Green Vehicle Routing Problem with Time-Varying Traffic Congestion

Yiyong Xiao and Abdullah Konak
School of Reliability and System Engineering, Beihang University, Beijing, China, xiaoyiyong@buaa.edu.cn
Information Sciences and Technology, Penn State Berks, Reading, PA 19610, konak@psu.edu

Abstract
We present a linear mixed integer programming model for the time-dependent heterogeneous green vehicle routing and scheduling problem (GVRSP) with the objective of minimizing total carbon dioxide emissions and weighted tardiness. Instead of discrete time intervals, the proposed model takes the traveled distances of arcs in different time periods as decision variables to determine the travel schedules of vehicles. We propose an exact dynamic programming method to calculate the optimal discrete departure/arriving time for the GVRSP. The dynamic programming method significantly reduces the computational complexity of the GVRSP when applying existing heuristic algorithms to solve large-sized problems. A genetic algorithm with dynamic programming (GA-DP) is developed to solve the formulated problem. Computational experiments are carried out to study the efficiency of the proposed hybrid solution approach with promising results.

Keywords CO₂ emissions, vehicle routing, mixed integer programming, dynamic programming, genetics algorithms, hybrid optimization

1. Introduction
It is well recognized that carbon dioxide (CO₂) is the major contributor of the global warming effect of the Earth during the past decades. Since it was first measured in 1958, the concentration of CO₂ in Earth’s atmosphere has been continuously increasing [27]. According to the International Energy Agency (IEA), the transportation sector, after electricity generation and heating, was the second-largest contributor of CO₂ emissions, representing 22% of the global CO₂ emissions in 2010, and almost three-quarters of the emissions from transportation were due to road transportation [1]. Traffic congestion, which results in low speeds with fluctuations on roads, often accompanied with frequent acceleration and deceleration, had greatly contributed to CO₂ emissions [2, 15]. Therefore, reducing fossil fuel consumption and CO₂ emissions due to road transportation through optimizing transportation operations is important for controlling the global warming. The objective of this paper is to develop new mathematical models and optimization methods for reducing CO₂ emissions in the context of the Vehicle Routing Problem (VRP) with traffic congestion considerations.

The Vehicle Routing Problem (VRP) has been a classic and important optimization problem in road transportation applications since its introduction by Dantzig and Ramser [6]. In general, the VRP is concerned with determining the optimal routes used by a fleet of vehicles, stationed at one or multiple depots, with the objective of fulfilling customers’ demands at minimum cost. Various versions of the VRP have been developed for different applications in the past half century, such as pickup and delivery VRP, capacitated VRP, VRP with multiple depots, VRP with time windows, split delivery VRP, time-dependent VRP,
etc. Surveys on various VRP formulations and algorithms can be found in Laporte et al. [23], Toth and Vigo [29], Golden et al. [15], and Eksioglu et al. [11].

In the recent decade, Green VRP [24, 10], which is characterized by the objective of balancing environmental and operational costs, has attracted the attention of researchers in the VRP literature. Ericsson et al. [13] identified the impact of traffic disturbances on fuel consumption and proposed a model for estimating the potential reduction in fuel consumption through route optimization. Kara et al. [20] proposed a cost function in terms of energy consumption for the VRP and named it as the Energy Minimizing VRP (EMVRP), which minimizes the total energy consumption in the route (instead of the total distance) to reduce fuel consumption and CO$_2$ emissions. Reducing CO$_2$ emissions also helps to save fuel cost because CO$_2$ emissions are usually proportional to the amount of fuel consumption [26].

Tavares et al. [28] took into account the effect of both road inclination and vehicle load on fuel consumption in waste collection operations. Kuo [21] proposed a mathematical model to calculate the total fuel consumption for the time-dependent VRP, considering both loading weight and the "non-passing" property. Figliozzi [14] proposed a partial-emissions minimizing VRP (EVRP) model to optimize the departure times of vehicles on the routes found by a time-dependent VRP algorithm. Xiao et al. [30] incorporated the fuel consumption rate resulted from a vehicle’s load (which is decreasing/increasing along the tour) into the VRP and proposed a mathematical model considering fuel consumption rate. Erdogan and Miller-Hooks [12] presented a Green VRP and developed solution techniques to aid organizations with alternative fuel vehicles working in a long driving range in conjunction with a limited refueling infrastructure. Kwon et al. [22] developed a heterogeneous fixed fleet VRP model that considers the CO$_2$ emissions trade cost in the objective function. Gaur et al. [16] studied the cumulative VRP where vehicles have finite capacity and an arbitrary number of depot offloads are allowed and proposed an approximation algorithm to minimize fuel consumption. Demir et al. [9] proposed an adaptive large neighborhood search algorithm (ALNS) to minimize the fuel consumption and the driving time with Pareto optimality.

Traffic congestion usually causes road conditions to be time-dependent, and the VRP under this case is called the time-dependent VRP (TDVRP) [25, 4, 14]. Compared with the distance-based VRP, the TDVRP considers the departure time at each node as a decision variable. Because the total travel time depends on node departure times as well as routes, the TDVRP is usually studied with an alternative objective such as minimizing the total travel time, fuel consumption, or CO$_2$ emissions [14, 21]. The TDVRP is a much more challenging problem to solve than the traditional VRP because the solution space is exponentially increased by the introduction of node departure time decisions which are often modeled as integer variables.

