Indeed, the best fleet mix is often non-homogeneous, with the key problem being to decide how many of each vehicle type to include in the fleet. The optimal fleet mix usually depends heavily on the vehicle routing requirements, although this dependence is rarely discussed or exploited in the optimization literature.

For an introduction to the present state of research in optimal fleet composition and vehicle routing, we direct the reader to two excellent survey papers by Baldacci et al. [2] and Hoff et al. [9]. The survey by Baldacci et al. [2] focuses solely on optimal routing techniques, whereas the survey by Hoff et al. [9] focuses on the simultaneous optimization of fleet composition and vehicle routing, particularly in the maritime transportation industry. Although fleet composition has been studied extensively over multiple decades, several important issues—such as problems with multi-period time horizons and capacitated heterogeneous fleets—have yet to receive adequate attention in the literature [7]. Our paper is focused on addressing this gap. In particular, we consider a combined opti-

*Corresponding author. Email: r.loxton@curtin.edu.au

mal fleet composition and vehicle routing problem over a multi-period planning horizon with vehicle capacity constraints.

The problem under consideration is closely related to the so-called *periodic vehicle routing problem* (PVRP) in which customers must be visited a prescribed number of times (visit frequency) over a specified planning horizon. Allowable combinations of visit periods are typically defined for each customer. For example, a customer could have allowable visit combinations $\{1, 3\}$ and $\{2, 4\}$, meaning that the customer must be visited twice, either during periods 1 and 3 or during periods 2 and 4.

Various heuristics and meta-heuristics have been developed to solve the PVRP. An example is the two-phase heuristic algorithm proposed by Chao et al. [4], which works as follows: in the first phase, a mixed-integer linear program (MILP) is solved to assign visit-day combinations to each of the customers; in the second phase, several improvement operators are applied while the vehicle capacity constraints are relaxed. Bertazzi et al. [3] considered a special case of the PVRP in which a single vehicle is used in each period. They proposed a solution algorithm that works by iteratively selecting an unvisited customer, assigning a valid combination of visit days, and, for each visit day, scheduling the customer in the best position of the current partial route. After a certain number of iterations, the process is deliberately interrupted and an improvement procedure is applied to the current solution. Alonso et al. [1] developed a tabu search method for an extension of the PVRP in which each vehicle can service more than one route per day, provided that the vehicle's daily operation time does not exceed a maximum limit. Moreover, there are constraints on the types of vehicles that can visit each customer. Other variants of the PVRP have been considered by Francis et al. [7] (service frequency is a decision variable) and Mourgaya and Vanderbeck [12] (routing cost minimization and daily workload balance are the main optimization criteria).

The references described in the previous paragraph focus solely on vehicle routing for a fixed vehicle fleet. Another body of literature focuses on the so-called *capacitated fleet mix problem* (CFMP), which involves determining a minimum-cost fleet composition subject to an upper bound on the fleet size. Salhi and Rand [16] give an overview of early papers on the CFMP, and in particular stress that most techniques for optimal fleet composition neglect vehicle routing considerations. Ghiani et al. [8] considered the CFMP for homogeneous fleets in which the fleet size (number of vehicles) must be chosen to minimize the sum of fixed, variable, and hiring costs. This CFMP model was extended by Loxton et al. to capacitated heterogeneous fleets in [10, 11]. Note, however, that both the original formulation by Ghiani et al. [8] and the extended formulation by Loxton et al. [10, 11] ignore the effect of vehicle routing on the optimal fleet composition.

In this paper, we extend the CFMP models in [8, 10, 11] to consider vehicle routing aspects. We develop a new heuristic algorithm based on cross entropy—a concept that has already found success in the vehicle routing area. For example, Chepuri and Homem-de-Mello [5] use cross entropy to solve vehicle routing problems with stochastic demands. Our algorithm combines cross entropy with dynamic programming and golden section search to solve the combined optimal fleet composition and vehicle routing problem as a bilevel optimization problem, where the vehicle routes are optimized in the outer level, and the fleet composition is optimized in the inner level.

The paper is organized as follows. In Section 2, we describe the problem setting, introduce key notation and terminology, and give the precise mathematical formulation of the optimization problem under consideration. Then, in Section 3, we describe the new two-stage heuristic algorithm—which is based on cross entropy, dynamic programming, and golden section search—for solving the problem formulated in Section 2. Finally, in Section 4, we present numerical results for a range of randomly-generated test problems.

The results show that our new heuristic algorithm can consistently generate near-optimal solutions for large-scale problems beyond the reach of existing optimization solvers.

2. Optimal fleet composition with multi-period vehicle routing

2.1 *Problem description*

We consider the problem of composing a heterogeneous vehicle fleet for delivering a single commodity to various customers over a given planning horizon. If vehicle requirements exceed fleet capacity during any time period, then additional vehicles must be hired to address the shortfall.

The parameters in the model are defined as follows:

- m = number of potential vehicle types
- n = number of customers
- T = number of periods in the planning horizon
- $V_{\max}$ = maximum fleet size
- Q_k = capacity of a type-k vehicle
- q_{it} = demand of customer i during period t
- α_k = fixed cost per period of a type-k vehicle
- β_k = variable cost per period of a type-k vehicle
- γ_k = hiring cost per period of a type-k vehicle
- c_{ij}^k = travelling cost of a type-k vehicle for link (i, j)

Note that, to ensure the problem is non-trivial, we must have $\alpha_k + \beta_k < \gamma_k$ (i.e., the cost of operating an owned vehicle is less than the cost of hiring the same vehicle). This assumption is also made in references [8, 10, 11].

We define a network $(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the node set and $\mathcal{A}$ is the arc set. The nodes in $\mathcal{N}$ represent the start/end depots and the various customers; the arcs in $\mathcal{A}$ represent the possible transportation links. Note that both fully-connected and partially-connected networks can be considered. The nodes in $\mathcal{N}$ are enumerated as follows:

$$\mathcal{N} = \{0, 1, \ldots, n, n+1\},$$

where 0 represents the start depot, $n+1$ represents the end depot, and $1, \ldots, n$ represent the customers. Note that 0 and $n+1$ may in fact refer to the same point.

Our decision variables are defined below:

- v_k = number of type-k vehicles in the fleet
- r_t^k = number of type-k vehicles required during period t
- x_{ijt}^k = binary variable indicating whether a type-k vehicle travels from customer i to customer j during period t ($x_{ijt}^k = 1$ if this occurs; $x_{ijt}^k = 0$ otherwise)
- y_{ijt} = commodity flow from node i to node j during period t

We assume that split service is prohibited (i.e., each customer is served by precisely one vehicle in each period, though the particular vehicle may change from period to period). We also assume that customer demands are always non-zero and each customer is directly accessible from the start depot (if this is not possible, then we can simply add a dummy arc with sufficiently high cost).

If $v_k < r_t^k$, then all v_k owned vehicles of type k are used during period t and an additional hiring cost is incurred for $r_t^k - v_k$ type-k vehicles. Conversely, if $v_k \geq r_t^k$,

then only r_t^k owned type-k vehicles are used during period t. Hence, the total number of owned type-k vehicles used during period t is $\min\{v_k, r_t^k\}$. Similarly, the total number of type-k vehicles hired during period t is $\max\{r_t^k - v_k, 0\}$. It thus follows that the total cost associated with type-k vehicles (consisting of fixed, variable, hiring, and transportation costs) is

$$C_k(v_k, r_t^k, x_{ijt}^k) = \underbrace{T\alpha_k v_k}_{\text{Fixed cost}} + \underbrace{\beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\}}_{\text{Variable cost}} + \underbrace{\gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\}}_{\text{Hiring cost}} + \underbrace{\sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k}_{\text{Travelling cost}}.$$

This cost function is an extension of the cost functions in [8, 10, 11]. Our aim is to minimize the overall cost of owning and operating the fleet subject to the operational constraints described below.

