Solving Ready Mixed Concrete Delivery Problems: Evolutionary Comparison between Column Generation and Robust Genetic Algorithm

Mojtaba Maghrebi¹, Vivek Periaraj², S. Travis Waller³ and Claude Sammut⁴

¹School of Civil and Environmental Engineering, The University of New South Wales, Sydney, NSW, 2052, Australia; maghrebi@unsw.edu.au
²Department of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ, USA
³School of Civil and Environmental Engineering, The University of New South Wales, Sydney, NSW, 2052, Australia and NICTA
⁴School of Computer Science Engineering, The University of New South Wales, Sydney, NSW, 2052, Australia

ABSTRACT

An effective resource allocation technique is required for each Ready Mixed Concrete (RMC). Finding the optimum solution for large scale RMC dispatching problems with available computing facilities is intractable. Two kinds of techniques have been implemented to deal with this problem: (i) evolutionary techniques and (ii) numerical techniques. For the purposes of this paper we selected a technique from each category and compared them under the same conditions. Robust Genetic Algorithm (Robust-GA) and Column Generation (CG) were selected and tested with different sizes of real RMC problems. The results show that on average CG solutions are obtained with a 20% reduced cost. However, Robust-GA converges 40% faster than CG, while the number of unassigned customers for both the techniques is almost the same.

INTRODUCTION

Resource allocation in Ready Mixed Concrete (RMC) is a very crucial task and can have a significant impact on the operation costs and subsequently on the profit. Despite many developments in this area, this task still suffers from a lack of practical solutions (Feng, Cheng, & Wu 2004; Maghrebi, Sammut, & Waller 2013; Naso, Surico, Turchiano, & Kaymak 2007) and in practice dispatching tasks are mainly handled by experts. This encourages researchers in this area to develop more applicable tools and techniques that have a strong application within the industry. The best possible decision for each allocation can be achieved by optimization. However, this optimization model is NP-hard (Maghrebi, Waller, & Sammut 2014) and can only be used optimally in small cases. To overcome this problem heuristic methods are recommended. However, although heuristic methods cannot guarantee the optimum solution, they are able to provide a near optimum solution. Calculating the accuracy of heuristic methods is not feasible because the optimum solution is not
obtainable. Therefore, only comparisons between the implemented heuristic methods can successfully reveal the superior methods. In this paper we compare two implemented methods: Robust Genetic Algorithm (Robust-GA) and Column Generation (CG). This paper consists of four sections. First, the relevant literature in this area is reviewed and the reason for selecting Robust-GA and CG is justified. Second, an RMC mathematical model is explained and two selected methods are discussed briefly. Third, two selected techniques are tested with a field data. Finally the results are deliberated and summarized.

RELATED WORKS

From a general point of view the literature that deals with RMC dispatching can be categorized as follows: (i) the papers that have been devoted to mathematically modelling RMC dispatching problems, (ii) the papers that have tried to find a better solution. In this paper we focus on the second category. However, from the first category we can cite (Asbach, Dorndorf, & Pesch 2009; Durbin & Hoffman 2008; Feng et al. 2004; Naso et al. 2007; Schmid, Doerner, Hartl, & Salazar-González 2010; Shangyao Yan, Lai, & Chen 2008). The second category also can be split into two sub-categories: (ii-I) evolutionary methods, (ii-II) numerically based methods. In this paper a recent method from each sub-category is selected and compared.