Recently, Bektas and Laporte [3] presented the pollution routing problem (PRP) by extending the classical VRP with time windows (VRPTW) as well as with a more comprehensive objective function including fuel, emission, and driver costs. In the PRP, the vehicle load and speed on each route segment are considered as decision variables. Demir et al. [8] developed a two-stage heuristic algorithm to solve the PRP of large-size instances with up to 200 nodes. In this two-stage approach, a large neighborhood search (LNS) heuristic is used to solve traditional VRPTW in the first stage, and the optimal speed for each arc of the route is determined optimally in the second state. Franceschetti et al. [15] extended the PRP to Time-Dependent PRP (TDP RP) by considering traffic congestion. In their approach, the planning horizon is divided into two periods: (1) the initial period of traffic congestion with a lower and constant speed and (2) the period with a speed range where the travel time linearly depends on the departure time.

In this paper, we consider a general case of the time-dependency for vehicle routing and multiple time periods for vehicle scheduling, with the objective of minimizing the total CO$_2$ emissions and tardiness. A tardiness objective is considered for the first time in the context of the green VRP. Tardiness objectives are frequently used in the scheduling literature to
model the timeliness of a service where it is impossible to satisfy all customer orders on time due to capacity limits. In such cases, the time that a customer is serviced late is penalized by a tardiness penalty coefficient representing the loss of customer goodwill. The magnitude of tardiness penalty represents the relative importance of customers. In this paper, roads are assumed to have various patterns of traffic congestion, and the scheduling horizon may have multiple periods and arbitrary departing times. The problem is formulated to optimize greenhouse gas emissions through both arcs routing and time scheduling. Therefore, it is referred to as the Green Vehicle Routing and Scheduling Problem (GVRSP). The proposed GVRSP formulation contributes to the existing body of research on the green time-dependent VRP in several ways. Compared to most green time-dependent VRP where only node departure times are considered as decision variables, in the GVRSP herein we consider: (i) multiple time periods for traveling arcs, (ii) the total traveled distances of arcs in each time period, and (iii) the optimal departure time and arrival times for arcs in each time period. In other words, we present an alternative formulation for modeling time-dependency using the total traveled distance of an arc in each period as the primary decision variable instead of the departure time of an arc. Thereby, an arc can be traveled in multiple time periods. The proposed GVRSP formulation is linear and can be solved optimally by most existing MIP solvers for small-sized problems. In this sense, the proposed formulation is a more general model for combined routing and scheduling decisions in the context of the GVRSP. In addition to a more flexible way of modeling time-dependency, we also consider multiple vehicle types in the proposed formulation. The existing GVRSP models in the literature generally assume a single vehicle type. However, CO\textsubscript{2} emissions highly depend on the vehicle type.

To solve the GVRSP efficiently, we propose a hybrid approach of a genetic algorithm (GA) and dynamic programming for the two groups of the decision variables, routing and scheduling. The GA searches for the routing and vehicle selection decisions, and dynamic programming is used to optimize the scheduling decision of the selected vehicles and routes. Therefore, the proposed hybrid approach is called the GA-DP. We demonstrate that the proposed approach can solve large-sized problem instances.

2. Problem Description and Formulation

The GVRSP herein is described as follows. A set of heterogeneous vehicles is going to visit \(n\) customers randomly located in a region, starting from and ending at a depot. Let set \(H\) be the set of vehicles and \(q\) denote the cardinality of set \(H\). Each vehicle \(h\) has a maximum capacity of \(C_h\) and a maximum travel range of \(L_h\) depending on its type. The problem is defined on a complete directed network \(G(N, A)\) where \(N = \{0, 1, \ldots, n\}\) is the node set, and \(A = \{(i, j) : i \neq j, i, j \in N\}\) is the set of arcs. Node 0 represents the depot, and \(N' = N \setminus \{0\}\) is the set of customers. Each arc \((i, j)\) represents the path between nodes \(i\) and \(j\), and the distance of arc \((i, j)\) is denoted by \(D_{ij}\). Each customer \(i\) has a demand \(R_i\) and requires a service time \(g_i\) within the time-window \([0, E_i]\) where \(E_i\) is the due-time of customer \(i\) such that a late visit will result in a tardiness penalty \(\omega_i\) per unit time tardy.

The time horizon is divided arbitrarily into \(m\) time periods such that the traffic speeds on roads are assumed to be fixed within a time period but may be different between two time periods. Let set \(K\) represent the set of time periods, and each period \(k \in K\) is identified by its beginning time \(b_k\) and ending time \(e_k\). The travel speed of vehicles on arc \((i, j)\) in time period \(k\) \((v_{ijk})\) is also assumed to be known, and the CO\textsubscript{2} emissions rate (Lb/mile) of vehicle \(h\) when traveling on arc \((i, j)\) in time period \(k\) \((c_{ijkh})\) can be calculated using emissions models from the literature. Note that all road and vehicle conditions such as speed, slope, traffic congestion, vehicle weight, can be factored into the calculation of \(c_{ijkh}\).