- Fleet size constraint:

$$\sum_{k=1}^{m} v_k \leq V_{\max}. \tag{1}$$

- Each customer is visited precisely once during each period (recall our assumption that the customer demands are always non-zero):

$$\sum_{k=1}^{m} \sum_{j\in\mathcal{N}:\,(j,i)\in\mathcal{A}} x_{jit}^k = 1, \quad \forall i \in \{1,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}. \tag{2}$$

- Vehicle routes cannot start or end at a customer:

$$\sum_{j\in\mathcal{N}:\,(j,i)\in\mathcal{A}} x_{jit}^k = \sum_{j\in\mathcal{N}:\,(i,j)\in\mathcal{A}} x_{ijt}^k, \quad \forall i \in \{1,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}, \quad \forall k \in \{1,\ldots,m\}. \tag{3}$$

- Customer demand requirements:

$$\sum_{j\in\mathcal{N}:\,(j,i)\in\mathcal{A}} y_{jit} - \sum_{j\in\mathcal{N}:\,(i,j)\in\mathcal{A}} y_{ijt} = q_{it}, \quad \forall i \in \{1,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}. \tag{4}$$

- Vehicle capacity constraints:

$$\sum_{k=1}^{m} Q_k x_{0it}^k \geq y_{0it}, \quad \forall i \in \{1,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}. \tag{5}$$

- Commodity is transported between two nodes only if the corresponding link is

traversed:

$$\sum_{k=1}^{m} Q_k x_{ijt}^k \geq y_{ijt}, \quad \forall (i,j) \in \mathcal{A}, \quad \forall t \in \{1,\ldots,T\}. \tag{6}$$

- The number of type-k vehicles used during any period is equal to the number of type-k vehicles departing from the start depot during that period:

$$\sum_{j=1}^{n} x_{0jt}^k = r_t^k, \quad \forall t \in \{1,\ldots,T\}, \quad \forall k \in \{1,\ldots,m\}. \tag{7}$$

- No commodity is delivered to the end depot:

$$\sum_{j \in \mathcal{N}:\, (j,n+1) \in \mathcal{A}} y_{j(n+1)t} = 0, \quad \forall t \in \{1,\ldots,T\}. \tag{8}$$

- Non-negativity constraints:

$$v_k \geq 0, \quad \forall k \in \{1,\ldots,m\}, \tag{9}$$

$$r_t^k \geq 0, \quad \forall t \in \{1,\ldots,T\}, \quad \forall k \in \{1,\ldots,m\}, \tag{10}$$

$$y_{ijt} \geq 0, \quad \forall (i,j) \in \mathcal{A}, \quad \forall t \in \{1,\ldots,T\}. \tag{11}$$

- Integer/binary constraints:

$$v_k \in \mathrm{Z}, \quad \forall k \in \{1,\ldots,m\}, \tag{12}$$

$$r_t^k \in \mathrm{Z}, \quad \forall t \in \{1,\ldots,T\}, \quad \forall k \in \{1,\ldots,m\}, \tag{13}$$

$$x_{ijt}^k \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A}, \quad \forall t \in \{1,\ldots,T\}, \quad \forall k \in \{1,\ldots,m\}. \tag{14}$$

Our optimization problem can now be stated formally as follows: Choose the decision variables v_k, r_t^k, x_{ijt}^k, and y_{ijt} to minimize the cost function

$$\begin{aligned} \sum_{k=1}^{m} C_k(v_k, r_t^k, x_{ijt}^k) = \sum_{k=1}^{m} \Bigg\{ & T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} \\ & + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} + \sum_{t=1}^{T} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ijt}^k \Bigg\} \end{aligned} \tag{15}$$

subject to the constraints (1)-(14).

2.2 *MILP formulation*

The fleet composition problem described above is a mixed-integer *nonlinear* program due to the presence of the nonlinear "min" and "max" terms in the cost function (15). This problem can, however, be transformed into a mixed-integer *linear* program by introducing

additional decision variables and constraints. First, let h_t^k denote the number of type-k vehicles hired during period t. Then the "min" and "max" terms can be rewritten as

$$\min\{v_k, r_t^k\} = r_t^k - h_t^k, \quad \max\{r_t^k - v_k, 0\} = h_t^k.$$

Thus, the cost function for type-k vehicles becomes

$$\begin{aligned}\tilde{C}_k(v_k, h_t^k, r_t^k, x_{ijt}^k) &= T\alpha_k v_k + \beta_k \sum_{t=1}^{T}(r_t^k - h_t^k) + \gamma_k \sum_{t=1}^{T} h_t^k + \sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k \\ &= T\alpha_k v_k + \beta_k \sum_{t=1}^{T} r_t^k + (\gamma_k - \beta_k)\sum_{t=1}^{T} h_t^k + \sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k.\end{aligned}$$

Accordingly, the overall cost function (15) becomes

$$\begin{aligned}\sum_{k=1}^{m} \tilde{C}_k(v_k, h_t^k, r_t^k, x_{ijt}^k) = \sum_{k=1}^{m}\Bigg\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} r_t^k + (\gamma_k - \beta_k)\sum_{t=1}^{T} h_t^k \\ + \sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k \Bigg\}.\end{aligned} \tag{16}$$

The following constraints are also required:

$$0 \le h_t^k \le r_t^k, \quad \forall t \in \{1, \ldots, T\}, \quad \forall k \in \{1, \ldots, m\}, \tag{17}$$

and

$$r_t^k \le h_t^k + v_k, \quad \forall t \in \{1, \ldots, T\}, \quad \forall k \in \{1, \ldots, m\}. \tag{18}$$

These constraints ensure that $h_t^k = 0$ when $r_t^k \le v_k$ and $h_t^k = r_t^k - v_k$ when $r_t^k > v_k$. To explain why, recall that $\alpha_k + \beta_k < \gamma_k$ and thus the coefficient $(\gamma_k - \beta_k)$ in (16) is positive. It therefore follows that the optimal value of h_t^k is the smallest value that ensures (18) is satisfied. If $r_t^k \le v_k$, constraint (18) is satisfied for any non-negative value of h_t^k, and thus the optimal solution must be $h_t^k = 0$. On the other hand, if $r_t^k > v_k$, then the minimum value of h_t^k satisfying constraint (18) is $h_t^k = r_t^k - v_k$.

The MILP formulation for our fleet composition problem can now be stated as follows: Choose the decision variables v_k, h_t^k, r_t^k, x_{ijt}^k, and y_{ijt} to minimize the cost function (16) subject to the original constraints (1)-(14) and the new constraints (17)-(18). In principle, this MILP formulation can be solved using commerical solvers such as CPLEX. The disadvantage with this approach, however, is that the MILP's dimension typically explodes as the number of customers increases. For example, if there are 10 customers, 5 time periods, 5 vehicle types, and the transportation network is fully-connected, then the MILP formulation contains 3,655 discrete-valued decision variables and 720 continuous-valued decision variables. If the number of customers is increased to 20 and everything else stays the same, then the new MILP formulation contains 12,155 discrete-valued decision variables and 2,420 continuous-valued decision variables. As we discuss in Section 4, CPLEX struggles to solve problems of this size.

