# Optimizing of ready-mixed concrete vehicle scheduling problem by hybrid heuristic algorithm

**Guochen Zhang[1]*, Jianchao Zeng[2]**

[1] *College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou, 730050, China*

[2] *Division of Industrial and System Engineering, Taiyuan University of Science and Technology, Taiyuan, 030024, China*

*Received 1 June 2014, www.cmnt.lv*

**Abstract**

RMC (Ready-mixed concrete) vehicle scheduling problem is a complex combinatorial optimization problem with intersection of several research areas, such as logistics, just-in-time production and supply chains etc. We integrates RMC production scheduling and vehicle dispatch problems in the same framework by network flow techniques and establishes a mixed integer programming model for ready-mixed concrete vehicle scheduling problem. Then, we put forward a hybrid heuristic algorithm to optimize the RMC vehicle scheduling problem based on the characteristics of ready-mixed concrete vehicle fleet operations. The main idea of proposed hybrid heuristic algorithm is decomposing complex RMC vehicle scheduling problem into simple problems. Finally, the algorithm is used to solve specific simulation examples and to verify the effectiveness of the proposed hybrid heuristic algorithm.

*Keywords:* RMC Vehicle Scheduling, Hybrid Heuristic Algorithm, Network Flow Technique, Mixed Integer Programming Model

## 1 Introduction

The scheduling of RMC vehicles is a complex combinatorial optimization problem including: vehicle sequence, the loading sequence in the depot, the time demands imposed by the construction industry, just-in-time manufacturing issues, and other problems. The scheduling environment for ready-mixed concrete vehicles is dynamic due to these uncertainties in transportation time, the uncertainty in time and quantity demanded by customer, the dynamic emergence of new customer demands, vehicle malfunction, depot malfunction, pump malfunction, weather variations, etc. [1]. To respond to these dynamic factors in the scheduling process, a rapid and efficient scheduling algorithm is needed. In this paper, we proposed a heuristic genetic algorithm for the scheduling of RMC vehicles.

There are many researches focus on different emphases of RMC scheduling problem. Tommelein analyzed the production and delivery characteristics of RMC vehicle scheduling problem, indicated it belonged to a special class of supply chain management problems [2]. Matsatsinis proposed a support decision system which was basically a multi-depot vehicle routing problem with time windows [3]. Feng studied a concrete delivery problem with a single batch plant; it is solved by genetic algorithm [4]. Naso proposed a non-liner model for RMC production and delivery problem with muti-depots and muti-sides. A two-phase approach based on genetic algorithms is used for optimization [5]. Yan integrated RMC production scheduling and vehicle dispatch problems in the same framework by network flow techniques. A method based on CPLEX was used for optimization [6-8]. Asbach proposed a mixed integer programming model for RMC scheduling problem based on the network flow method. The neighborhood search method based on heuristic was used for optimization [9].

So far, most of the researchers focus their attention on static scheduling of RMC vehicle [10-17]. The scheduling environment for RMC vehicles is dynamic. To respond to the dynamic factors in the scheduling process, rescheduling is needed since previously formulated plans may not then be performed on time, and a fast and efficient algorithm can be used for RMC companies to get the optimized scheduling scheme. In this paper, according to the characteristic of RMC scheduling problem, we integrates RMC production scheduling and vehicle dispatch problems in the same framework by network flow techniques and establishes a mixed integer programming model for RMC vehicle scheduling problem. Then we proposed a hybrid heuristic algorithm to optimize the RMC vehicle scheduling problem.

The rest of this paper is organized as follows: Section 2 provides a detailed description of the problem; Section 3 a programming model is established to solve RMC vehicle scheduling problem based on the network flow mode; Section 4 introduces the heuristic genetic algorithm; Section 5 applies the algorithm to specific instance and analyses the simulation results; and Section 6 draws the necessary conclusions.

## 2 RMC vehicle scheduling problem

At the start time of scheduling, there is a depot denoted by $D$, which has a vehicle fleet $K = \{K_1, ..., K_r\}$ for delivering concrete to customers. All vehicles in $K$ present same type and loading capacity, each vehicle starts its journey on the considered working day at the depot and

* ***Corresponding author's*** e-mail: imzgc@hotmail.com

has to return to the depot at the end of the working day. And, there are sets $C = \{C_1, ..., C_n\}$ of $n$ customers, each customer has a positive demand of concrete $q(c) \in Q^+$. If the full demand $q(c)$ of customer $c$ is not satisfied, penalty cost $\beta_1 \in R$and $\beta_2 \in R$ are accurate; $\beta_1$ refers to the punishment when customer demand is not fully met; while, $\beta_2$ is the punishment of the quantity that customer also need. The relationship of these two penalty costs is $\beta_1 \gg \beta_2$, which means that first we try to fully meet customer demand and if cannot, we should deliver as much as concrete to the customer. The remaining demands that fail to meet can use other ways to satisfy, such as out sourcing. This is expected to be investigated in our future research.

A vehicle which arrives to customer $c$ requires a service time $S(c) \in D$ for parking and unloading, etc. A delivery vehicle has to begin supplying customer $c$ in the time interval $[a(c), b(c)]$. It means the process of deliveries to customer $c$ need to respect a hard time-window $[a(c), b(c)]$. The customer is also likely to specify a first delivery deadline $b^{'}(c)$, which means the first delivery of the day must start in the time interval $[a(c), b^{'}(c)]$. The customer will specify two time intervals $mintl(c) \in N$ and $maxtl(c) \in N$ in which deliveries have to respect. Say, a subset of vehicle fleet conducts a set of $n_c$ deliveries to customer $c$. So, customer $c$ is supplied at chronological times $w_1, w_2, ..., w_{n_c}$. A minimum time lag $mintl(c)$ and a maximum time lag $maxtl(c)$ enforce the condition those two consecutive deliveries $(i, i+1)$ with $i \in \{1, ..., n-1\}$ are not too far or too close in time, i.e. $mintl(c) \le w_{i+1} - w_i \le maxtl(c)$.