The problem involves four decisions for each vehicle as follows: (1) whether to use a vehicle or not, (2) the route to visit customers if a vehicle is used, (3) the departure and arrival time schedule for each node in the tour, and (4) the distances to be traveled on each arc in each period.
time period. The objective of the problem is to minimize the total amount of CO$_2$ emissions and tardiness penalties. Comparing to the traditional TD-VRP, the GVRSP formulated in this paper is more general as idle time can be scheduled anywhere in the tour to avoid traffic congestions. The decision variables of the model are as follows:

- $X_{ij}$ binary variable indicating whether arc $(i, j)$ is traveled ($X_{ij} = 1$) or not ($X_{ij} = 0$)
- $y_{ijh}$ binary variable indicating whether arc $(i, j)$ is traveled by vehicle $h$ (in any period) ($y_{ijh} = 1$) or not ($y_{ijh} = 0$)
- $x_{ijkh}$ binary variable indicating whether arc $(i, j)$ is traveled by vehicle $h$ in period $k$ ($x_{ijkh} = 1$) or not ($x_{ijkh} = 0$)
- $d_{ijkh}$ continuous variable indicating the traveled distance of arc $(i, j)$ in time period $k$ by vehicle $h$
- $\tau_{ijkh}$ continuous variable indicating the travel time of vehicle $h$ on arc $(i, j)$ in time period $k$
- $l_i$ continuous variable indicating the departure time from node $i$ (the earliest departure time for $i = 0$)
- $a_i$ continuous variable indicating the arrival time at node $i$ (the latest arrival time for $i = 0$)
- $O_i$ tardiness of node $i$

**Problem GVRSP:**

Minimize $F = \sum_{h=1}^{q} \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} d_{ijkh} \cdot c_{ijkh} + \sum_{i=1}^{n} \omega_i \cdot O_i$

Subject to:

1. $\sum_{j=0}^{n} X_{ij} = 1 \quad \forall i \in N'$
2. $\sum_{i=0}^{n} X_{ij} = 1 \quad \forall j \in N'$
3. $X_{ij} = \sum_{h=1}^{q} y_{ijh} \quad \forall (i, j) \in A$
4. $y_{ijh} \geq x_{ijkh} \quad \forall (i, j) \in A, k \in K, h \in H$
5. $y_{ijh} \leq \sum_{k=1}^{m} x_{ijkh} \quad \forall (i, j) \in A, h \in H$
6. $\sum_{i=0: i \neq j}^{n} y_{ijh} = \sum_{i=0: i \neq j}^{n} y_{jih} \quad \forall h \in H, j \in N$
7. $\sum_{j=0}^{n} y_{b0j} \leq 1 \quad \forall h \in H$
8. $d_{ijkh} \leq D_{ij} \cdot x_{ijkh} \quad \forall (i, j) \in A, k \in K, h \in H$
9. $\sum_{k=1}^{m} d_{ijkh} = X_{ij} \cdot D_{ij} \quad \forall (i, j) \in A$
10. $\tau_{ijkh} = 60 \cdot d_{ijkh} / v_{ijk} \quad \forall (i, j) \in A, k \in K, h \in H$
11. $\sum_{(i, j) \in A} \tau_{ijkh} \leq e_k - b_k \quad \forall k \in K, h \in H$
12. $l_i \leq e_k - \tau_{ijkh} + e_m \cdot (1 - x_{ijkh}) \quad \forall (i, j) \in A, k \in K, h \in H$
$$a_j \geq b_k + \tau_{ijk} - e_m \cdot (1 - x_{ijk}) \quad \forall (i,j) \in A, k \in K, h \in H$$  \hspace{1cm} (13)

$$a_j \geq l_i + \sum_{k=1}^{m} \sum_{h=1}^{q} \tau_{ijk} - e_m \cdot (1 - X_{ij}) \quad \forall (i,j) \in A$$  \hspace{1cm} (14)

$$a_i + g_i \leq l_i \quad \forall i \in N'$$  \hspace{1cm} (15)

$$a_0 \leq e_m$$  \hspace{1cm} (16)

$$\sum_{j=1}^{n} \sum_{i=0}^{n} R_j \cdot y_{ijh} \leq C_h \quad \forall h$$  \hspace{1cm} (17)

$$\sum_{(i,j) \in A} D_{ij} \cdot y_{ijh} \leq O_h \quad \forall h$$  \hspace{1cm} (18)

$$O_i \geq a_i - E_i \quad \forall i \in N'$$  \hspace{1cm} (19)

In Problem GVRSP, Constraints (1) and (2) ensure that each customer is visited only once. Constraint (3) states that a selected arc can be traveled only by one vehicle. Constraints (4) and (5) force variable $y_{ijh}$ to be consistent with variable $x_{ijk}$. Constraint (6) is the node balance constraint for each vehicle, ensuring that each vehicle enters and departs from each node in equal numbers of times. Constraint (7) indicates that each vehicle leaves the depot at most only one time (i.e., a vehicle cannot be used again after it returns back to the depot). Constraints (8) ensure that $d_{ijk}$ becomes zero if $x_{ijk} = 0$ and $d_{ijk}$ becomes bounded by $D_{ij}$ by Constraints (9). Constraint (10) guarantees that the total distance of arc $(i,j)$ must be traveled once the arc is selected. Constraint (11) calculates the travel time of $d_{ijk}$ (the travel time is measured in seconds, and the travel speed unit is miles-per-hour). Note that Constraint (11) is linear because $v_{ijk}$ is a calculated parameter. Thereby, it is possible to express the non-linear relationship between CO$_2$ emissions and the travel-speed as linear constraints without requiring auxiliary binary variables for linearization. Constraint (12) ensures that the total travel time of each vehicle in each time period is less than the duration of the time period. Constraint (13) ensures that if arc $(i,j)$ is traveled in time period $k$, the departure time of node $i$ must be before the end time of time period $k$ subtracted by the travel time of arc $(i,j)$ in time period $k$. Similarly, Constraint (14) states that the arrival time of node $j$ must be greater than or equal to the start of time period $k$ plus the travel time of arc $(i,j)$ in time period $k$. These two constraints are only active for $x_{ijk} = 1$. Note that for $x_{ijk} = 0$, Constraints (12) and (13) still hold because $e_m$ is a large enough number that guarantees the validity of the inequities. Constraint (15) is a disjunctive constraint to calculate the earliest arrival time at node $j$. Note that Constraint (15) is only active for the arcs that are selected in a tour (i.e., $X_{ij} = 1$). Constraint (16) ensures that the vehicle’s departure time must be after its arrival time plus the service time on each customer node $i$. Constraints (14) and (15) also eliminate any sub-tours among customers. Constraint (16) requires that the return time to the depot must not exceed the end time of the last period. Constraints (17) and (18) are for vehicles’ capacity and travel length limits, respectively. Constraint (19) is a linear expression for calculating the tardiness value.