Note that our problem formulation does not contain any subtour elimination constraints. This is because subtours are automatically prohibited by the assumption of non-zero customer demands (i.e., $q_{it} > 0$ for all i and t). Indeed, suppose to the contrary that there exists a subtour $\{i_1, \ldots, i_p, i_1\}$ such that

$$x^k_{i_1 i_2 t} = \cdots = x^k_{i_{p-1} i_p t} = x^k_{i_p i_1 t} = 1.$$

Then constraints (2) and (4) imply

$$y_{i_{l-1} i_l t} - y_{i_l i_{l+1} t} = q_{i_l t}, \quad \forall l \in \{1, \ldots, p\},$$

where $i_0 = i_p$ and $i_{p+1} = i_1$. Thus, for each $l = 1, \ldots, p$,

$$y_{i_l i_{l+1} t} = y_{i_p i_1 t} - \sum_{d=1}^{l} q_{i_d t}.$$

Applying this equation for $l = p$ gives

$$y_{i_p i_1 t} = y_{i_p i_1 t} - \sum_{d=1}^{p} q_{i_d t}$$

and hence

$$\sum_{d=1}^{p} q_{i_d t} = 0.$$

But since the customer demands are always non-zero, this is a contradiction, as required.

3. A two-stage heuristic algorithm

We now introduce a two-stage heuristic algorithm for solving the combined fleet composition and vehicle routing problem described in Section 2. This algorithm is based on the following bi-level decomposition:

$$\min_{v_k, r^k_t, x^k_{ijt}, y_{ijt}} \sum_{k=1}^{m} C_k(v_k, r^k_t, x^k_{ijt}) = \min_{r^k_t, x^k_{ijt}, y_{ijt}} \min_{v_k} \sum_{k=1}^{m} C_k(v_k, r^k_t, x^k_{ijt}), \tag{19}$$

where the minimizations are subject to constraints (1)-(14). The inner minimization on the right-hand side of (19) involves determining the best fleet mix given fixed vehicle routes (as defined by r^k_t, x^k_{ijt}, and y_{ijt} in the outer stage). This is a CFMP that can be solved using the dynamic programming method developed in [10, 11]. The outer minimization on the right-hand side of (19) involves determining optimal vehicle routes for servicing the customer demands in each period. The two stages of our heuristic algorithm are described further below.

3.1 *Inner-stage optimization: Dynamic programming and golden section search*

The last term of C_k depends only on x_{ijt}^k and is therefore constant in the inner stage. Thus, the inner-level optimization problem can be stated as follows: Given r_t^k, x_{ijt}^k, and y_{ijt}, choose v_k to minimize the cost function

$$\sum_{k=1}^{m} \hat{C}_k(v_k) = \sum_{k=1}^{m} \left\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} \right\}$$

subject to constraints (1) and (9). Consider the following subproblem for fixed l and ξ:

$$\min_{\substack{v_1,\ldots,v_l \geq 0 \\ v_1+\cdots+v_l \leq \xi}} \sum_{k=1}^{l} \left\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} \right\}, \tag{20}$$

where $v_1, \ldots, v_l$ are integer decision variables. Let $g_l(\xi)$ denote the optimal cost of (20). For $l = 1$,

$$g_1(\xi) = \min_{v_1 \in \{0,\ldots,\xi\}} \left\{ T\alpha_1 v_1 + \beta_1 \sum_{t=1}^{T} \min\{v_1, r_t^1\} + \gamma_1 \sum_{t=1}^{T} \max\{r_t^1 - v_1, 0\} \right\}.$$

For $l \geq 2$, it follows from the *principle of optimality* that

$$\begin{aligned}
g_l(\xi) &= \min_{\substack{v_1,\ldots,v_l \geq 0 \\ v_1+\cdots+v_l \leq \xi}} \sum_{k=1}^{l} \left\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} \right\} \\
&= \min_{v_l \in \{0,\ldots,\xi\}} \min_{\substack{v_1,\ldots,v_{l-1} \geq 0 \\ v_1+\cdots+v_{l-1} \leq \xi - v_l}} \sum_{k=1}^{l} \left\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} \right\} \\
&= \min_{v_l \in \{0,\ldots,\xi\}} \Bigg[T\alpha_l v_l + \beta_l \sum_{t=1}^{T} \min\{v_l, r_t^l\} + \gamma_l \sum_{t=1}^{T} \max\{r_t^l - v_l, 0\} \\
&\qquad + \min_{\substack{v_1,\ldots,v_{l-1} \geq 0 \\ v_1+\cdots+v_{l-1} \leq \xi - v_l}} \sum_{k=1}^{l-1} \left\{ T\alpha_k v_k + \beta_k \sum_{t=1}^{T} \min\{v_k, r_t^k\} + \gamma_k \sum_{t=1}^{T} \max\{r_t^k - v_k, 0\} \right\} \Bigg] \\
&= \min_{v_l \in \{0,\ldots,\xi\}} \left[T\alpha_l v_l + \beta_l \sum_{t=1}^{T} \min\{v_l, r_t^l\} + \gamma_l \sum_{t=1}^{T} \max\{r_t^l - v_l, 0\} + g_{l-1}(\xi - v_l) \right].
\end{aligned} \tag{21}$$

Starting with $g_0 = 0$, we can use equation (21) to compute g_1, g_2, and so on until $g_m(V_{\max})$, which is the optimal cost of the inner-stage optimization problem. To successfully implement this recursion process, we need a fast method for performing the minimization on the right-hand side of (21). Instead of applying the brute force method of individually testing each value of v_l, we can apply the golden section method for much faster convergence, as shown in reference [11].

To do this, we extend the domain of g_l from $\{0, 1, \ldots, \xi\}$ to $[0, \infty)$ as follows:

$$g_l(\eta) = g_l(\lfloor \eta \rfloor) + (\eta - \lfloor \eta \rfloor) \cdot \big\{ g_l(\lfloor \eta \rfloor + 1) - g_l(\lfloor \eta \rfloor) \big\},$$

for each $\eta \in [0, \infty)$. It can be shown that this extended function is convex on $[0, \infty)$ (see references [10, 11]).

Suppose now that we relax the integer constraint on the right-hand side of equation (21). Then we obtain the following relaxed subproblem:

$$\min_{v_l \in [0,\xi]} h_l(v_l) = T\alpha_l v_l + \beta_l \sum_{t=1}^{T} \min\{v_l, r_t^l\} + \gamma_l \sum_{t=1}^{T} \max\{r_t^l - v_l, 0\} + g_{l-1}(\xi - v_l). \quad (22)$$

According to the results in [10, 11], since g_{l-1} is convex, h_l is also convex. Moreover, if ξ is an integer, then the minimization problem defined in (22) always has an integer solution. Thus, the integer constraint on the right-hand side of equation (21) is actually redundant, and it follows that we can compute $g_l(\xi)$ by solving (22). This problem is a convex optimization problem (see [10, 11]) and can be solved extremely quickly using the well-known golden section method. Please see [10, 11] for more details.

3.2 *Outer-stage optimization: Cross entropy*

The cross entropy method was originally introduced by Rubinstein [13] to estimate the probabilities of rare events in complex stochastic networks. It was soon realized that cross entropy can be adapted to solve difficult combinatorial optimization problems. This is done by transforming the deterministic optimization problem into a related stochastic optimization problem and then applying the rare event simulation techniques developed by Rubinstein and others [6, 14, 15, 17].

A typical iteration of the cross entropy method involves the following two steps [15]:

1. Generate a random sample of potential solutions according to some random mechanism (which depends on various adjustable parameters).
2. Based on the random sample obtained in Step 1, update the parameters of the random mechanism with the aim of producing an improved random sample in the next iteration.

For optimization problems, we start with an initial probability distribution defined over the feasible region (normally just the uniform distribution). At each iteration, the probability distribution is updated adaptively based on the random sample collected in the previous iteration. As the algorithm progresses, the probability distribution ideally converges to a discrete uniform distribution that assigns equal positive probabilities to global optima and zero probabilities elsewhere.

Our heuristic algorithm for solving the fleet composition problem (19) uses cross entropy to generate random samples of routes in the outer-stage optimization problem. Since some of the randomly-generated routes may violate the vehicle capacity constraints (5), the algorithm includes steps to decompose routes when necessary.

