A Real Geographical Application for the School Bus Routing Problem

Marcelo Fonseca Faraj\textsuperscript{1}, João Fernando Machry Sarubbi\textsuperscript{1}, Cristiano M. Silva\textsuperscript{2}, Marcelo Franco Porto\textsuperscript{3} and Nilson Tadeu Ramos Nunes\textsuperscript{3}

\textbf{Abstract}—This work presents our research and implementation of the School Bus Routing Problem applied to the rural area of a Brazilian city. We use a complete set of real georeferenced data containing a sample of 944 students, 23 schools, and the full road network of a city of population of 280,000 inhabitants occupying an area of 2,348 km\textsuperscript{2}. Our goal is to optimize the daily transportation of students considering the real publicly available fleet composed of heterogeneous vehicles. As our solution is being evaluated for application over the entire country, we have included some ‘quality’ parameters for tuning the application to distinct economic/social realities. A mixed integer linear programming model and a GRASP based algorithm are proposed to solve the routing problem.

\section{Introduction}

School Bus Routing Problem (SBRP) \cite{5,11,12,14,17} is a classical combinatorial optimization problem which consists of generating routes of school buses given a set of roads, schools, students, vehicles and garages.

School busing is not an option for many children, but the unique means of transportation to schools. In Brazil, just the state of Minas Gerais transports 800,000 rural students every day incurring in high costs for municipalities.

In this work we propose a solution to optimize the students transportation using a complete set of real georeferenced data containing 944 students, 23 schools, and the entire road network of a Brazilian city. Based on real data, we have assumed municipalities with an heterogeneous fleet composed of three-capacity buses: 15, 30, and 40 passangers.

We have implemented an exact and a heuristic approach to generate bus school routes. Our exact approach is a new mathematical mixed integer flow formulation that generates optimal routes for the problem. Our heuristic approach is a GRASP-based algorithm \cite{7} \cite{16}.

Our results demonstrate that our strategy offers a close-to-optimum performance in an extremely competitive computational time.

The remainder of the paper is organized as follows: Section II describes the School Bus Routing Problem (SBRP) and its five sub-problems. Section III describes the algorithms used to solve the problem. Section III-B.1 describes the heuristic approach based and a GRASP based algorithm. Section III-B.2 shows the mixed integer linear formulation of the route generation step of the SBRP. Section IV presents the computational results, and Section V presents the final remarks.

\section{The School Bus Routing Problem}

The School Bus Routing Problem was first proposed by Newton and Thomas in 1969 \cite{12}. According to Park and Kim \cite{14}, the SBRP seeks to plan an efficient schedule for a fleet of school buses where each bus picks up students from various bus stops and delivers them to their designated schools while satisfying various constraints such as the maximum capacity of a bus, the maximum riding time of a student in a bus, and the time window of a school.

The SBRP can be divided into five sub-problems: (i) data preparation; (ii) bus stop selection; (iii) bus route generation; (iv) school bell time adjustment; and (v) route scheduling \cite{5}.

\textbf{Data preparation} \cite{10} consists of generating a single network containing students residences, schools and garages.

\textbf{Bus stop selection} \cite{3,5,6} considers previous network in order to determine the location of stops. Some constraints may also be used such as ‘maximum walk distance from residence to bus stop’ \cite{3}. It also assigns each student to nearest stop. It is possible to affirm that the bus stop selection step consists of two phases: (i) creation of buses stops; (ii) assignment of students to a bus stop.

\textbf{Routes generation} is the most important step of SBRP: data from the previous steps are processed in order to generate the set of best routes. This step may be reduced to an instance of classical Vehicle Routing Problem (VRP) \cite{2,18,19}. Routes generation can be classified according to the fleet type: (i) heterogeneous \cite{8}; (ii) homogeneous \cite{13} \cite{5}. We can also classify the routes generation according to the type of load: (i) mixed load \cite{15}, when students from different schools can use the same bus; (ii) single load, when the bus transports students from a single school.

\textbf{School bell time adjustment} \cite{5} and \textbf{route scheduling} \cite{13} are required when buses transport students from more than one school. These steps are responsible for (i) adjusting the time arrival of students at schools, and (ii) setting the exact sequence in which each bus route will be accomplished.

A few papers in the literature deal with all stages of the SBRP \cite{14}. Newton and Thomas \cite{12} solve the route generation through a Traveling Salesman Problem (TSP) \cite{4} heuristic. Desrosiers et al. \cite{5} address all five steps and Chapleau et al. \cite{3} examine the first three steps...
for application in urban areas. Park and Kim [14] note that route generation step is the most researched, while school bell time adjustment is the least one.

In this work we focus on SBRP and our goal is to minimize the distance traveled by all buses. Our solution addresses multiple schools, students from rural areas, heterogeneous fleet, and single load. Our constraints are: vehicle capacity, maximum riding time of a student in a bus and maximum walking distance from home to bus stop.

III. ALGORITHM

Our scenario is presented in Figure 1, and it requires the application of the first three steps of the School Bus Routing Problem. We assume each school bus just picks up or delivers students from a single school.

Data preparation step is responsible for projecting the home of students and schools to the nearest road. Such issue arises because our source database contains the exact georeferenced location of each entity, and we have to bring these entities to the road network. After projecting entities, we partition the street (edge) considering the point of projection. Suppose a given road $\Gamma$ from points $\rho_1$ to $\rho_2$. Suppose also that road $\Gamma$ holds a student house. Then, the student house is projected at road $\Gamma$ at point $\rho_3$. After that, we split road $\Gamma$ in two distinct roads: road $\Gamma_1$ from points $\rho_1$ to $\rho_3$ and road $\Gamma_2$ from points $\rho_3$ to $\rho_2$.