Evolutionary methods have been widely used in the literature. Garcia, Lozano, Smith, Kwok, and Villa (2002) implemented a GA based method for solving a single depot RMC problem. Similarly, Feng et al. (2004) modelled a single depot but used larger instances for validating their model. The instances that have been considered by them are much smaller than the instances that are used in this paper. Naso et al. (2007) introduced a GA algorithm which is very similar to the methods that were presented earlier, but their model can deal with multi-depot RMC problems. Lu (2002) presented the concept of integrating Discrete Event Simulation (DES) with evolutionary methods and this idea has been tested with different methods such as GA (Cao, Lu, & Zhang 2004; Lu & Lam 2005) and Particle Swarm Optimization (PSO) (Lu, Wu, & Zhang 2006; Wu, Lu, & Zhang 2005). Silva et al. (2005) compared GA with Ant Colony Optimization (ACO) and suggested a GA-ACO method for solving RMC problems and recently Srichandum and Rujirayanyong (2010) compared Bee Colony Optimization (BCO) and Tabu Search (TS) with GA in this context. Despite developments in this area, the solution structure among most introduced methods is pretty much same, especially in the GA based method where the chromosome structure consists of two merged parts: the first part defines the sources of deliveries; the second part expresses the priorities of customers. The solution structure in these techniques is quite simple and easy to understand. However, a cumbersome computing process must be completed in each iteration to check the constraints or after achieving a premature solution. Then, via supplementary algorithms, any infeasibilities in the achieved solution can be adjusted, mostly by out-sources or idle resources. To overcome this issue Maghrebi, Waller, and Sammut (2013) presented an evolutionary based method which can solve the RMC dispatching problem without needing any supplementary algorithm. This
technique is selected among the evolutionary methods and is compared with a numerically based method which is selected in the following paragraph.

Rather than only looking at evolutionary methods some other numerical approaches have also been studied. Shangyao Yan et al. (2008) introduced a numerical method for solving the RMC optimization problem by cutting the solution space and incorporating the branch and bound technique and the linear programming method. S Yan, Lin, and Jiang (2012) used decomposition and relaxation techniques coupled with a mathematical solver to solve the problem. Asbach et al. (2009) made the mathematical modelling much simpler by dividing the depots and customers into sub-depots and sub-customers and then used large scale instances for testing their introduced large neighborhood search and decomposition methods. More recently, Maghrebi, Periaraj, Waller, and Sammut (2013) implemented a Column Generation (CG) method which is amenable to Dantzig-Wolfe reformulation for solving large scale models which available computing facilities cannot optimally solve in polynomial time.

The Robust-GA method was developed in the School of Civil and Environmental Engineering and the School of Computer Science Engineering of the University of New South Wales (UNSW), while the CG method was developed in collaboration with the University of Arizona. The design aim of the Robust-GA method is to attain a high quality solution within a short period of time and strategy of the CG method by using optimization techniques in a way that can solve the model in polynomial time. These are two totally different approaches for dealing with RMC dispatching problems and are compared in terms of quality and time. Since both techniques were recently developed they have not yet been compared with same field data. Therefore, in this paper both of these techniques are assessed with the same metrics and using the exact same instances. Moreover, up until now an evolutionary technique has not been compared with a near optimum solution in this context.

**SYSTEM FORMULATION**

In this section, RMC dispatching is modelled mathematically and then both the selected Robust-GA and CG are briefly discussed and the metrics are explained.

The original RMC formulation assumes the dispatching problem is a graph in which depots and customers are nodes and a delivery is depicted by an arc between a depot and a customer. To retain the unity throughout the formulation and the algorithm, all required parameters are defined in (Table 1).

\[
\text{Minimize } \sum_{u} \sum_{v} \sum_{k} z_{uvk} x_{uvk} - \sum_{c} \beta_{c} y_{c}
\]

Subject to:
\[
\sum_{u \in U_{u}} \sum_{v} \sum_{k} x_{uvk} = 1 \quad \forall \ k \in K
\]

\[
\sum_{u} \sum_{v \in V_{f}} \sum_{k} x_{uvk} = 1 \quad \forall \ k \in K
\]

\[
\sum_{u} \sum_{v} x_{uvk} - \sum_{v} \sum_{w} x_{uvwk} = 0 \quad \forall \ k \in K, \ v \in C \cup D
\]
Table 1. Parameters definition