The routes are generated using a random variable $z_i(t)$, which represents the node serving node i during period t. This random variable is defined by a $n \times (n+1)$ probability matrix $P(t) = [P_{ij}(t)]$, where $P_{ij}(t)$ denotes the probability that $z_i(t) = j$. Clearly, the sum of each row in $P(t)$ is one. Note also that the column index for $P(t)$ ranges from 0 to n (since $z_i(t)$ can assume any integer in $\{0, \ldots, n\}$). Furthermore, let $P_i(t)$ denote the

ith row of $P(t)$.

Initially, each node can be served via any of its connected nodes with equal likelihood. Thus, in the first iteration,

$$P_{ij}(t) = \begin{cases} \frac{1}{|\mathcal{A}_i|}, & \text{if } (j,i) \in \mathcal{A}_i, \\ 0, & \text{if } (j,i) \notin \mathcal{A}_i, \end{cases}$$

where $\mathcal{A}_i$ denotes the set of arcs entering node i. For example, if customer 1 is connected to the depot, customer 5, and customer 6, then the first row of $P(t)$ is

$$\left[\tfrac{1}{3}\ 0\ 0\ 0\ 0\ \tfrac{1}{3}\ \tfrac{1}{3}\ 0 \cdots 0\right].$$

The procedure for randomly generating routes using $P(t)$ is described below. Note that the probabilities in $P(t)$ are updated whenever a new value for $z_i(t)$ is selected. This is to ensure that the algorithm creates a series of valid paths starting at node 0. For example, if customer 1 is served via customer 2 during any period, then the probability that another customer is served via customer 2 in the same period is zero, and the probability that customer 2 itself is served via customer 1 is also zero. Moreover, when selecting the value of $z_i(t)$, we need to exclude any node on the unique path starting at i to avoid cycles (the unique path starting at node i could consist of just i itself).

ALGORITHM 3.1 Generates random vehicle routes using the distribution matrix $P(t)$.

1. Set $1 \to t$ and $1 \to i$.
2. Set $P_i(t) \to \theta = [\theta_0, \ldots, \theta_n]$.
3. Set $0 \to \theta_{z_j(t)}$ for all $j = 1, \ldots, i-1$ such that $z_j(t) \neq 0$.
4. Determine the set $\mathcal{M}_i$ consisting of all nodes on the unique path starting at node i (note that $i \in \mathcal{M}_i$ and thus $\mathcal{M}_i \neq \emptyset$).
5. Set $0 \to \theta_j$ for all $j \in \mathcal{M}_i$.
6. Normalize θ:

$$\frac{\theta}{\sum_{j=0}^{n} \theta_j} \to \theta.$$

7. Generate a random instance of $z_i(t)$ using the distribution defined by θ.
8. If $i < n$, then set $i + 1 \to i$ and return to Step 2. Otherwise, go to Step 9.
9. If $t < T$, then set $1 \to i$ and $t + 1 \to t$ and return to Step 2. Otherwise, stop.

The output of Algorithm 3.1 is a set of values $z_i(t)$, $i = 1, \ldots, n$, $t = 1, \ldots, T$, that define the vehicle routes during each period. Our cross entropy heuristic works by continually invoking Algorithm 3.1 to generate different sets of routes. The cost corresponding to each set of routes is computed by solving the inner-stage optimization problem, and then this information is used to update the probability matrix $P(t)$, with the aim of producing better routes in the next iteration. Ideally, $P(t)$ should converge to a periodic certainty matrix in which each row has exactly one element equal to 1 and all other elements equal to 0. Since $0 \leq P_{ij}(t) \leq 1$, we have $P_{ij}(t)^2 \leq P_{ij}(t)$. Hence,

$$\sum_{j=0}^{n} P_{ij}(t)^2 \leq \sum_{j=0}^{n} P_{ij}(t) = 1.$$

This implies that the Euclidean norm of the probability matrix $P(t)$ satisfies

$$\|P(t)\| = \sqrt{\sum_{i=1}^{n}\sum_{j=0}^{n} P_{ij}(t)^2} \leq \sqrt{n}.$$

Any $n \times (n+1)$ periodic certainty matrix has the maximum Euclidean norm of $\sqrt{n}$. Hence, we use the following convergence criterion:

$$-\epsilon < \|P(t)\| - \sqrt{n} < \epsilon, \quad \forall t \in \{1, \ldots, T\},$$

where $\epsilon > 0$ is a given tolerance parameter. Our cross entropy algorithm terminates when this inequality is satisfied, or when the best upper bound for the optimal cost does not change over $p_{\max}$ iterations.

The conceptual framework for the cross entropy algorithm is described below. The input parameters are: M (sample size), M_e (elite sample size), $\alpha \in [0,1]$ (smoothing parameter), ϵ (tolerance parameter), $p_{\max}$ (convergence indicator). We use $C^*_{\max}$ to denote the upper bound for the optimal cost.

ALGORITHM 3.2 Solves the outer-stage optimization problem using cross entropy.

1. Set $\infty \to C^*_{\max}$ and $0 \to p$.
2. Define the probability matrix $P(t)$ as follows:

$$P_{ij}(t) = \begin{cases} \frac{1}{|\mathcal{A}_i|}, & \text{if } (j,i) \in \mathcal{A}_i, \\ 0, & \text{if } (j,i) \notin \mathcal{A}_i, \end{cases}$$

   where $\mathcal{A}_i$ denotes the set of arcs entering node i.
3. Using Algorithm 3.1 with probability matrix $P(t)$, generate M sets of vehicle routes $\{\mathcal{R}_1, \ldots, \mathcal{R}_M\}$, where each set of routes satisfies constraints (2) and (3).
4. For each $l = 1, \ldots, M$, determine the values of $(r^k_t, x^k_{ijt}, y_{ijt})$ corresponding to $\mathcal{R}_l$. Each set of values $(r^k_t, x^k_{ijt}, y_{ijt})$ satisfies constraints (2)-(8), (10)-(11), and (13)-(14).
5. For each $l = 1, \ldots, M$, solve the inner-level optimization problem in (19) corresponding to $\mathcal{R}_l$. Let $C^*_1, \ldots, C^*_M$ denote the optimal costs obtained.
6. If $\min\{C^*_1, \ldots, C^*_M\} < C^*_{\max}$, then set $\min\{C^*_1, \ldots, C^*_M\} \to C^*_{\max}$ and $0 \to p$. Otherwise, set $p + 1 \to p$.
7. If $p = p_{\max}$, then stop: take the best set of routes found thus far as the optimal solution for the outer-stage optimization. Otherwise, go to Step 8.
8. If $-\epsilon < \|P(t)\| - \sqrt{n} < \epsilon$ for each $t = 1, \ldots, T$, then stop: take the best set of routes found thus far as the optimal solution for the outer-stage optimization. Otherwise, go to Step 9.
9. Form the elite sample by collecting the best M_e elements of $\{\mathcal{R}_1, \ldots, \mathcal{R}_M\}$ (as measured by cost).
10. For each $i = 1, \ldots, n$, $j = 0, \ldots, n$, and $t = 1, \ldots, T$, determine $v_{ij}(t)$, the number of times customer i is served via customer j during period t in the elite sample.

11. Update the probability matrix $P(t)$:

$$(1-\alpha)P_{ij}(t) + \alpha\frac{v_{ij}(t)}{M_e} \to P_{ij}(t),$$
$$\forall i \in \{1,\ldots,n\}, \quad \forall j \in \{0,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}.$$

12. Return to Step 3.

In Algorithm 3.2, each set of routes $\mathcal{R}_l$ is defined by the corresponding values of $z_i(t)$ (obtained via Algorithm 3.1). The smoothing parameter α governs the relative contribution between historical data and elite sample data when updating the probability matrix in Step 11. We want the updated probability matrix to mimic the best routes in the elite sample. Note that

$$\sum_{j=0}^{n} v_{ij}(t) = M_e, \quad \forall i \in \{1,\ldots,n\}, \quad \forall t \in \{1,\ldots,T\}.$$

Thus, the $n \times (n+1)$ matrix with (i,j)th element equal to $v_{ij}(t)/M_e$ is a valid probability matrix in the same form as $P(t)$. This ensures that the matrix update in Step 11 is feasible.