The depot $D$ also presents a service time $S(D) \in N$ for parking the vehicles and filling them, etc. For the limited production rate of the depot, a minimum time lag $mintl(D) \in N$ confirms that two consecutive loading operations at batch plant $D$ are not too close in time. There is a time window $[a(D), b(D)]$ assigned to depot $D$ which restricts the time of reloading. A cost $\alpha(K) \in R$ specifies using a vehicle for delivering concrete during the working day. Owing to the perishable nature of concrete, the maximum time that concrete may reside in the vehicles is represented by parameter $\gamma \in N$.

## 3 Mathematical mode

Using the parameters given in the last section, we define a mixed integer programming model based network flow model. In the model, each possible delivery is conducted to a customer, each possible reload at the depot and the starting and ending points of vehicles as a node are presented. The graph $G = (V, E, t, Z)$ is as follows.

The batch plant $D$ is represented by $D^G = \{D_1, ..., D_{n(D)}\}$. Each $D_i$ is a possible reloading at depot, where $n(D) = \lceil (b(D) - a(D))/mintl(D) \rceil$ is the maximum number of possible reloading of vehicles at the depot during the considered working day. Analogously, this research represented every customer $C_i \in C$ in the model by $C_i^G = \{C_{i,1}, ..., C_{i,n(C_i)}\}$ customer nodes, where $n(C_i) = \lceil q(C_i)/q(K) \rceil$ is obviously the maximum number of necessary deliveries to $C_i$. The node set $V$ of $G$ is defined as $V = C^G \cup D^G \cup S \cup P$, where $C^G = \sum_{i=1}^{n} C_i^G$, $S$ represents the artificial starting node and $P$ is the artificial ending node. The edge set $E$ of $G$ is defined as $(u, v)$ where $u \in V, v \in V$. In graph $G$, $t : (C^G \cup D^G \cup S \cup P) \times (C^G \cup D^G \cup S \cup P)$ is the time used by the vehicle to travel across the edge of $E$, where $\times$ is the symbol represent Descartes. And $Z : (C^G \cup D^G \cup S \cup P) \times (C^G \cup D^G \cup S \cup P)$ is the cost of the edge. Since there are various kinds of edges, among which, some edges should be removed from the process of scheduling and some are feasible for scheduling.

$(u, v) \in S \times C^G$ and $(u, v) \in C^G \times C^G$, as two kinds of edges, represent the vehicles travel from the starting node to the customer node, and from customer node to customer node respectively. It is obvious that concrete must have been loaded in vehicles in advance of the delivery to customers. The nature of concrete decides that vehicle cannot service two customers in the same round. So these two kinds of edges should be removed.

$(u, v) \in D^G \times D^G$ and $(u, v) \in D^G \times P$, as two kinds of edges, denote a vehicle travel from a depot node to another depot node or to the ending node respectively. Because vehicles have to service customers after being loaded, those two kinds of edges should be removed.

The edge $(u, v) \in (C^G \cup D^G \cup P) \times S$ represents that the vehicles have reached the starting node. And the edge $(u, v) \in P \times (C^G \cup D^G \cup S)$ indicates the vehicles have started from the ending node. Apparently, those two kinds of edges should be removed.

The remaining types of edges which fits for scheduling are kept in the model. They are described as follows:

The type of edge $(u, v) \in S \times D^G$ represents the starting time of the vehicles service for customer; it refers to the first loading operation of the day. In the model, all vehicles come back to the batch plant after finishing their all jobs, so the time used in this case is zero $t(u, v) = 0$ and the cost $Z(u, v) = \alpha(K)$ of the edge is the cost of using the vehicle for delivering concrete during the working day.

$(u, v) \in S \times P$ displays the vehicles from starting node to ending node directly. It refers that those vehicles are not used in the working day. And there is no time needed for travel and no cost is attached to this type of edge.

$(u,v) \in D^G \times C^G$ and $(u,v) \in C^G \times D^G$ are two types of edges that represent the vehicle reloading in depot for servicing a customer and have finished unloading at customer node and then return to depot for another reloading. By attaching those two types of edges, time function $t(u,v) = dis(u,v)/Vspeed$ is the time spent for delivering concrete to customer or for the return trip from customer to depot, where $dis(u,v)$ is the distance between node $u$ and $v$, $Vspeed$ is the vehicle speed in travelling, in this paper we assume the $Vspeed$ is a constant. And $Z(u,v) = p \cdot t(u,v) + q \cdot (w_v - w_u - t(u,v))$ is the cost of an edge, where $p$ is the transportation cost per unit time, $w_u$ is the time of the vehicle loading at depot or unloading at a construction side, $w_v$ is the time of the vehicle arrive at the node $v$.