It should be noted that although there are 10 different types of decision variables in Problem GVRSP, most of them are tightly bounded (directly or indirectly) to independent decision $d_{ijk}$ variables. In addition, all mathematical expressions in the model are linear. Therefore, it is possible to optimally solve Problem GVRSP using existing off-the-shelf solvers. In the experimental section, we used AMPL/CPLEX (version 12.4.0.1) to optimally solve Problem GVRSP for small-sized problem instances.
3. Description of the Hybrid Genetic Algorithm and Dynamic Programming

Problem GVRP can be optimally solved only for small-sized problems. In this section, we propose an exact dynamic programming (DP) procedure to calculate the optimal departure and arrival time of each vehicle at each node for a given set of tours. Since the DP method has polynomial computational complexity, the complexity of the GVRSP can be reduced significantly by dividing the problem in two parts, routing and scheduling. One of the challenges of applying meta-heuristic algorithms to solve the GVRSP is devising an encoding schema to effectively represent the problem’s binary and continuous decision variables. Note that the continuous schedule decision variables depend on the binary route decision variables. Therefore, when the routes of a solution are perturbed by local or global search operators, the schedule decision variables must also be modified accordingly. Otherwise, the structure of the solution will be destroyed. This makes the GVRSP difficult to solve by meta-heuristics. If the problem is solved in two phases as routing and scheduling as proposed in this paper, the solution encoding is significantly simplified because the schedule decision variables can be determined by the DP procedure for a given set of routes.

3.1. Solution Encoding

In the GA, the routes of a solution are encoded as an ordered-string vector. We use $S^h = \{s^h_p : p = 1, 2, ..., n^h - 1\}$ to represent the sequence of the customer nodes visited by vehicle $h$, starting from and returning back to depot 0, where

- $n^h$ the number of arcs traveled by vehicle $h$ (e.g., if a vehicle $h$ visits only one customer and returns back to the depot, then $n^h = 2$),
- $s^h_p$ the index of the customer node at the $p$th position of vehicle $h$’s tour,
- $p$ node index of a vehicle’s tour.

For a problem with a maximum of $q$ vehicles, $S$ is constructed as $S = \{0 + S^1 + 0 + S^2 + 0 + ... + 0 + S^q + 0\}$ with zeroes between the route strings representing the start of a new tour. For example in a problem with three vehicles and five customers, if a solution has $S^1 = \{3, 5, 2\}$, $S^2 = \{\}$, $S^3 = \{1, 4\}$, then the solution in the GA is represented as $S = \{0, 3, 5, 2, 0, 0, 1, 4, 0\}$.

3.2. Dynamic Programming

Let $r$ denote the index of arcs traveled by a vehicle for $r = 1, 2, ..., n^h$ (e.g., $(s^h_0, s^h_1)$ is the first arc traveled by vehicle $h$). For each vehicle $h$, we use integer variables $l^h_r$ and $a^h_r$ to represent the departure time (vehicle $h$ starts traveling the $r$th arc of $S^h$) and the arrival time (vehicle $h$ finishes traveling the $r$th arc of $S^h$) of arc $r$, respectively. Note that in Problem GVRSP, variables $l_i$ and $a_i$ are defined in the continuous domain, but they are discretized as integer variables for DP. If the time unit is small enough, the difference between the schedules based on continuous and discrete times will be very small and can be ignored in real-life applications.

In this section, we introduce a DP procedure to determine the optimal departure time $l^h_r$ and arrival time $a^h_r$ for a given $S^h$. The planning horizon is divided into $T$ integer units of time, $t = 0, 1, 2, ..., T$, which represent the discretized time points when the vehicle can depart from a node and arrive at the next one according to route $S^h$. For a given $S^h$, the DP model has $n^h$ stages. In stage $r$, the state of vehicle $h$ is defined by $t$ and $t'$, representing the departure and arrival times of the $r$th arc such that $t, t' \in [0, T]$ and $t \leq t'$. For vehicle $h$, we use the following notations in the DP model.
For all used vehicles, and the total optimal objective value of all used vehicles is given as

Using the same procedure, the optimal departure times and arriving times can be calculated

A pre-calculated matrix for each individual sub-tour of a candidate solution as it can be obtained by directly from