Step 4 of Algorithm 3.2 involves determining the values of $(r_t^k, x_{ijt}^k, y_{ijt})$ corresponding to the routes generated in Step 3 (which are defined by the values of $z_i(t)$). The problem with using these routes directly is that some of them may be too long for a single vehicle to service. For example, consider a route with three customers, each requiring 10 units of commodity. Since the route requires total deliveries of 30 units, if the capacity of each vehicle is less than 30, then the vehicle capacity constraints will never be satisfied for this route. Such routes must first be decomposed into shorter routes before determining the corresponding values of $(r_t^k, x_{ijt}^k, y_{ijt})$. This is done using the following procedure.

ALGORITHM 3.3 Determines the values of $(r_t^k, x_{ijt}^k, y_{ijt})$ corresponding to a given set of routes $\mathcal{R}_l$ (each route starts at node 0 and ends at node $n+1$).

1. Set $0 \to x_{ijt}^k$ and $0 \to y_{ijt}$ for all $(i,j) \in \mathcal{A}$, $t \in \{1,\ldots,T\}$, and $k \in \{1,\ldots,m\}$.
2. Set $1 \to t$.
3. Collect the vehicle routes for period t into a set $\mathcal{R}_l(t)$.
4. Select $\psi \in \mathcal{R}_l(t)$.
5. List the nodes of ψ in order of occurrence as $i_0, i_1, \ldots, i_p$, where $i_0 = 0$ and $i_p = n+1$.
6. Compute

$$\xi_j = \xi_{j-1} + q_{i_j t}, \quad \forall j \in \{1,\ldots,p-1\},$$

   where $\xi_0 = 0$.
7. If $\xi_{p-1} > \max_k Q_k$, then define $j^* \in \{1,\ldots,p-1\}$ as the unique integer such that

$$\xi_{j^*-1} < \max_k Q_k \le \xi_{j^*}.$$

8. If $\xi_{p-1} > \max_k Q_k$, then ψ must be decomposed: replace $\psi = \{i_0, i_1, \ldots, i_p\}$ in $\mathcal{R}_l(t)$ with $\{0, i_1, \ldots, i_{j^*-1}, n+1\}$ and $\{0, i_{j^*}, \ldots, i_{p-1}, n+1\}$ and return to Step 4.

Otherwise, route decomposition is not required: set

$$x^{k^*}_{i_{j-1}i_j t} = 1, \quad \forall j \in \{1, \ldots, p\},$$
$$y_{i_{j-1}i_j t} = \xi_{p-1} - \sum_{l=1}^{j-1} q_{i_l t}, \quad \forall j \in \{1, \ldots, p\},$$

where

$$k^* = \arg\min\{\, \alpha_k + \beta_k + \gamma_k : \xi_{p-1} \le Q_k \,\}.$$

9. Remove $\psi = \{i_0, i_1, \ldots, i_p\}$ from $\mathcal{R}_l(t)$.
10. If $\mathcal{R}_l(t) \neq \emptyset$, then return to Step 4. Otherwise, go to Step 11.
11. If $t < T$, then set $t + 1 \to t$ and return to Step 3. Otherwise, define

$$r^k_t = \sum_{j=1}^{n} x^k_{0jt}, \quad \forall t \in \{1, \ldots, T\}, \quad \forall k \in \{1, \ldots, m\}.$$

Algorithm 3.3 can be used to implement Step 4 in Algorithm 3.2. Notice that, when assigning vehicles to routes, Algorithm 3.3 always chooses the least expensive vehicle type (see the definition of k^* in Step 8). Although this is certainly a logical choice, it may not be the best choice. In fact, assigning the least expensive vehicle type in Algorithm 3.3 could result in hiring an additional vehicle while an owned vehicle sits idle. For example, suppose that a route can be serviced by either type-1 or type-2 vehicles, with type-1 vehicles the cheapest option. Suppose also that, due to routing requirements in other periods, the optimal fleet mix includes type-2 vehicles, but not type-1 vehicles. In this case, Algorithm 3.2 will assign a type-1 vehicle to the route, thus incurring an unnecessary hiring cost when type-2 owned vehicles are available. To avoid situations like this, we can implement the following improvement procedure to modify the values of $(r^k_t, x^k_{ijt}, y_{ijt})$.

Algorithm 3.4 Modifies vehicle assignments to eliminate unnecessary hiring costs, given a fleet composition $(v_1, \ldots, v_m)$ and set of routes defined by $(r^k_t, x^k_{ijt}, y_{ijt})$.

1. Set $1 \to t$.
2. Set $v_k \to w_k$, $k = 1, \ldots, m$.
3. Collect the vehicle routes for period t into a set $\mathcal{R}_l(t)$.
4. Select $\psi \in \mathcal{R}_l(t)$.
5. List the nodes of ψ in order of occurrence as $i_0, i_1, \ldots, i_p$, where $i_0 = 0$ and $i_p = n+1$.
6. Compute the total commodity load for route ψ:

$$\xi = \sum_{j=1}^{p-1} q_{i_j t}.$$

7. If $\{\, k \in \{1, \ldots, m\} : \xi \le Q_k,\ w_k > 0 \,\} \neq \emptyset$, then set

$$\arg\min\{\, \alpha_k + \beta_k + \gamma_k : \xi \le Q_k,\ w_k > 0 \,\} \to k^*,$$
$$w_{k^*} - 1 \to w_{k^*}.$$

Otherwise, set

$$\arg\min\{\,\alpha_k+\beta_k+\gamma_k:\ \xi\le Q_k\,\}\to k^*.$$

8. Set

$$\tilde{x}^{k^*}_{i_{j-1}i_jt}=1,\quad \forall j\in\{1,\dots,p\},$$
$$\tilde{y}_{i_{j-1}i_jt}=\xi-\sum_{l=1}^{j-1}q_{i_lt},\quad \forall j\in\{1,\dots,p\}.$$

9. Remove $\psi=\{i_0,i_1,\dots,i_p\}$ from $\mathcal{R}_l(t)$.
10. If $\mathcal{R}_l(t)\neq\emptyset$, then return to Step 4. Otherwise, go to Step 11.
11. If $t<T$, then set $t+1\to t$ and return to Step 2. Otherwise, define

$$\tilde{r}^k_t=\sum_{j=1}^{n}\tilde{x}^k_{0jt},\quad \forall t\in\{1,\dots,T\},\quad \forall k\in\{1,\dots,m\}.$$

Note that k^* in Algorithm 3.4 is the minimum-cost vehicle type among all idle vehicle types in the fleet. After applying Algorithms 3.3 and 3.4, we can solve the inner-level optimization problems corresponding to $(r^k_t,x^k_{ijt},y_{ijt})$ and $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$. The best solution (in terms of cost) out of $(r^k_t,x^k_{ijt},y_{ijt})$ and $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$ should then be used in the cross entropy procedure, with the inferior solution discarded. The following algorithm is a modification of Algorithm 3.2 that incorporates this idea.

ALGORITHM 3.5 Solves the outer-stage optimization problem using cross entropy with the improvement procedure in Algorithm 3.4.