$(u,v) \in C^G \times P$ is the last unloading operation of the working day for vehicles. The time function indicates the travel time that the vehicle goes back to the depot from construction side and the cost function stands for its transportation expense. On the basis of the network flow models above mentioned, a mixed integer programming model for solving RMC vehicle scheduling problem is constructed. The decision variables of the model are listed as follows:

$$x_{u,v} = \begin{cases} 1 & \text{if there is a truck from node } u \text{ to } v \\ 0 & otherwise \end{cases}$$

$$w_u = \begin{cases} T & \text{time of a vehicle reached } u, \forall (u,v) \in E, \\ & x_{uv} = 1 \\ \varnothing & \forall (u,v) \in E, x_{uv} = 0 \end{cases}$$

$$y_c = \begin{cases} 1 & \text{if the total demad } q(c) \text{ of customer } c \\ & \text{is satisfied, } \forall c \in C \\ 0 & \text{otherwise} \end{cases}$$

In view of a simpler notation, the paper defines two sets $\Delta^+(u) : \{v \mid (u,v) \in E\}$ of node $u \in V$ as the subsequent nodes of $u$ regarding graph $G$, and $\Delta^-(u)\{v \mid (u,v) \in E\}$ of node $u \in V$ as the precursor node of $u$.

$$Min \sum_{(u,v)\in E} x_{uv} \cdot Z(u,v) + \sum_{c \in C} [q'(c) \cdot \beta_2 + (1 - y_c) \cdot \beta_1] \quad (1)$$

$$\sum_{v \in \Delta^+(S)} x_{Sv} = K \quad (2)$$

$$\sum_{u \in \Delta^-(E)} x_{uE} = K \quad (3)$$

$$\sum_{u \in \Delta^-(v)} x_{uv} - \sum_{u \in \Delta^+(v)} x_{vu} = 0 \quad \forall v \in C^G \cup D^G \quad (4)$$

$$\sum_{v \in \Delta^+(u)} x_{uv} \le 1 \quad \forall u \in C^G \quad (5)$$

$$\sum_{v \in \Delta^+(u)} x_{uv} \le 1 \quad \forall u \in D^G \quad (6)$$

$$\sum_{v \in \Delta^+(C_{i,j+1})} x_{C_{i,j+1},v} - \sum_{v \in \Delta^+(C_{i,j})} x_{C_{i,j},v} \le 0 \quad \forall i \in \{1,...,n\}, \forall j \in \{1,...,n(C_i)-1\} \quad (7)$$

$$q(C_i) - \sum_{u \in C_i^G} \sum_{v \in \Delta^+(u)} x_{uv} \cdot q(k) = q'(C_i) \quad \forall i \in \{1,...,n\} \quad (8)$$

$$\sum_{u \in C_i^G} \sum_{v \in \Delta^+(u)} x_{uv} \cdot q(k) \ge q(C_i) \cdot y_{C_i} \quad \forall i \in \{1,...,n\} \quad (9)$$

$$-M \cdot (1 - x_{uv}) + S(u) + t(u,v) \le w_v - w_u \quad \forall (u,v) \in E \quad (10)$$

$$M \cdot (1 - x_{uv}) + \gamma + S(u) \ge w_v - w_u \quad \forall (u,v) \in E \quad with\ u \in D^G, v \in C^G \quad (11)$$

$$w_u \ge a(u) \quad \forall u \in C^G \cup D^G \quad (12)$$

$$w_u \le b(u) \quad \forall u \in C^G \cup D^G \quad (13)$$

$$w_{C_{i,1}} \le b'(u) \quad \forall i \in \{1,...,n\} \quad (14)$$

$$w_{C_{i,j+1}} - w_{C_{i,j}} \ge mintl(C_i) \quad \forall i \in \{1,...,n\} \forall j \in \{1,...,n(C_i)-1\} \quad (15)$$

$$w_{C_{i,j+1}} - w_{C_{i,j}} \le maxtl(C_i) \quad \forall i \in \{1,...,n\} \forall j \in \{1,...,n(C_i)-1\} \quad (16)$$

$$w_{D_{i+1}} - w_{D_i} \ge mintl(D) \quad \forall j \in \{1,...,n(D)-1\} \quad (17)$$

$$x_{uv} \in \{0,1\} \quad \forall (u,v) \in E \quad (18)$$

$$w_u \in T \quad \forall u \in C^G \cup D^G \quad (19)$$

$$y_c \in \{0,1\} \quad \forall c \in C \quad (20)$$

The objective function (1) minimizes the total sum of costs of edges applied by arbitrary vehicle. So that, concrete delivery to customers as more frequent as possible by penalty costs $\beta_1$ and $\beta_2$ is ensured. Constraints (2) and (3) demonstrate that the number of vehicles which depart from the starting node and reach the ending node must be equal to $K$. Constraints (4) are flow conservation ones for all nodes except the technical starting node and ending node; Constraints (5) are used to ensure that each customer node is employed at the most once; Constraints (6) states that the depot can load only one vehicle at a time; Constraints (7) confirms that a customer's demand must be satisfied sequentially. It is a fact that a vehicle supplies a customer node $C_{ij}$, however no vehicle supplies a customer node $C_{ij'}$ with $j' > j$; Equation (8) is used to calculate the rest of the demands of a unsatisfied customer; In constraints (9) , the $y_c$ variables indicates whether the demand of customer $c$ is satisfied or not; Constraints (10) make sure that a vehicle form node $u$ to node $v$ comply with the time constraints by using a big constant $M$. In the constraints $x$ variables and $w$ variables are connected by enforcing the travel times and service times. Constraints (11) is used to ensure that no concretes are kept in a concrete mixer longer than $\gamma$. Constraints (12) and (13) ensure that the time windows are respected. Constrains (14) enforce that the first delivery to a customer regards the first delivery deadline $b^{'}(c)$. Two consecutive deliveries for a customer must be abide by the time lag constraints, this is prevented with constrains (15) and (16). Constraints (17) and (6) are used to ensure the restriction of productivity of batch plant. Finally, constraints (18)-(20) define the domains of the decision variables.