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>set of customers</td>
</tr>
<tr>
<td>D</td>
<td>set of depots</td>
</tr>
<tr>
<td>K</td>
<td>set of vehicles</td>
</tr>
<tr>
<td>U_s</td>
<td>set of starting points</td>
</tr>
<tr>
<td>V_f</td>
<td>set of ending points</td>
</tr>
<tr>
<td>S(u)</td>
<td>service time at the depot u</td>
</tr>
<tr>
<td>t(u,v,k)</td>
<td>travel time between u and v with vehicle k</td>
</tr>
<tr>
<td>q(k)</td>
<td>maximum capacity of vehicle k</td>
</tr>
<tr>
<td>q(c)</td>
<td>demand of customer c</td>
</tr>
<tr>
<td>W_o</td>
<td>time at location o</td>
</tr>
<tr>
<td>βc(c)</td>
<td>penalty of unsatisfying the customer c</td>
</tr>
<tr>
<td>M</td>
<td>a big constant</td>
</tr>
<tr>
<td>ϖ</td>
<td>maximum time that concrete can be hauled</td>
</tr>
<tr>
<td>x(u,v,k)</td>
<td>1 if route between u and v with vehicle k is selected, 0 otherwise</td>
</tr>
<tr>
<td>y_c</td>
<td>1 if total demand of customer c is supplied, 0 otherwise</td>
</tr>
<tr>
<td>Z(u,v,k)</td>
<td>cost of travel between u and v with vehicle k</td>
</tr>
</tbody>
</table>

\[
\sum_{u \in D} \sum_{k} x_{u v k} \leq 1 \quad \forall v \in C \quad (5)
\]
\[
\sum_{v \in C} \sum_{k} x_{u v k} \leq 1 \quad \forall u \in D \quad (6)
\]
\[
\sum_{u \in D} \sum_{k} q_k x_{u v k} \geq q_c y_c \quad \forall c, v \in C \quad (7)
\]
\[
-M \left(1 - x_{u v k}\right) + s_u + t_{u v k} \leq w_v - w_u \quad \forall (u, v, k) \in E \quad (8)
\]
\[
M \left(1 - x_{u v k}\right) + \gamma + s_u \geq w_v - w_u \quad \forall (u, v, k) \in E \quad (9)
\]

The objective function (equation 1) forces optimization to find feasible solutions for all customers and penalizes if a feasible solution for customer (c) cannot be found by applying zero to \( y_c \). Therefore, due to the large value of \( \beta_c(c) \), optimization attempts to avoid unsupplied customers. The (equation 2) ensures that a truck at the start of the day leaves once from its base and similarly (equation 3) necessitates the return of a truck to the depot/its home at the end of the day. In reality, a truck arrives at either a depot or a customer then leaves that node after loading/unloading. This concept is called conservation of flow and equation (4) ensures this issue if \( u \in C \) then \( (v \in D \text{ and } j \in C) \) but if \( u \in D \) then \( (v \in C \text{ and } j \in D \cup U_f) \). In this formulation, a depot is divided into a set of sub-depots based on the number of possible loadings. Similarly, a customer is divided into a set of sub-customers according to the number of required deliveries. To simply the text, from here on a depot means a sub-depot which can load a truck only at a specific time; similarly, a customer means a sub-customer that requires a delivery only at a specific time. Therefore, (equation 5) and (equation 6) respectively certify sending a truck only to a customer and a depot only supplies a customer. (equation 7) checks demand satisfaction for customers. (equation 8) and (equation 9) are designed to control timing issues. (equation 8) ensures that concrete will be supplied to customers within
the specified time, and similarly equation (9) ensures that the travel time for each
customer will not exceed the permitted time for delivery (Υ) because fresh concrete is
a perishable material and it is not possible to haul it more than (Υ) which varies
according to the type of concrete. The time window is not applicable when \( w_o \) is a
fixed number. This means that each order must be delivered at the requested time.
The opposite applies if \( u_{bo} \leq w_o \leq l_{bo} \) has the RMC dispatching model with a time
window. This means that time at node \( o \) there is an upper bound and also a lower
bound.

As mentioned above, introduced heuristic methods in the literature achieve
the solution by defining the source and prioritizing customers. For example, in GA,
ACO, DPSO, BCO or TS based methods the solution array consists of two equal
merged parts where the length of each part is equal to the number of customers and a
cell in each part belongs to a customer. This means that each customer has two cells
in the solution array which respectively express the source and priority of each
customer. When the solution is achieved, the feasibility of the solution is checked and
the infeasible solutions are allocated via out-sources or idle resources. The
infeasibility occurs when there is no match between the acquired priorities and
available resources. Although there are different versions of this approach in the
literature, most of the steps in the introduced methods are pretty much the same.
However, (Maghrebi, Waller, et al. (2013)) presented a new way of constructing the
structure of the solution. Although their idea was only tested with GA, it can be
applied to all discrete evolutionary techniques such as ACO, DPSO, BCO or TS. This
can be done by changing only the evaluation algorithm while the proposed structure
is used. According to (Maghrebi, Waller, et al. (2013)), the solution structure for an
RMC that supposes to supply \( i \) customers consists of a chromosome with \( 2 \times i \) gens.
The gens 1 to \( i \) are intended to find depot allocation for customers 1 to \( i \) and gens \( i+1 \)
to \( 2 \times i \) are dedicated to finding the most efficient way of allocating trucks for
customers 1 to \( i \). They also presented a new way of producing the initial population in
order to avoid infeasibilities in the first generation.