1. Set $\infty\to C^*_{\max}$ and $0\to p$.
2. Define the probability matrix $P(t)$ as follows:

$$P_{ij}(t)=\begin{cases}\frac{1}{|\mathcal{A}_i|}, & \text{if } (j,i)\in\mathcal{A}_i,\\ 0, & \text{if } (j,i)\notin\mathcal{A}_i,\end{cases}$$

   where $\mathcal{A}_i$ denotes the set of arcs entering node i.
3. Using Algorithm 3.1 with probability matrix $P(t)$, generate M sets of vehicle routes $\{\mathcal{R}_1,\dots,\mathcal{R}_M\}$.
4. Set $1\to l$.
5. Use Algorithm 3.3 to determine the values of $(r^k_t,x^k_{ijt},y_{ijt})$ corresponding to $\mathcal{R}_l$.
6. Solve the inner-level optimization problem corresponding to $(r^k_t,x^k_{ijt},y_{ijt})$.
7. Use Algorithm 3.4 to determine the values of $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$ corresponding to $\mathcal{R}_l$.
8. Solve the inner-level optimization problem corresponding to $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$.
9. If the optimal cost of $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$ is less than the optimal cost of $(r^k_t,x^k_{ijt},y_{ijt})$, then keep $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$ and discard $(r^k_t,x^k_{ijt},y_{ijt})$. Otherwise, keep $(r^k_t,x^k_{ijt},y_{ijt})$ and discard $(\tilde{r}^k_t,\tilde{x}^k_{ijt},\tilde{y}_{ijt})$.
10. If $l<M$, then set $l+1\to l$ and return to Step 5. Otherwise, go to Step 11.
11. Let $C^*_1,\dots,C^*_M$ denote the optimal costs corresponding to $\mathcal{R}_1,\dots,\mathcal{R}_M$.

12. If $\min\{C_1^*, \ldots, C_M^*\} < C_{\max}^*$, then set $\min\{C_1^*, \ldots, C_M^*\} \to C_{\max}^*$ and $0 \to p$. Otherwise, set $p + 1 \to p$.
13. If $p = p_{\max}$, then stop: take the best set of routes found thus far as the optimal solution for the outer-stage optimization. Otherwise, go to Step 14.
14. If $-\epsilon < \|P(t)\| - \sqrt{n} < \epsilon$ for each $t = 1, \ldots, T$, then stop: take the best set of routes found thus far as the optimal solution for the outer-stage optimization. Otherwise, go to Step 15.
15. Form the elite sample by collecting the best M_e elements of $\{\mathcal{R}_1, \ldots, \mathcal{R}_M\}$ (as measured by cost).
16. For each $i = 1, \ldots, n$, $j = 0, \ldots, n$, and $t = 1, \ldots, T$, determine $v_{ij}(t)$, the number of times customer i is served via customer j during period t in the elite sample.
17. Update the probability matrix $P(t)$:

$$(1-\alpha)P_{ij}(t) + \alpha\frac{v_{ij}(t)}{M_e} \to P_{ij}(t),$$
$$\forall i \in \{1, \ldots, n\}, \quad \forall j \in \{0, \ldots, n\}, \quad \forall t \in \{1, \ldots, T\}.$$

18. Return to Step 3.

4. Numerical results

For numerical testing, we implemented Algorithm 3.5 in the Fortran programming language. We considered three categories of test problems:

- Small-size problems ($n = 6$, $m = 6$, $T = 6$)
- Medium-size problems ($n = 20$, $m = 10$, $T = 5$)
- Large-size problems ($n = 60$, $m = 5$, $T = 4$)

For each category, we randomly generated 20 feasible problem instances. These instances vary in complexity, as the feasible region can be extremely tight for some choices of the problem parameters.

Recall that Section 2 gives a MILP formulation for the optimal fleet composition problem. An alternative MILP formulation can be obtained by introducing new nonnegative decision variables δ_t^{+k} and δ_t^{-k}, $t = 1, \ldots, T$, $k = 1, \ldots, m$, such that

$$\left|v_k - r_t^k\right| = \delta_t^{+k} + \delta_t^{-k}. \tag{23}$$

The "min" and "max" terms in the cost function (15) can then be rewritten as follows:

$$\min\{v_k, r_t^k\} = \tfrac{1}{2}r_t^k + \tfrac{1}{2}v_k - \tfrac{1}{2}\left|v_k - r_t^k\right| = \tfrac{1}{2}r_t^k + \tfrac{1}{2}v_k - \tfrac{1}{2}\delta_t^{+k} - \tfrac{1}{2}\delta_t^{-k}$$

and

$$\max\{r_t^k - v_k, 0\} = \tfrac{1}{2}r_t^k - \tfrac{1}{2}v_k + \tfrac{1}{2}\left|v_k - r_t^k\right| = \tfrac{1}{2}r_t^k - \tfrac{1}{2}v_k + \tfrac{1}{2}\delta_t^{+k} + \tfrac{1}{2}\delta_t^{-k}.$$

The cost function for type-k vehicles thus becomes

$$\tilde{C}_k(v_k, r_t^k, x_{ijt}^k, \delta_t^{+k}, \delta_t^{-k}) = T\alpha_k v_k + \tfrac{1}{2}\beta_k \sum_{t=1}^{T} \left\{ r_t^k + v_k - \delta_t^{+k} - \delta_t^{-k} \right\} + \tfrac{1}{2}\gamma_k \sum_{t=1}^{T} \left\{ r_t^k - v_k + \delta_t^{+k} + \delta_t^{-k} \right\} + \sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k.$$

Hence, the overall cost function (15) becomes

$$\sum_{k=1}^{m} \tilde{C}_k(v_k, r_t^k, x_{ijt}^k, \delta_t^{+k}, \delta_t^{-k}) = \sum_{k=1}^{m} \Bigg\{ T\alpha_k v_k + \tfrac{1}{2}\beta_k \sum_{t=1}^{T} \left\{ r_t^k + v_k - \delta_t^{+k} - \delta_t^{-k} \right\} + \tfrac{1}{2}\gamma_k \sum_{t=1}^{T} \left\{ r_t^k - v_k + \delta_t^{+k} + \delta_t^{-k} \right\} + \sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k \Bigg\}.$$

This equation can be rearranged as follows:

$$\sum_{k=1}^{m} \tilde{C}_k(v_k, r_t^k, x_{ijt}^k, \delta_t^{+k}, \delta_t^{-k}) = \sum_{k=1}^{m} \Bigg\{ T\alpha_k v_k + \tfrac{1}{2}(\gamma_k - \beta_k) \sum_{t=1}^{T} \left\{ \delta_t^{+k} + \delta_t^{-k} \right\} + \tfrac{1}{2}(\beta_k - \gamma_k) T v_k + \tfrac{1}{2}(\beta_k + \gamma_k) \sum_{t=1}^{T} r_t^k + \sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ijt}^k \Bigg\}. \tag{24}$$

The following constraints must also be imposed:

$$v_k - r_t^k = \delta_t^{+k} - \delta_t^{-k}, \quad \forall t \in \{1, \ldots, T\}, \quad \forall k \in \{1, \ldots, m\}, \tag{25}$$

and

$$\delta_t^{+k} \geq 0, \quad \delta_t^{-k} \geq 0, \quad \forall t \in \{1, \ldots, T\}, \quad \forall k \in \{1, \ldots, m\}. \tag{26}$$

The alternative MILP formulation is stated as follows: Choose the decision variables v_k, r_t^k, x_{ijt}^k, y_{ijt}, δ_t^{+k}, and δ_t^{-k} to minimize the cost function (24) subject to the original constraints (1)-(14) and the new constraints (25) and (26). In our numerical experience, this new MILP formulation is easier to solve than the formulation in Section 2.