## 4 The hybrid heuristic algorithm

The hybrid heuristic algorithm proposed in this study was based on the concept of transforming the complex scheduling of RMC vehicles in to a processing of combinations of simple problems of several limited operations each time. In the scheduling process of the RMC vehicles, it was necessary to arrange available times and vehicles in the depot for each customer to maximize the number of operations while minimizing the operating cost of each vehicle. The available time at the depot was limited by the loading rate thereof, that is, by the minimum time interval $mintl(D)$ of two continuous loadings in the depot. It was assumed that the loading times of $n$ continuous operations in the depot were:

$w_{u1},...,w_{ui},...,w_{un}$respectively. Therefore, given $i<j$, $w_{ui},w_{uj}$need to satisfy$w_{uj}-w_{ui}\geq mintl(D)$. It was time-consuming to provide available depot time for all customers at one time and at minimum cost. Moreover, such conduct was unsuitable for the rapid scheduling of RMC vehicles. Therefore, the present study proposed a hybrid heuristic algorithm to achieve rapid scheduling of RMC vehicles.

Firstly, according to the loading rates at the depot, each time of possible loading at the depot was fixed. That is, according to the work-starting time $a(D)$ and work-ending time $b(D)$ at the depot, all loading time points became fixed as:

$a(D),a(D)+mintl(D),..,a(D)+\quad m\cdot mintl(D)$ respectively, where $m=\lceil (b(D)-a(D))/mintl(D)\rceil$.

Subsequently, the hybrid heuristic algorithm proposed was used to process one operation for each customer at each time to yield the scheduling results, namely, in the form of a pair composed of the loading-starting time and unloading-starting time of each operation (noted as $<w_{ui},w_{vi}>$). For example, there were $n$ customers. It was necessary to process the $i_{th}$ operations of customers $C_1,...,C_n$, $i\leq max(n(C_1),..,n(C_n))$. In the case of the operation number $n(C_j)$ as demanded by the customer being smaller than $i$, the operations for customer $C_j$ had been completely scheduled.

### 4.1 PRETREAMENT

For each execution of the algorithm, it was firstly necessary to process an operation for each customer to get the specified loading-starting time range $(Et_1,Et_2)$ of the operation. Please see Figure 1, as for one operation of customer $C_i$, $St_1(C_i)$ and $St_2(C_i)$ represent the earliest and latest loading-starting time respectively, without considering the wait time at construction side. $Et_1(C_i)$ and $Et_2(C_i)$ refer to the earliest and latest loading-starting times of this operation, the relationship between $Et$ and $St$ are shown by Equation(22) and (23). $REt(C_i)$ was the real unloading-starting time of the optimized operation. $RSt(C_i)$ was the loading-starting time of the operation when neglecting the waiting time in the depot. See Equation(21).

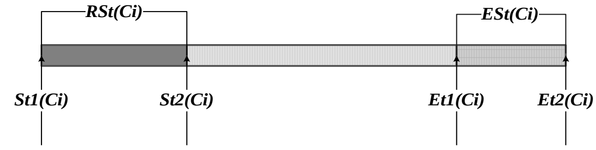


FIGURE 1 The duration of an operation

$$REt(C_i)=RSt(C_i)+dis(C_i)/Vspeed+S(C_i) \quad (21)$$

$$Et_1(C_i)=St_1(C_i)+dis(C_i)/Vspeed+S(C_i) \quad (22)$$

$$Et_2(C_i)=St_2(C_i)+dis(C_i)/Vspeed+S(C_i) \quad (23)$$

As for customer $C_i$, if operation $j$ as selected was the first operation for customer$C_i$, $Et_1(C_i)=a(C_i)$ and $Et_2(C_i)=b^{'}(C_i)$. Otherwise,

$Et_1(C_i)=w_{v,(j-1)}+S(C_i)+mintl(C_i)$,

$Et_2(C_i)=w_{v,(j-1)}+S(C_i)+maxtl(C_i)$,

where, $w_{v,(j-1)}$ is the unloading-starting time of the $(j-1)_{th}$ operation of customer $C_i$.

### 4.2 DESCRIPTIONS OF THE ALGORITHM

The hybrid algorithm proposed in this study was based on the concept that the smaller the cross-section of the time range of the operations of $n$ customers, the fewer depots and vehicles were occupied, and the more operations could thus be completed. Meanwhile, the operating cost of each vehicle should be minimized in the execution process of the algorithm, Therefore, given limited vehicle numbers, it was able to complete the customer's operations as far as possible and minimize vehicle operating cost. The algorithm proceeded as follows:

| **Hybrid heuristic algorithm** | |
|---|---|
| Step1 | Fix the time for depot and setting $j=1$. |
| Step2 | Select the $j_{th}$ operation of customers $C_1,...,C_n$. For an arbitrary customer $C_i$, $j>n(C_i)$ represents the fact that the operations of customer $C_i$ have been completely arranged and no further treatment of the operations of customer $C_i$ was necessary. |
| Step3 | Pre-process the $m(m\leq n)$ times of operations selected in Step 2 to get the starting time range $[Et_1,Et_2]$of each operation. |
| Step4 | Run the minimum sub-algorithm in the cross-section to obtain $REt(C_i)$ and $RSt(C_i)$of the $m$ |

| | |
|---|---|
| | operations selected in step 2. |
| Step5 | Execute the selection process of depot nodes to obtain the depot node $w_u$ serving the $m$ times of operation respectively. The $m$ time pairs $< w_u, w_v >$ were inserted in the scheduling results. For an arbitrary customer $C_i$, $w_v = REt(C_i)$. |
| Step6 | Set $j = j + 1$ and repeating Step2 to Step5 until all customer operations were arranged. |
| Step7 | According to the solutions obtained, execute the scheduling process for the vehicles and calculate the objective value. |
| Step8 | Return to the scheduling results sequence, vehicle flow data, and objective value obtained. |