The worst case computational complexity of the dynamic programming procedure is $O((nT)^2)$. For each sub-tour $h$ of a solution, the calculation of $z_{rtt'}^h$ is computationally expensive although it can be done in polynomial time. Fortunately, the calculation of $z_{rtt'}^h$ can be omitted using a pre-calculated matrix $Z_{ijtt'}^h$—the optimal objective function value of traveling arc $(i,j)$ by vehicle $h$ if the vehicle departs from node $i$ at time $t$ and arrives at node $j$ at time $t'$. In the actual implementation of the DP procedure, $z_{rtt'}^h$ is not calculated for each individual sub-tour of a candidate solution as it can be obtained by directly from a pre-calculated matrix $Z_{ijtt'}^h$. In Algorithm 1, the procedure for pre-calculating $Z_{ijtt'}^h$ is provided. Note that this procedure finds the optimum solution for the linear programming model given above. The procedure always selects the period that leads to minimum CO$_2$ emissions for the vehicle to travel. In Algorithm 1, Line 1 calculates the length of the time
interval that \([t, t']\) and \([b_k, e_k]\) overlap for each time period \(k\). Between Lines 3 and 17, vehicle \(h\) travels arc \((i, j)\) on the time periods with minimum possible CO\(_2\) emissions. In Lines 18 and 19, the tardiness penalty part of the objective function is calculated. A drawback of this approach is that matrix \(Z_{ijtt'}^h\) can be very large. Nonetheless, matrix \(Z_{ijtt'}^h\) is also a sparse matrix which can be efficiently stored in the computer memory using sparse matrix storage techniques. As shown in the computational experiment section, large-sized problems can be solved efficiently.

Algorithm 1 Calculation the optimal objective value for \(Z_{ijtt'}^h\)

```
1: Set \(\mu_{ktt'} ← \max(0, \min(e_k, t') - \max(b_k, t)) \forall k \in K\)
2: \(obj ← 0, dis ← D_{ij}\)
3: while \(dis > 0\) do
4: \(k^* ← \arg \min_{k \in K}(c_{i j k h} : \mu_{ktt'} > 0)\)
5: if \(k^*\) does not exists then return \(-1\)
6: end if
7: if \(dis > v_{i j k h} \cdot \mu_{ktt'} / 60\) then
8: \(d_{i j k h} ← v_{i j k h} \cdot \mu_{ktt'} / 60\)
9: \(obj ← obj + c_{i j k h} \cdot d_{i j k h}\)
10: \(\mu_{ktt'} ← 0\)
11: \(dis ← dis - d_{i j k h}\)
12: else
13: \(d_{i j k h} ← dis\)
14: \(obj ← obj + c_{i j k h} \cdot dis\)
15: \(dis ← 0\)
16: end if
17: end while
18: if \(t' > E_{j(r)}\) then
19: \(obj ← obj + \omega_{j(r)} \cdot (t' - E_{j(r)})\)
20: end if
21: Return \(obj\)
```

The following two properties on \(Z_{ijtt'}^h\) and \(f_{rt}^h\) can be used to further improve the computational efficiency of the DP procedure further.

**Property 1** \(Z_{ijtt'}^h\), if exists, is a monotonically increasing function of \(t\) as follows:

\[
Z_{ijtt'}^h ≥ Z_{ijtt''}^h \quad \forall h, i, j, t' < t_2
\]  

**Property 2** For a tour \(S^h\), \(f_{rt}^h\), if exists, is a monotonically increasing function of the departure time \(t\) of arc \(r\) as follows:

\[
f_{rt}^h ≤ f_{rt'}^h \quad \forall h, r \quad \text{and} \quad t_1 < t_2
\]  

Using the above two properties, for each arc \(r\) in \(S^h\), the earliest arrival time \(t^*\) (denoted as \(t^*_1\)) that leads to optimal \(z_{rt}^h\) can be pre-calculated for each departure time \(t\) in \([0, T]\). In addition, the latest departure time (denoted as \(t^*_2\)) that leads to optimal \(f_{rt}^h\) can be predetermined in the previous calculation for arc \(r + 1\). Therefore, \(f_{rt}^h\) can be calculated more efficiently using the following recursive backward equation:

\[
f_{rt}^h = \begin{cases} 
    z_{rt}^h & \text{if } t^*_1 ≤ t^*_2 \text{ or } r = n^h \\
    \min(z_{rt}^h + f_{(t+1)t'}^h : t' ≤ t^*_1) & \text{if } t^*_1 > t^*_2 \text{ and } r < n^h
\end{cases}
\]  

\[
(25)
\]
Thus, the calculation of $f_{rt}^h$ can be either obtained directly from pre-calculated matrix (for the cases $t_1^r \leq t_2^r$ or $r = n^h$) or from the minimum $z_{rt}^h + f_{r+1,t}^h$ such that $t'$ is within $[t_2^r, t_1^r]$. This approach further simplifies the computational complexity of the DP procedure. Note that if $t_1^r \leq t_2^r$, then the vehicle can have an idle time of $t_2^r - t_1^r$ before starting to travel the $r$th arc in $S^h$, which does not impact the objective function value. If $f_{rt}^h$ cannot be computed for $t = 0$ (i.e., the $r$th arc cannot be traveled before completing the route within $[0, T]$), then the route is infeasible. For this case, in order to include infeasible solutions in the evolution process of the GA, we set $f_{rt}^h = \beta \cdot \max\{z_{rt}^h, \forall t, t\}' + f_{(r+1)t}^h$ for the $r$th arc and all arcs before it in $S^h$, where $\beta$ is a dynamic penalty coefficient which is introduced in the following section. In other words, the worst objective value of traveling the $r$th arc is used as a penalty term if it cannot be traversed completely.