To explain why δ_t^{+k} and δ_t^{-k} satisfy (23), first note that $\gamma_k > \beta_k$ and thus the coefficient in front of $\delta_t^{+k} + \delta_t^{-k}$ in (24) is positive. Hence, constraint (25) ensures that any optimal solution satisfies $\delta_t^{+k}\delta_t^{-k} = 0$. If $\delta_t^{+k} > 0$, then $\delta_t^{-k} = 0$ and

$$\delta_t^{+k} + \delta_t^{-k} = \delta_t^{+k} = \delta_t^{+k} - \delta_t^{-k} = v_k - r_t^k = \left| v_k - r_t^k \right|, \tag{27}$$

as required. Similarly, if $\delta_t^{-k} > 0$, then $\delta_t^{+k} = 0$ and

$$\delta_t^{+k} + \delta_t^{-k} = \delta_t^{-k} = \delta_t^{-k} - \delta_t^{+k} = r_t^k - v_k = \left| v_k - r_t^k \right|. \tag{28}$$

Equations (27) and (28) show that the transformed cost function (24) reconciles with the original cost function (15) through identity (23).

For the numerical simulations, we randomly generated 20 small-size problems, 20 medium-size problems, and 20 large-size problems with the parameters given at the beginning of this section. The computer used in our simulations was equipped with an Intel 2 Core CPU with 2GB RAM. Each test instance was transformed into a MILP using the formulation described above and then solved using CPLEX 12.4. For each small-size and medium-size problem instance, CPLEX was run until termination and our bilevel heuristic was run 1,000 times; for each large-size problem instance, CPLEX was run for a fixed time of 2 hours and our bilevel heuristic was run 100 times. Tables 1-3 show the average run times and average cost values for our algorithm compared with the corresponding values for CPLEX. The percentage of routes that need to be decomposed is also given. Note that CPLEX could only verify optimality (i.e., obtain a zero optimality gap) in 9 of the 20 small-size problems, and none of the medium-size and large-size problems. For the instances where the optimality gap from CPLEX is zero, we have reported the optimality gap for our heuristic (see the "Av. Cost" column in Table 1). The average gap for our heuristic over the 9 feasible small-size problem instances is 3.25%, which shows that the heuristic can achieve near-optimal solutions in a short period of time. It is not possible to compare solutions when CPLEX's optimality gap is positive because the solution obtained by CPLEX in this case may not be feasible for the original problem. Recall that our algorithm always produces a feasible solution, whereas CPLEX may produce infeasible results (only the optimal solution of the MILP formulation described in (24)-(26) is guaranteed to be feasible for the original problem).

Table 4 gives a comparison between our algorithm and CPLEX for a fixed running time of 2 hours, where the "Difference %" column gives the percentage improvement of CPLEX compared with our heuristic. The test problems for this comparison are the same as the large-size test problems in Table 3. CPLEX produces feasible solutions in only 6 of the 20 test instances and the "Difference %" column is only applicable for these instances. For these "CPLEX feasible" instances, our algorithm produces feasible results within 3% of CPLEX's solution on average. For the other instances, our algorithm produces a feasible solution whereas CPLEX does not.

The results clearly show that our heuristic algorithm compares well with CPLEX in terms of optimal objective value, and importantly, it can generate feasible solutions in situations where CPLEX struggles. Moreover, in terms of computational time, our algorithm is significantly quicker than CPLEX: in all instances in Tables 1-3, our algorithm took less than 3.5% of CPLEX's computation time.

5. Conclusion

In this paper, we considered a combined optimal fleet composition and multi-period vehicle routing problem. As shown in Section 2, this problem can actually be formulated as a large-scale MILP, but our numerical experience shows that commercial MILP solvers such as CPLEX are ineffective with large-scale problem instances. To overcome this challenge, we developed a two-stage heuristic algorithm based on cross entropy, dynamic programming, and golden section search. Extensive numerical testing (see Section 4) indicates that this heuristic has excellent convergence properties, consistently yielding optimal or near-optimal solutions in rapid time. Future work will involve extending the algorithm to more complex problem settings, including those with uncertain demand requirements and time window constraints.

	CPLEX			Bilevel Heuristic			
Instance	Time [s]	Cost	Gap %	Av. Time [s]	Av. Cost	Av. Gap %	RO %
1	1.0478×10^4	6.335×10^3	0.00	3.86	6.931×10^3	8.60	23.18
2	4.9480×10^3	7.494×10^3	0.00	4.40	7.510×10^3	0.21	23.90
3	1.9020×10^4	7.145×10^3	0.00	4.66	7.504×10^3	4.78	22.80
4	3.5404×10^4	7.467×10^3	0.23	4.29	7.491×10^3	-	23.87
5	6.4584×10^4	7.347×10^3	3.73	4.29	7.366×10^3	-	24.30
6	5.9070×10^3	7.253×10^3	0.00	4.46	7.618×10^3	4.79	23.70
7	8.1305×10^4	8.157×10^3	0.00	6.30	8.208×10^3	0.62	17.29
8	5.4600×10^3	7.366×10^3	0.00	4.77	7.572×10^3	2.72	22.39
9	2.1450×10^4	6.858×10^3	3.16	5.05	7.012×10^3	-	21.07
10	1.6788×10^4	6.594×10^3	0.30	4.75	7.094×10^3	-	21.04
11	1.2488×10^4	7.187×10^3	6.16	6.54	7.466×10^3	-	15.97
12	5.6729×10^4	6.769×10^3	4.43	4.52	7.071×10^3	-	20.92
13	1.3167×10^4	7.396×10^3	9.47	5.24	7.313×10^3	-	20.55
14	1.1668×10^4	7.667×10^3	0.00	4.86	7.952×10^3	3.58	21.93
15	2.5860×10^3	7.775×10^3	14.62	5.41	7.589×10^3	-	20.04
16	5.6930×10^3	7.504×10^3	0.00	7.34	7.763×10^3	3.34	15.44
17	2.0410×10^4	8.112×10^3	7.48	6.31	7.963×10^3	-	16.46
18	4.5510×10^3	8.204×10^3	9.25	5.78	8.157×10^3	-	18.52
19	2.5409×10^4	7.811×10^3	5.43	5.81	7.889×10^3	-	17.90
20	4.8000×10^2	8.095×10^3	0.00	7.16	8.147×10^3	0.64	16.09

Table 1. Numerical results for a suite of 20 randomly-generated small-size problems, where "RO" refers to the route overload percentage (i.e., the percentage of routes that are overloaded and need to be decomposed in our heuristic algorithm).

References

[1] F. Alonso, M.J. Alvarez, and J.E. Beasley, *A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions*. J. Oper. Res. Soc. 59 (2008), pp. 963–976.

[2] R. Baldacci, M. Battarra, and D. Vigo, *Routing a heterogeneous fleet of vehicles*, in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, and E. Wasil, eds., Springer, New York, USA, 2008, pp. 3–27.

[3] L. Bertazzi, G. Paletta, and M.G. Speranza, *An improved heuristic for the period traveling salesman problem*, Comput. Oper. Res. 31 (2004), pp. 1215–1222.

[4] I.M. Chao, B.L. Golden, and E. Wasil, *A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions*, Amer. J. Math. Management Sci. 13 (1993), pp. 371–406.

[5] K. Chepuri and T. Homem-de-Mello, *Solving the vehicle routing problem with stochastic demands using the cross-entropy method*, Ann. Oper. Res. 134 (2005), pp. 153–181.

[6] A. Eshragh, J. Filar, and A. Nazar, *A projection-adapted cross entropy (PACE) method for transmission network planning*, Energy Syst. 2 (2011), pp. 189–208.

[7] P.M. Francis, K.R. Smilowitz, and M. Tzur, *The period vehicle routing problem and its extensions*, in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, and E. Wasil, eds., Springer, New York, USA, 2008, pp. 73–102.

[8] G. Ghiani, G. Laporte, and R. Musmanno, *Introduction to Logistics Systems Planning and Control*, John Wiley, Chichester, UK, 2004.

[9] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen, *Industrial aspects and literature survey: Fleet composition and routing*, Comput. Oper. Res. 37 (2010), pp. 2041–2061.