## 4.3 MINIMUM SUB-ALGORITHM IN THE CROSS-SECTION

In Step 4 of the hybrid heuristic algorithm, the minimum sub-algorithm in the cross-section was purposed to specify the unloading-starting time $REt(C_i)$ of the $m$ times of operations selected and ensure the minimum cross-section of the operation-completion time of the $m$ customers. The mathematical model for minimizing the cross-section is given in Equation (13)-(17), with a decision variant of $REt(C_i)$.

$$Min \sum_{\forall i \in n} \sum_{l \neq i} min(EAt(C_i), EAt(C_l)) - max(RSt(C_i), RSt(C_l)) \tag{24}$$

$$REt(C_i) \leq Et_2(C_i) \tag{25}$$

$$REt(C_i) \geq Et_1(C_i) \tag{26}$$

$$REt(C_i) = RSt(C_i) + dis(C_i)/Vspeed + S(C_i) \tag{27}$$

$$EAt(C_i) = REt(C_i) + S(C_i) + (n(C_i) - j) \cdot (S(C_i) + (mintl(C_i) + maxtl(C_i))/2) \tag{28}$$

Equation(24) is the target of the model. It represents that, since the $j_{th}$ operation, the cross-sections of the average operation-completion time range of different customers were minimized. Equation (25) and (26) denote the valuation of the range of decision variant $REt(C_i)$. Equation (27) signifies the relationship between $REt(C_i)$ and $RSt(C_i)$; Equation (28) refers to the average time taken to complete all operations when the unloading-starting time of the $j_{th}$ operation was $REt(C_i)$. Where, $n(C_i) - j$ was the number of operations for customer $C_i$ that had not been arranged; $S(C_i) + (mintl(C_i) + maxtl(C_i))/2$ was the average time interval between unloading events in any two continuous operations. According to Equation (28), the optimization involved the completion time range of each operation and the customer demand. It was thus able to schedule the concrete delivery vehicles more effectively so as to complete customer operations to the greatest extent. The model above was solved using a genetic algorithm.

## 4.4 SELECTING DEPOT NODE

After obtaining $REt(C_i)$ for the $m$ operations, the available depot nodes for these $m$ operations should be determined according to $REt(C_i)$ as follows: Step 1, determine the allowable starting time ranges of the operations; Step 2, select an available time for each depot node within the time range as the loading-starting time for each operation. The process is listed in detail as follows.

| **The process of selecting depot nodes** | |
|---|---|
| Step1 | Set $j = 1$. |
| Step2 | Calculate the allowable starting time range $[Stime, Etime]$ of an operation of customer $C_j$. $Stime = REt(C_i) - \gamma$, where $\gamma$ is the maximum time for which the concrete remained in vehicle. $Etime = REt(C_j) - dis(C_j)/Vspeed - S(D) = RSt(C_j)$ |
| Step3 | Select an unused depot node $w_u$ that is closest to $RSt(C_j)$ in the time range $[Stime, Etime]$. Since $RSt(C_j)$ is the latest starting time that excludes the waiting time in the depot. Therefore, such a selection was targeted to minimize the waiting time of vehicles at the depot. |
| Step4 | Insert the time pair obtained $< w_u, w_v >$ into the scheduling result sequence, where, $w_v = REt(C_j)$ . |
| Step5 | Set $j = j + 1$ and repeat Step2 to 4 until $j > m$. |

## 4.5 THE VEHICLE SCHEDULING PROCESS

After obtaining the scheduling result sequence for operations for all customers, it was necessary to arrange vehicles for these operations: this process is shown as follows:

| **The vehicle scheduling process** | |
|---|---|
| Step1 | Arrange all customer operations in scheduling result sequence according to $w_u$ (from big to small) |
| Step2 | According to the operation sequence in Step 1, search for the first operation $< w_{ui}, w_{vi} >$ that has not been inserted in the vehicle flow from the first vehicle $k$, $< w_{ui}, w_{vi} >$ is regarded as the first operation needed to be completed by the vehicle and is inserted in the vehicle flow of vehicle $k$. In addition, we note that $< w_{u(k,s)}, w_{v(k,s)} >=< w_{ui}, w_{vi} >$, where $s$ refers to the current operation of the vehicle. In case of the absence of such an operation, all customer operations have been completed. |
| Step3 | According to the operation sequence in Step 1, search $< w_{u(i+1)}, w_{v,(i+1)} >$ backwards from position $< w_{ui}, w_{vi} >$ of the first operation of vehicle $k$. If the $(i+1)_{th}$ time operation has been inserted in other vehicle flows, search the $(i+2)_{th}$ operation backwards until a certain operation $j$ has not been inserted into other vehicle flow. Subsequently, verify whether $< w_{v(k,s)}, w_{uj} >$ satisfies the restriction of transportation or not. If $< w_{v(k,s)}, w_{uj} >$ does not satisfy the constraint, search the $(j+1)_{th}$ operation backwards continuously; if $< w_{v(k,s)}, w_{uj} >$ satisfies the constraint, vehicle $k$ can return to the depot in time for the next load and serve for $j$ times after the first operation was completed. In this case, set $w_{u(k,s+1)}, w_{v,(k,s+1)} =< w_{uj}, w_{vj} >$and |

| | |
|---|---|
| | $s = s + 1$ to search for the next operation of this vehicle until all operations were completed cyclically. |
| Step4 | Set $k = k + 1$ and repeating step2 and 3 until the vehicle flow of all vehicles are assigned an operation. |
| Step5 | Search all the vehicle flows that have a completed assignment. In case of the presence of the $j_{th}$ operation, and in the absence of the $(j-1)_{th}$ operation of customer $C_i$ in the vehicle flow, that is, the $j_{th}$ instead of the $(j-1)_{th}$ operation has been completed, the $j_{th}$ operation of customer $C_i$ is deleted since the operations of any arbitrary customer should be completed in sequence. |
| Step6 | According to vehicle flow information, determine how many customer demands are not completely satisfied and how many customer operations remain incomplete. Correspondingly, calculate the penalty on the second part of the objective value of unfinished customer operations. The vehicle flow of arbitrary vehicle $< w_{u(k,1)}, w_{v(k,1)} >$, $< w_{u(k,2)}, w_{v(k,2)} >, \ldots, < w_{u(k,e)}, w_{v(k,e)} >$ is represented by time pair sequence. According to the vehicle flow, calculate the first part of this vehicle flow. By summing the two parts, the objective value of each scheduling result was obtained. |