3.3. Description of the Genetic Algorithm

3.3.1. Solution Initialization. We use a random construction heuristic to initialize the population as follows: (1) initialize $S$ with only $g + 1$ zeros (vehicles), i.e., $S = 0, 0, 0, \ldots, 0$, (2) find a customer $i$ who has the least increase in the objective function value and insert it into a position $p$ in $S$, (3) repeat step 2 until all customers are inserted into $S$, (4) use Insert and Swap operators to randomly perturb the tour. After $S$ is initialized, the DP procedure is called to calculate the schedule of the vehicles.

3.3.2. Genetic Operators. Three genetic operators, which are Insert, Swap, and Crossover, are used in turn to reproduce new routes from the existing ones in the population. The Insert operator relocates a randomly selected node to a randomly selected position in $S$. The Swap operator swaps the positions of two randomly selected nodes in $S$.

The Crossover operator is a one-point partially matched crossover \[17\], which recombines two randomly selected parents to produce two offspring. Each of the two offspring inherits an unchanged-part from one parent and a re-sequenced-part from the other parent as shown in Figure 1. The unchanged part is transferred from one parent to an offspring exactly. The genes of the re-sequenced-part are transferred to the offspring after rearranging them according to the sequence of the same genes in the other parent. An example of the crossover operator is given in Figure 1 where Parent 1=$\{0, 3, 5, 2, 0, 1, 4, 0\}$ and Parent 2=$\{0, 2, 4, 0, 1, 3, 0, 5, 0\}$ are recombined at a point $p$ to generate two offspring $\{0, 2, 3, 4, 0, 0, 1, 4, 0\}$ and $\{0, 2, 4, 0, 3, 5, 0, 1, 0\}$. In the figure, the genes indicated by italic typefaces in Parent 1 and Parent 2 are the genes in one parent that have the same sequence in the other parent, and deleting them produces the re-sequenced part of the offspring.

The three operators are used in turn to reproduce a new population of solutions. Therefore, the size of the population expands to $5\lambda$ from $\lambda$ after all operators are applied.

3.3.3. Fitness Calculation and Selection. The fitness of solution $i$ is defined as follows:

$$P_i = \exp\left(-\frac{F_i - F_{\min}}{F_{\text{avg}} - F_{\min}}\right)$$

where $F_i$ is solution $i$’s objective function value calculated by the DP procedure, and $F_{\min}$ and $F_{\text{avg}}$ are respectively the minimum and average objective function values in the population. $P_i$ is the probability for solution $i$ to stay in the population during the selection procedure. For a solution with the average objective function value, the fitness probability is $\exp(-1) \approx 0.37$.

Selection is the procedure of reducing the size of the expanded population (due to reproduction by mutation and crossover) to a fixed size by deleting individuals based on their fitness probabilities. To maintain the diversity of the population, we use the clustering method introduced by Dellaert and Jeunet \[7\] to group individuals with identical objective values as one cluster and retain at most 10 individuals in the best cluster and only one individual in any other clusters. Therefore, an individual can be removed from the population either because it is not fit enough or because of the size of its cluster.
3.3.4. Dynamic Penalty Function  As discussed earlier, it is possible that infeasible solutions appear in the population because routes cannot be completed within the planning horizon or vehicles’ capacity and/or travel distance limits might be exceeded. In heuristic algorithms, infeasible solutions are often allowed because they smooth the solution space and may help to connect current solutions toward better ones. In addition, optimal solutions often reside on the boundaries of infeasible and feasible regions of the solution space. In the GA-DP, infeasible solutions are allowed in the population, but their ratio in the population is controlled using a dynamic penalty coefficient as follows:

\[
\beta = \begin{cases} 
1.1\beta & \text{if } \lambda^*/\lambda < 0.85 \\
(1.1)^{-1}\beta & \text{if } \lambda^*/\lambda > 0.95
\end{cases}
\]

where \(\lambda\) is the population size, \(\lambda^*\) is the number of feasible solutions, and \(\beta\) is the dynamic penalty coefficient. The dynamic penalty coefficient is calculated after the selection process in order to keep the feasibility ratio of the population within \([0.85, 0.95]\). If the feasibility ratio is lower than 0.85, then the penalty coefficient \(\beta\) is increased, and if it is greater than 0.95 then it is decreased. The maximum and minimum values for \(\beta\) are restricted within \([0, 100]\). In Algorithm 2, the pseudo code of the GA-DP is given. Parameter \(g\) is the stopping criteria (the number of consecutive generations without improvement in the best feasible solution).

Algorithm 2 Framework of the GA-DP

Randomly generate \(\lambda\) solutions and evaluate them using the DP procedure

while A new best feasible solution has been found in the last \(g\) iterations do
    Call Insert(), Swap(), and Crossover() operators to reproduce offspring
    Call the DP procedure to determine the schedules and finesses of the offspring
    Combine the parent and offspring populations
    Call Selection to select \(\lambda\) solutions for the next population
    Update the penalty coefficient \(\beta\)
end while
Return the best feasible solution

4. Computational Experiments

First, we used 30 small-sized problem instances with 5, 10, and 15 customers served by different types of heavy-duty vehicles to benchmark the performance of the GA-DP algorithm.
with respect to the optimal solutions found using Problem GVRSP. The problem instances were generated to simulate a general case of time-dependent road conditions. The length of the planning horizon was 150 minutes and divided into five 30-minute time periods. The customers were located randomly in an area of $20 \times 20$ square miles, and the average travel speeds of the roads in different time periods were random numbers in $[15, 75]$ mph. Four types of vehicles were available to serve customers, and the number of available vehicles of each type was increased for each additional group of five customers. The customers’ demand was zero and vehicles’ capacity and travel length were unlimited in this first set of computational experiments. The CO$_2$ emission rates (LB/mile) with respect to four different type of vehicles under different travel speeds were calculated using the CO$_2$ emissions model developed by Hickman [19] as shown in Figure 2.