[10] R. Loxton and Q. Lin, *Optimal fleet composition via dynamic programming and golden section search*, J. Ind. Manag. Optim. 7 (2011), pp. 875–890.

[11] R. Loxton, Q. Lin, and K. L. Teo, *A stochastic fleet composition problem*, Comput. Oper. Res. 39 (2012), pp. 3177–3184.

[12] M. Mourgaya and F. Vanderbeck, *Column generation based heuristic for tactical planning in multi-period vehicle routing*, European J. Oper. Res. 183 (2007), pp. 1028–1041.

[13] R.Y. Rubinstein, *Optimization of computer simulation models with rare events*, European J. Oper. Res. 99 (1997), pp. 89–112.

[14] R.Y. Rubinstein, *The cross-entropy method for combinatorial and continuous optimization*,

	CPLEX			Bilevel Heuristic		
Instance	Time [s]	Cost	Gap %	Av. Time [s]	Av. Cost	RO %
1	9.5810×10^3	2.0579×10^4	20.83	16.30	2.0376×10^4	38.50
2	7.8110×10^3	1.9954×10^4	21.54	16.47	1.8545×10^4	36.92
3	6.9354×10^4	2.0132×10^4	18.45	4.66	1.9375×10^4	38.27
4	5.3000×10^3	1.6336×10^4	19.72	15.96	1.5434×10^4	33.69
5	6.9170×10^3	1.5105×10^4	17.25	16.23	1.5271×10^4	32.44
6	9.6190×10^3	1.6125×10^4	18.11	15.89	1.5481×10^4	33.92
7	5.7630×10^3	1.5454×10^4	16.72	16.22	1.5343×10^4	32.37
8	7.1230×10^3	1.9507×10^4	21.20	16.95	1.8367×10^4	35.32
9	5.5520×10^3	1.6041×10^4	18.98	16.32	1.5705×10^4	32.47
10	6.2110×10^3	1.7187×10^4	16.61	16.35	1.7038×10^4	35.65
11	6.3830×10^3	1.9733×10^4	18.59	16.88	1.9488×10^4	37.04
12	8.9670×10^3	1.9613×10^4	17.37	16.83	1.9005×10^4	37.15
13	5.4330×10^3	1.3910×10^4	14.00	15.88	1.3942×10^4	31.74
14	9.9250×10^3	1.5326×10^4	21.27	16.00	1.4285×10^4	32.02
15	7.7130×10^3	1.3970×10^4	14.97	16.06	1.3997×10^4	31.92
16	6.2870×10^3	2.1237×10^4	18.60	16.25	2.1286×10^4	38.74
17	1.6629×10^4	1.9791×10^4	21.09	16.08	1.9110×10^4	37.91
18	1.0839×10^4	1.5704×10^4	19.03	16.03	1.4759×10^4	32.15
19	3.4266×10^4	1.9502×10^4	17.61	16.76	1.9469×10^4	36.96
20	3.4266×10^4	1.7995×10^4	22.68	16.24	1.6407×10^4	34.50

Table 2. Numerical results for a suite of 20 randomly-generated medium-size problems, where "RO" refers to the route overload percentage (i.e., the percentage of routes that are overloaded and need to be decomposed in our heuristic algorithm).

Methodol. Comput. Appl. Probab. 1 (1999), pp. 127-190.

[15] R.Y. Rubinstein and D.P. Kroese, *The Cross-entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, Springer, New York, USA, 2004.

[16] S. Salhi and G.K. Rand (1993), *Incorporating vehicle routing into the vehicle fleet composition problem*, European J. Oper. Res. 66 (1993), pp. 313–330.

[17] K. Sebaa, H. Guéguen, and M. Boudour, *Mixed integer non-linear programming via the cross-entropy approach for power system stabilisers location and tuning*, IET Gener. Transm. Distrib. 4 (2010), pp. 928–939.

	CPLEX			Bilevel Heuristic		
Instance	Time [s]	Cost	Gap %	Av. Time [s]	Av. Cost	RO %
1	7.2×10^3	5.3239×10^4	21.10	102.94	5.2620×10^4	41.67
2	7.2×10^3	5.5372×10^4	20.65	102.67	5.5343×10^4	41.72
3	7.2×10^3	5.5273×10^4	18.40	102.91	5.6623×10^4	41.83
4	7.2×10^3	4.0153×10^4	28.00	103.81	4.1721×10^4	41.42
5	7.2×10^3	4.2752×10^4	23.94	102.42	4.0085×10^4	40.34
6	7.2×10^3	4.2615×10^4	16.50	99.07	4.5745×10^4	41.46
7	7.2×10^3	4.2728×10^4	21.50	99.01	4.3827×10^4	41.53
8	7.2×10^3	4.6420×10^4	19.50	100.75	4.8419×10^4	42.23
9	7.2×10^3	4.1085×10^4	18.50	104.99	4.0503×10^4	41.27
10	7.2×10^3	5.2795×10^4	23.20	102.45	4.9525×10^4	42.14
11	7.2×10^3	5.2404×10^4	17.20	102.92	5.5103×10^4	42.78
12	7.2×10^3	4.6431×10^4	32.29	99.76	3.9211×10^4	40.86
13	7.2×10^3	4.8294×10^4	33.05	104.49	3.8304×10^4	40.13
14	7.2×10^3	4.3490×10^4	21.43	100.44	4.3201×10^4	42.28
15	7.2×10^3	4.5189×10^4	27.80	99.21	4.0479×10^4	41.41
16	7.2×10^3	5.2562×10^4	20.66	101.95	5.4934×10^4	42.85
17	7.2×10^3	4.9057×10^4	18.26	103.93	4.7761×10^4	42.40
18	7.2×10^3	5.3574×10^4	31.44	103.19	4.4465×10^4	41.50
19	7.2×10^3	5.8138×10^4	35.20	103.09	5.1038×10^4	43.17
20	7.2×10^3	4.6574×10^4	26.05	99.87	4.7627×10^4	42.91

Table 3. Numerical results for a suite of 20 randomly-generated large-size problems, where "RO" refers to the route overload percentage (i.e., the percentage of routes that are overloaded and need to be decomposed in our heuristic algorithm).

	CPLEX			Bilevel Heuristic	
Instance	Cost	Gap %	Feasible Routes?	Cost	Difference %
1	5.3239×10^4	21.10	No	5.2015×10^3	-
2	5.5372×10^4	20.65	No	5.4346×10^3	-
3	5.5273×10^4	18.40	Yes	5.5931×10^3	1.20
4	4.0153×10^4	28.00	Yes	4.0621×10^3	1.20
5	4.2752×10^4	23.94	No	3.9280×10^3	-
6	4.2615×10^4	16.50	Yes	4.5033×10^3	5.70
7	4.2728×10^4	21.50	No	4.2629×10^3	-
8	4.6420×10^4	19.50	Yes	4.7607×10^3	2.60
9	4.1085×10^4	18.50	No	3.9642×10^3	-
10	5.2795×10^4	23.20	No	4.8676×10^3	-
11	5.2404×10^4	17.20	Yes	5.4160×10^3	3.40
12	4.6431×10^4	32.29	No	3.8168×10^3	-
13	4.8294×10^4	33.05	No	3.7204×10^3	-
14	4.3490×10^4	21.43	No	4.2355×10^3	-
15	4.5189×10^4	27.80	No	3.9266×10^3	-
16	5.2562×10^4	20.66	Yes	5.3221×10^3	1.30
17	4.9057×10^4	18.26	No	4.6963×10^3	-
18	5.3574×10^4	31.44	No	4.3845×10^3	-
19	5.8138×10^4	35.20	No	5.0175×10^3	-
20	4.6574×10^4	26.05	No	4.6175×10^3	-

Table 4. Comparing CPLEX and the bilevel heuristic over a fixed run time of 2 hours: numerical results for the same 20 large-size problems as for Table 3.