Column Generation is an efficient algorithm for solving large scale linear
programs and integer programs to obtain near optimal solutions. Since most of the
variables of the optimal solution are non-basic and assume a value of zero, only a
subset of variables needs to be considered. Column Generation leverages this
principle and generates only those variables that have the potential to improve the
objective function: that is, those columns that have the most negative reduced cost in
the minimization context. Column Generation solves a sequence of master and
pricing problems. The duals from the optimal solution of the master problem are used
in the computation of reduced costs. The pricing problem is the minimization of the
reduced costs and from each solution of the pricing problem to optimality or near
optimality, one or more negative reduced cost columns are generated. A feasible
solution from each solving of the pricing problem is added to the master problem, and
the process is repeated until no more negative reduced cost columns can be generated.
Upon termination of the Column Generation phase, the original problem is solved
with the generated columns using the branch-and-cut algorithm.
RESULTS AND DISCUSSION

Both of the selected algorithms were tested with a data set and using the same metrics. 6 instances were selected randomly from the available field data of an active RMC in Adelaide (Australia). The data set covers a period of two months. This RMC has 4 active batch plants and around 50 trucks in this area. The data set covers two months and the minimum and maximum numbers of deliveries in a day are respectively 19 and 198. In more than 70% of instances, it is necessary to send more than 100 deliveries. Also, on 4 days the demand is fewer than 50 deliveries. Figure 1 depicts the number of deliveries for all the days in the field data. This figure helps shows the size of instances available in the field data and also the size of the RMC in Adelaide.

![Figure 1. Number of deliveries for all days in the dataset](image)

Both the CG and the Robust-GA are assessed from two main perspectives: the quality of the solutions and the speed of acquiring the solution. For quality, the sum of travelled distances between depots to customers is selected as the index. This reflects the size of the operational costs and obviously a lower delivery cost is desirable. For speed, the elapsed time of the computing process is considered as metric. The summary of the results for all selected instances is embedded in Table 2.

<table>
<thead>
<tr>
<th>Table 2. Summary of the results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

In terms of cost, as expected the CG outperforms Robust-GA because CG uses techniques that are used for acquiring optimum solutions. On average, the CG achieved solutions at 20% less cost than Robust-GA. However, in terms of time, Robust-GA converges faster. On average, Robust-GA is 40% faster than the CG. The interesting point is that by increasing the size of the instances, the gap between the elapsed times of the CG and Robust-GA increases exponentially, while the difference between the achieved costs by the CG and Robust-GA for all instances is in a range. Figure 2 illustrates this issue. Although the CG obtains a solution at a lesser cost than Robust-GA, in terms of the number of unassigned customers the performances of the...
CG and Robust-GA are the same. This means that the Robust-GA is able to provide a feasible solution for around 99% of customers, even for large instances, while dealing with infeasibilities among the solution is one of the challenges of the presented evolutionary techniques.

![Figure 2. Comparison between elapsed time and the size of the instances](image)

CONCLUSION

The concrete delivery industry needs to ensure practical resource allocation techniques. With the available computing facilities optimal solutions for large scale Ready Mixed Concrete (RMC) problems cannot be found. Two types of solutions have been recommended to solve RMC dispatching problem: (i) evolutionary techniques and (ii) numerical techniques. In this paper we selected a technique from each category and compared them in the same conditions. Robust Genetic Algorithm (Robust-GA) and Column Generation (CG) were selected and tested with different sizes of real RMC problems. The results show that on average CG obtains solutions at 20% lesser cost. However, Robust-GA converges 40% faster than CG, while the number of unassigned customers for both the techniques is almost the same.

REFERENCES