![Figure 2. CO$_2$ emissions rates and travel speeds for heavy-duty vehicles to generate test problems.](image)

We solved the 30 small-sized instances using Problem GRVSP with AMPL/CPLEX (version 12.4.0.1) in an iMac computer equipped with 2.50GHz Intel Core i5-2400S CPU and then solved them using the GA-DP in a PC computer equipped with CPU Intel Core i5-3571 @ 3.4GHz and the MS Window 7 system. The GA-DP was coded in Visual C++ 6.0. The population size was 50, and the stopping condition was 50 in all runs. The results and comparisons are given in Table 1. In order to study the impact of the discretization of continuous decision variables in the DP procedure, the test problems were also solved for three different cases of departure/arrival time decision variables: (i) Continuous, (ii) Discrete with one-minute unit of time, and (iii) Discrete five-minute unit of time. The solutions under column CPLEX were obtained by solving Problem GVRSP using AMPL/CPLEX. For cases (ii) and (iii), the continuous decision variables $a_i$ and $l_i$ were respectively redefined as integer variables. Column Gap is the optimality gap of the best solution found by AMPL/CPLEX in 3600 CPU seconds (note that 0% gap indicates the best solution is optimal). It can be observed that the solutions found by the discretized versions of Problem GVRSP are slightly worse than those found by the continuous version for the problem groups $5 \times 5 \times 150$ and $5 \times 10 \times 150$ for which optimum or close to optimum solutions could be found in 3600 CPU seconds, indicating some degradation in solution quality when the departure and arrival times are discretized. Column Dev% under CPLEX in cases (ii) and (iii) indicates the percentage degradation of solution quality compared to the solutions found in case (i). As expected, the optimal objective function value increased as a larger unit of time was used.
in the discretization process for the problem groups $5 \times 5 \times 5$ and $5 \times 10 \times 10$. In addition, Dev\% increased with the increasing problem size. Another observation is that Problem GVRSP required significantly shorter CPU times with the continuous departure/arrival time decision variables than those with discrete ones. This is an advantage of the developed modeling framework in this paper compared to the other models that use the discrete time domain in the literature. Unfortunately, Problem GVRSP cannot be solved optimally for larger sized problems. For problem group $5 \times 5 \times 5$, AMPL/CPLEX was not able to find optimal solutions in $3600$ CPU seconds. Interestingly, some of the solutions found using a larger unit of time discretization in AMPL/CPLEX were better than those found using a smaller unit of time discretization. For example, in instance 5 of problem group 5, the best objective function values were 169.19 and 148.92 with one and five-minute unit of time discretization, respectively. Because a larger unit of time discretization reduces the time requirement of the GA-DP reduced as a larger unit of time discretization was used, the average solution quality was degraded. In summary, the results in Table 1 indicate that the GA-DP is very effective in finding good solutions in very short times. However, caution should be used to select a proper unit of time discretization.

In Table 1, the GA-DP was able to find the optimal solutions in all 10 runs of problem groups $5 \times 5 \times 5$ and $5 \times 10 \times 10$. For problem group $5 \times 5 \times 5$, the GA-DP was able to find better solutions than those found by AMPL/CPLEX (21\% lower objective function value on the average) in much shorter time (6.7 seconds on the average). While the CPU time requirement of the GA-DP reduced as a larger unit of time discretization was used, the average solution quality was degraded. In summary, the results in Table 1 indicate that the GA-DP is very effective in finding good solutions in very short times. However, caution should be used to select a proper unit of time discretization.

Next, we tested the efficiency of the GA-DP using the 14 well-known CVRP benchmark instances from Christofides and Eilon [9], where the number of total nodes ranges from 51 to 200 and with constraints on vehicles capacity, maximum route length. We changed the original problem instances to be time-dependent by (1) adding a time horizon $[0, 300]$ in which the customer service must be fulfilled, (2) associating each arc with a random traffic speed between $[15, 75]$ for each time period, (3) assigning each customer with a service...
time-window \([0, T_i]\) and a tardiness penalty coefficient in \([0.5, 1.5]\), and (4) using the four types of heavy-duty vehicles as shown in Figure 2 to estimate the CO\(_2\) emission rates. If an instance required more than four vehicles, then the fifth vehicle is of the first type and sixth vehicle will be of the second type, and so on. The time horizon \([0, 300]\) was discretized with five-minute time intervals. The GA-DP algorithm was run with population size \(\lambda = 100\) and stopping condition \(g = 100\). To examine the effect of the dynamic penalty function, we first ran a rigid GA-DP algorithm where only feasible solutions were allowed in the population (denoted as 10 runs of GA-DP*), and then ran the GA-DP algorithm with the dynamic penalty function (denoted as 10 runs of the GA-DP). The computational results are listed in Table 2. Column \(\text{Ava. Veh.}\) indicates the number of available vehicles, column \(\text{Used Veh.}\) indicates the number of vehicles used in the best solution. The comparison shows that the solution quality of the GA-DP is better that that of GA-DP* in terms of both average and minimum objective function values. The GA-DP* could not even find feasible solutions for problem No. 9. The results also show that the GA-DP is computationally efficient as all the problems were solved in reasonable CPU times, i.e, from 1 minute to 18 minutes. Nevertheless, the GA-DP was able to solve problems with 200 nodes and 60 discrete time points. Note that since the formulated problem is studied for the first time, we cannot compare the results with previous one from the literature.