## 5 Numerical experiment and analysis

To verify the algorithm proposed, real world data obtained from a typical working week from a RMC distributing firm are applied to perform test.

| Instance1 | *K=20, mintl(D)=4min, q(k)=6$m^3$, S(D)=7min, Vspeed=40km/h* | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **ID** | **Demand** | **Distance** | **a(c)** | **b(c)** | **b'(c)** | **mintl(c)** | **maxtl(c)** | **S(c)** |
| 1 | 90$m^3$ | 22km | 8:00 | 16:00 | 8:20 | 5min | 10min | 9min |
| 2 | 30$m^3$ | 13km | 7:00 | 16:00 | 7:30 | 3min | 9min | 5min |
| 3 | 38$m^3$ | 9km | 7:30 | 15:00 | 7:50 | 6min | 13min | 12min |
| 4 | 72$m^3$ | 31km | 8:30 | 17:00 | 9:00 | 3min | 10min | 7min |
| 5 | 66$m^3$ | 23km | 8:50 | 17:00 | 9:10 | 4min | 8min | 10min |
| **Instance2** | | | | | | | | |
| **ID** | **Demand** | **Distance** | **a(c)** | **b(c)** | **b'(c)** | **mintl(c)** | **maxtl(c)** | **S(c)** |
| 1 | 66$m^3$ | 32km | 9:00 | 17:00 | 9:20 | 4min | 9min | 8min |
| 2 | 36$m^3$ | 8km | 8:30 | 16:00 | 8:50 | 6min | 12min | 10min |
| 3 | 43$m^3$ | 9km | 7:30 | 15:00 | 7:50 | 5min | 10min | 16min |
| 4 | 52$m^3$ | 16km | 8:00 | 17:00 | 8:20 | 3min | 8min | 9min |
| 5 | 36$m^3$ | 15km | 9:20 | 17:00 | 9:40 | 4min | 10min | 7mini |
| 6 | 26$m^3$ | 13km | 9:30 | 16:00 | 9:50 | 5min | 9min | 9min |
| **Instance3** | | | | | | | | |
| **ID** | **Demand** | **Distance** | **a(c)** | **b(c)** | **b'(c)** | **mintl(c)** | **maxtl(c)** | **S(c)** |
| 1 | 102$m^3$ | 13km | 7:30 | 17:00 | 7:50 | 3min | 7min | 7min |
| 2 | 16$m^3$ | 26km | 8:30 | 16:00 | 8:50 | 0min | 9min | 8min |
| 3 | 32$m^3$ | 11km | 9:30 | 15:00 | 9:50 | 6min | 13min | 9min |
| 4 | 96$m^3$ | 23km | 7:50 | 17:00 | 8:20 | 3min | 9min | 16min |
| 5 | 22$m^3$ | 9km | 8:20 | 17:00 | 8:50 | 4min | 11min | 10min |
| **Instance4** | | | | | | | | |
| **ID** | **Demand** | **Distance** | **a(c)** | **b(c)** | **b'(c)** | **mintl(c)** | **maxtl(c)** | **S(c)** |
| 1 | 32$m^3$ | 16km | 10:00 | 17:00 | 10:20 | 0min | 5min | 13min |
| 2 | 38$m^3$ | 18km | 9:30 | 16:00 | 9:50 | 6min | 10min | 11min |
| 3 | 66$m^3$ | 22km | 7:50 | 15:00 | 8:10 | 4min | 9min | 15min |
| 4 | 23$m^3$ | 9km | 8:20 | 17:00 | 8:40 | 0min | 6min | 8min |
| 5 | 17$m^3$ | 26km | 8:30 | 17:00 | 8:50 | 5min | 12min | 9min |
| 6 | 29$m^3$ | 15km | 7:30 | 16:00 | 7:50 | 3min | 10min | 13min |
| 7 | 46$m^3$ | 11km | 9:00 | 18:00 | 9:20 | 3min | 8min | 9min |
| **Instance5** | | | | | | | | |
| **ID** | **Demand** | **Distance** | **a(c)** | **b(c)** | **b'(c)** | **mintl(c)** | **maxtl(c)** | **S(c)** |
| 1 | 88$m^3$ | 18km | 7:30 | 16:00 | 8:00 | 2min | 10min | 9min |
| 2 | 21$m^3$ | 23km | 9:30 | 12:00 | 9:50 | 0min | 5min | 8min |
| 3 | 36$m^3$ | 19km | 7:30 | 15:00 | 7:50 | 3min | 11min | 16min |
| 4 | 54$m^3$ | 26km | 9:00 | 17:00 | 9:20 | 4min | 8min | 10min |
| 5 | 9$m^3$ | 8km | 8:30 | 17:00 | 9:00 | 2min | 9min | 13min |
| 6 | 69$m^3$ | 17km | 10:10 | 16:00 | 10:30 | 0min | 13min | 18min |

FIGURE 2 Demand data

The real world data and constraints are modelled by the aforementioned MIP model. The test data contain five days customer demand data, as instance 1 to instance 5 . Each customer demand data contains customer ID, demand for RMC, distance between depot and construction side, etc. There are twenty vehicles in the depot and the loading capacity of each vehicle is $6m^3$. Please see Figure 2.