### Table 2. Experiments on 14 instances in Christofides and Eilon [5]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50 \times 5 \times 60</td>
<td>7</td>
<td>7 160</td>
<td></td>
<td>1187.1</td>
<td>1148.9</td>
<td>31\text{a}</td>
<td>1189.5</td>
<td>1099.1</td>
<td>51\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>75 \times 5 \times 60</td>
<td>13</td>
<td>11 140</td>
<td></td>
<td>1824.7</td>
<td>1696.9</td>
<td>79\text{b}</td>
<td>1728.0</td>
<td>1615.6</td>
<td>159\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100 \times 5 \times 60</td>
<td>12</td>
<td>10 200</td>
<td></td>
<td>2038.0</td>
<td>1925.9</td>
<td>182\text{a}</td>
<td>2108.7</td>
<td>1953.8</td>
<td>177\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>150 \times 5 \times 60</td>
<td>16</td>
<td>14 200</td>
<td></td>
<td>2724.2</td>
<td>2601.2</td>
<td>444\text{a}</td>
<td>2772.2</td>
<td>2555.0</td>
<td>565\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>199 \times 5 \times 60</td>
<td>20</td>
<td>18 200</td>
<td></td>
<td>3429.5</td>
<td>3353.0</td>
<td>452\text{c}</td>
<td>3444.1</td>
<td>3115.0</td>
<td>1055\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>50 \times 5 \times 60</td>
<td>7</td>
<td>6 160 200</td>
<td></td>
<td>1296.7</td>
<td>1245.6</td>
<td>54\text{c}</td>
<td>1251.7</td>
<td>1178.5</td>
<td>44\text{c}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>75 \times 5 \times 60</td>
<td>13</td>
<td>11 140 160</td>
<td></td>
<td>1884.3</td>
<td>1756.9</td>
<td>109\text{a}</td>
<td>1705.0</td>
<td>1631.4</td>
<td>160\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>100 \times 5 \times 60</td>
<td>11</td>
<td>9 200 230</td>
<td>2088.0</td>
<td>1965.5</td>
<td>149\text{a}</td>
<td>1939.0</td>
<td>1814.0</td>
<td>199\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>150 \times 5 \times 60</td>
<td>15</td>
<td>13 200 200</td>
<td></td>
<td>2794.2</td>
<td>2536.8</td>
<td>54\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>199 \times 5 \times 60</td>
<td>21</td>
<td>18 200 200</td>
<td></td>
<td>3178.4</td>
<td>3099.5</td>
<td>554\text{a}</td>
<td>3210.3</td>
<td>3028.7</td>
<td>628\text{a}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>120 \times 5 \times 60</td>
<td>11</td>
<td>9 200</td>
<td>3905.1</td>
<td>3594.6</td>
<td>279\text{a}</td>
<td>3943.8</td>
<td>3700.2</td>
<td>322\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>100 \times 5 \times 60</td>
<td>12</td>
<td>10 200</td>
<td>2034.0</td>
<td>1795.6</td>
<td>153\text{a}</td>
<td>2088.5</td>
<td>1773.8</td>
<td>134\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>120 \times 5 \times 60</td>
<td>12</td>
<td>10 200 720</td>
<td>10304.2</td>
<td>4007.5</td>
<td>250\text{b}</td>
<td>6019.9</td>
<td>3879.4</td>
<td>313\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>100 \times 5 \times 60</td>
<td>22</td>
<td>10 200 1040</td>
<td>2047.8</td>
<td>1861.4</td>
<td>227\text{b}</td>
<td>1958.0</td>
<td>1809.8</td>
<td>347\text{a}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>2919</td>
<td>2311</td>
<td>227</td>
<td>2582</td>
<td>2263</td>
<td>336</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 5. Conclusions

We studied a time-dependent vehicle routing and scheduling problem, where a fleet of heterogeneous vehicles are to service a number of customers to minimize CO\(_2\) emissions and tardiness, with constraints on time-window, capacity and travel length. We developed a linear mixed-integer mathematical model of the problem. The model optimizes both the routes and the schedules (in different time periods) for a number of vehicles of mixed types so that idle times on a tour can be scheduled optimally to avoid traffic congestion and reduce CO\(_2\) emissions.

For large-sized problems, we presented an exact dynamic programming method to optimize the scheduling part of the problem. The dynamic programming method can find schedules for all arcs traveled by the vehicles with optimal discrete departure/arriving time and continuous travel distance in different periods. We also developed methods to improve the efficiency of the dynamic programming method. We should mention that with the dynamic programming procedure, many existing heuristic algorithms for VRPs can be applied to solve the GVRSP. We only presented a GA to demonstrate how a very difficult problem in terms of solution encoding can be reduced to a workable structure for heuristics using a hybrid approach. We should also mention that the dynamic programming method has two drawbacks: (1) it may lead to sub-optimal solutions because it depends on discrete departure/leaving times (averagely 1.5% in comparison to the LMIP model) and (2) it has a large...
memory requirement. However, these drawbacks will be insignificant for real-life applications. We developed a genetic algorithm with dynamic programming to solve the GVRSP with near-optimal solutions.

Acknowledgement

This work is partly supported by the National Natural Science Foundation of China under Grants No. 71271009.

References