### 5.1 RESULT

The algorithm is validated by using the five instances above mentioned. Please see figure 3.

The number of customers is abbreviated NC; total demand of customers is TD; total number of jobs that should be delivered is TJ; cancelled jobs is CJ; used vehicles is UV; the total delivery and return time of vehicles is TDT ; the average waiting time per delivery is AWT; The objective value is OBJ; And the execution time of the algorithm is RT;

| Instance | NC | TD | TJ | CJ | UV | TDT | AWT | OBJ | RT |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 296 | 50 | 0 | 19 | 3249min | 8.5min | 9543.5 | 1.6319sec |
| 2 | 6 | 259 | 45 | 0 | 19 | 2313min | 16.5min | 9579.5 | 1.3387sec |
| 3 | 5 | 268 | 46 | 0 | 16 | 2307min | 3min | 8008 | 1.8406sec |
| 4 | 7 | 251 | 43 | 0 | 15 | 2223min | 13.5min | 7840.5 | 1.3776sec |
| 5 | 6 | 277 | 48 | 0 | 16 | 2790min | 10.5min | 8063 | 1.7890sec |

FIGURE 3 The result

### 5.2 ANALYSIS OF THE RESULT

In order to verify the validity of the proposed algorithm (we denoted it as HGA algorithm). We use two-phase algorithm presented in literature [10] (here we denoted it as 2GA algorithm) and a heuristic algorithm based on neighborhood search method presented in literature [12] (here we denoted as HN algorithm) to optimize the aforementioned instances. Please see figure 4.

| | Instance1 | | | | Instance2 | | | | Instance3 | | | | Instance4 | | | | Instance5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HN | 2GA | HGA | Compare | HN | 2GA | HGA | Compare | HN | 2GA | HGA | Compare | HN | 2GA | HGA | Compare | HN | 2GA | HGA | Compare |
| RT | 368.34sec | 471.13sec | 1.6319sec | HN 0.44% 2GA 0.35% | 642.18sec | 441.41sec | 1.3387sec | HN 0.21% 2GA 0.30% | 377.24sec | 380.35sec | 1.8406sec | HN 0.49% 2GA 0.48% | 892.39sec | 428.38sec | 1.3776sec | HN 0.15% 2GA 0.32% | 681.71sec | 402.07sec | 1.7890sec | HN 0.26% 2GA 0.44% |
| UV | 19 | 19.2 | 18.7 | | 18 | 19.6 | 19.2 | | 16 | 16.8 | 16 | | 16 | 16.3 | 15.2 | | 17 | 16.1 | 16 | |
| CJ | 0 | 0 | 0 | | 1 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | | 0 | 0 | 0 | |
| TDT | 3249min | 3249min | 3249min | | 2286min | 2313min | 2313min | | 2307min | 2307min | 2307min | | 2223min | 2223min | 2223min | | 2790min | 2790min | 2790min | |
| AWT | 83.5min | 136.25min | 14.05min | | 132min | 129.35min | 26.5min | | 88.5min | 67.65min | 2.95min | | 30.5min | 103.25min | 113.55min | | 9min | 125.25min | 20.85min | |
| OBJ | 12999.5 | 14771.35 | 12792.15 | HN -207.352 GA -1979.2 | 15682 | 15421.25 | 11892.5 | HN -3789.5 2GA -3528.75 | 10572.5 | 10979.35 | 10313.25 | HN -259.25 2GA -666.1 | 10498.5 | 10600.5 | 10063.5 | HN -435 2GA -537 | 11317 | 11215.75 | 10849.35 | HN -467.65 2GA -366.4 |

FIGURE 4 The result and compare by HGA, HN, and 2GA

From the result we can obtain the result of HGA algorithm is better than the result of 2GA algorithm and HN algorithm. The average execution time of HN algorithm and 2GA algorithm is 592.372sec and 424.668sec. The average execution time of the proposed HGA algorithm is 1.5956sec. It is obvious that the execution time of HGA is significantly less than the time of HN and 2GA. Then we use aforementioned instance1 as a base instance to optimize by HGA, HN, and 2GA algorithm with adding operations to the customers of instance1. Every execution of these 3 algorithms we add an operation to each customer of instance1, then compare the result of these 3 algorithms. Please see figure 5 and figure 6. In figure 5 the objective values are shown, and the execution times are shown in figure 6.

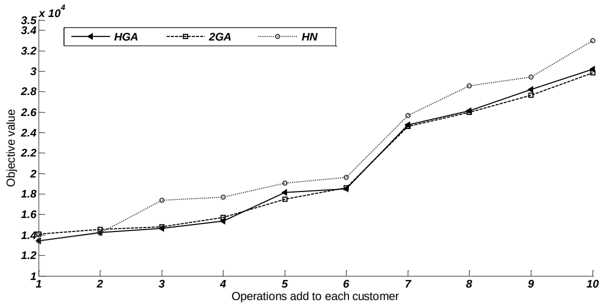


FIGURE 5 Objective value of HGA,HN and 2GA in add operations to customer

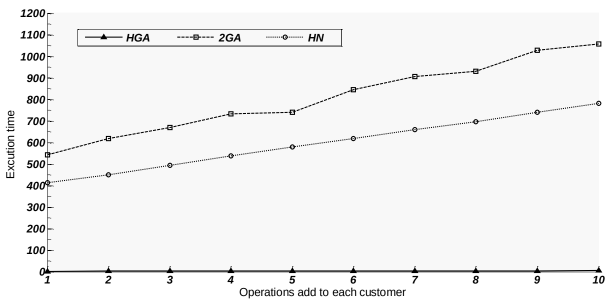


FIGURE 6 Execution time of HGA,HN and 2GA in add operations to customer

It is can be seen in figure 5 and figure 6 that the objecttive value of HGA algorithm is a little better than HN and 2GA in the number of operations under the condition of less, in the number of operations increased, the effect of HN algorithm is reduced obviously, the result of HGA algorithm and the 2GA algorithm were basically identical. But the execution time of HGA is much better than 2GA and HN algorithm, that is to say, the optimized scheduling scheme can be obtained quickly by the HGA algorithm.

## 6 Conclusions

The main purpose of this research is to schedule the vehicle fleet of depot to service customers under static condition. First we analyze the characteristics of ready mixed concrete delivery problem and the important of a rapid effectiveness algorithm for RMC companies. Then we developed a MIP model for delivery RMC, and optimize instances of the problem by a heuristic approach based on GA algorithm. Through the analysis of the result we can see the algorithm is effective and fast for RMC vehicle scheduling problem.

Because of the nature feature of concrete and the characteristics of ready mixed concrete scheduling problem, timeliness is more important for delivery concrete, the study of a rapid algorithm for RMC scheduling problem is necessary. So far we have only considered the problem from the static viewpoint, that all input data is available at the time of computing the solution. We assume that there are no changes of orders, time windows, etc. during the working day. This is actually not realistic, as on a working day from minute to minute the plan needs, required time of customer, the state of depot, etc. In the future work we should study rescheduling strategy of RMC vehicle scheduling problem to deal with the dynamic environment and use the rapid algorithm we proposed in the rescheduling operation. Moreover the Randomness and fuzziness of RMC scheduling problem should be studied.

## References

[1] Durbin M, Hoffman K (2008) OR Practice-The Dance of the Thirty-Ton Trucks: Dispatching and Scheduling in a Dynamic Environment. *Operations research*, 56(1), 3-19.
[2] Tommelein I. D. & Li, A (1999) Just-in-time concrete delivery: mapping alternatives for vertical supply chain integration, *Proceedings IGLC*, 97
[3] Matsatsinis NF (2004) Towards a decision support system for the ready concrete distribution system: A case of a Greek company. *European Journal of Operational Research*, 152(2), 487-499.
[4] Feng C-W, Cheng T-M, Wu H-T (2004) Optimizing the schedule of dispatching RMC trucks through genetic algorithms. *Automation in Construction*, 13(3), 327-340.
[5] Naso D, Surico M, Turchiano B, Kaymak U (2007) Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. *European Journal of Operational Research*, 177(3), 2069-2099.
[6] Yan S, Lai W (2007) An optimal scheduling model for ready mixed concrete supply with overtime considerations. *Automation in Construction*, 16(6), 734-744.
[7] Yan S, Lai W, Chen M (2008) Production scheduling and truck dispatching of ready mixed concrete. *Transportation Research Part E: Logistics and Transportation Review*, 44(1), 164-179.
[8] Yan S, Lin H-C, Liu Y-C (2011) Optimal schedule adjustments for supplying ready mixed concrete following incidents. *Automation in Construction* , 20(8), 1041-1050.
[9] Asbach L, Dorndorf U, Pesch E (2009) Analysis, modeling and solution of the concrete delivery problem. *European Journal of Operational Research*, 193(3), 820-835.
[10] Feng C-W, Wu H-T (2006) Integrating fmGA and CYCLONE to optimize the schedule of dispatching RMC trucks. *Automation in Construction*, 15(2), 186-199.
[11] Graham D, Smith SD, Crapper M (2004) Improving concrete placement simulation with a case-based reasoning input. *Civil Engineering and Environmental Systems*, 21(2), 137-150.
[12] Darren Graham L, Smith SD, Dunlop P (2005) Lognormal distribution provides an optimum representation of the concrete delivery and placement process. *Journal of construction engineering and management*, 131(2), 230-238.
[13] Graham LD, Forbes DR, Smith SD (2006) Modeling the ready mixed concrete delivery system with neural networks. *Automation in Construction*, 15(5), 656-663.
[14] Lin P-C, Wang J, Huang S-H, Wang Y-T (2010) Dispatching ready mixed concrete trucks under demand postponement and weight limit regulation. *Automation in Construction*, 19(6), 798-807.

[15] Schmid V, Doerner KF, Hartl RF et al. (2009) A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1), 70-85.
[16] Schmid V, Doerner KF, Hartl RF, Salazar-González J-J (2010) Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers and Operations Research*, 37(3), 559-574.
[17] Min W, Sui Pheng L (2007) Modeling just-in-time purchasing in the ready mixed concrete industry. *International Journal of Production Economics*,107(1), 190-201.

## Authors

**Guochen Zhang, 8. 9. 1980, China**

**Guochen Zhang received his MSc in Computer Application from Taiyuan University of Science and Technology in 2006. He is currently a Lecturer at the Department of Computer Science and Technology, TUST, Taiyuan, China. He is currently pursuing his PhD in Control Theory and Control Engineering at Lanzhou University of Technology. His current research interests include production scheduling and intelligent optimization.**

**Jianchao Zeng, 8. 10. 1963, China**

**Jianchao Zeng is a Professor and Tutor of PhD student at the Department of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan, P.R. China. He is now the Vice President of TUST. He received his MSc and PhD in System Engineering from Xi'an Jiaotong University in 1985 and 1990, respectively. His current research focuses on modeling and control of complex systems, intelligent computation, swarm intelligence and swarm robotics. He has published more than 200 international journal and conference papers. Since the 1990s, he has been an Invited Reviewer of several famous scientific journals.**