Fig. 1. Geographical Map.

Buses Stops Generation & Students Assignment

As previously mentioned, bus stop selection has two phases: (a) generation of bus stops; and, (b) assignment of students to bus stops. Such steps requires us to split the students in two distinct sets: (a) ’groupable’ students; (b) ’non-groupable’ students.

Set ’groupable’ contains all students having at least one student distant less than $\beta m$, where $\beta$ is maximum walking distance from the student’s residence. We define bus stops for the students belonging to ’groupable’ using a heuristic for Dominant Set Problem. Set ’non-groupable’ holds isolated students: a bus stop is created in front of the residence of every student belonging to ’non-groupable students’.

Heristic for Dominant Set Problem consists of computing a priority $P$ for each vertex of a graph $G=(V,E)$, where $V$ represents students residences and $E$ represents the road network. Priority is directly proportional to square of the vertex degree and inversely proportional to the sum of the degrees of neighbor vertexes. Vertexes are interactively inserted into dominant set $DS$ according to its priority (highest to lowest). At each step we remove all redundancies from $DS$. Heristic has computational cost $\Theta(|Students| \times |Stops|)$. Thereafter, we compute the geographical distance between every pair of entities using a shortest-path algorithm.

Figure 3 shows an example of assignment of students to buses stops: circles represent students, squares represent stops and the color represents ’assignment’: students assigned to a given stop share the same color of the stop.

Fig. 2 shows an example of the projection: green points represents exact location of entities. Red points represent their projection on nearest road. After projecting all entities, we are able to generate the road network consisting of all road segments, schools, students house and available stops.

B. Routes Generation

Routes generation is the most significant phase of the algorithm, responsible for defining the path followed by each bus. We have implemented a single load algorithm where each bus just holds students from a single school. We assume that buses always depart from the school, and our approach is able to handle a variable number of vehicles categories. Route generation is based on two approaches:

- GRASP\(^1\)-like heuristic;
- Mixed integer linear model.

\(^1\)Greedy Randomized Adaptive Search Procedure.
1) GRASP Algorithm: We use a version of the well-known Greedy Randomized Adaptive Search Procedure (GRASP) [7] [16]. This metaheuristic consists of two phases: (a) construction phase; (b) local search phase. Both phases are repeated for each iteration. Construction phase consists of a randomized greedy function building up an initial solution. Solution is then used in the local search. Final result is simply the best solution found over all iterations. Algorithm 2 presents our GRASP-like heuristic.

**Algorithm 1: GRASP**

Data: Max_Iterations

for it ← 1 to Max_Iterations do
  Construction_Phase();
  Local_Search();
end

In construction phase, a randomized greedy technique provides feasible solutions. This feasible solution is iteratively constructed, one element at a time. We have selected Nearest Neighbor Algorithm [1,9], but instead of always selecting best solution, we built a restricted candidate list (RCL) of good elements, and we randomly selected one of them (not necessarily the top candidate). A RCL parameter $\alpha$ determines the level of greediness or randomness in the construction phase. When $\alpha$ is zero we have a full greedy solution. When $\alpha$ is one we have a total random solution. Algorithm 3 presents our construction phase algorithm.

**Algorithm 2: Construction Phase**

Data: $\alpha, N, Garage, G = (V, E)$

Initial_Solution = {Garage};
Stop = Garage;
for i ← 2 to N do
  Build_RCL(Vertex,$\alpha$);
  Stop = Randomly_Choose_RCL_Element();
  Initial_Solution = Initial_Solution ∪ {Stop};
end

Our local search is also different from the most local search already used. Instead of using a single local search algorithm we proposed the use of a combination of two simple local search methods based in a Descent UpHill method [20].

**Algorithm 3: Local Search**

Data: it, $G = (V, E)$

2-opt_Procedure();
Shift().Procedure();
if Found_Better_Solution() then
  Update_Better_Solution();
  Local_Search();
end

First, a restricted 2-exchange method [9] is applied to improve the solution of the first phase. Our second local search method is an insertion algorithm. Given a partial solution from construction phase, this local search works as follows: each new element is inserted into all possible positions, shifting neighbor elements. This procedure is executed for all elements belonging to the partial solution. An example is shown in Figure 5.

2) Mathematical model: In order to compare the quality of the solution given by the GRASP-like heuristic we have also implemented a linear mixed integer formulation to solve the route generation step. Suppose the sets:

- $B$: set of buses where $B = \{1, 2, ..., b\}$;
- $P$: set of stops where $P = \{0, 1, ..., p\}$;
- $A$: set of students where $A = \{1, 2, ..., a\}$.

Our goal is to generate routes considering sets $A,B$ and $P$ in order to minimize the total distance traveled by all buses. For this problem we can define a mixed-integer linear programming formulation $M$ with the following sets of variables:

- $x_{ijk} = \begin{cases} 1 & \text{if the arc } (i, j) \text{ is visited by the bus } k \\ 0 & \text{otherwise.} \end{cases}$
- $v_{jk} = \begin{cases} 1 & \text{if the stop } j \text{ is visited by the bus } k \\ 0 & \text{otherwise.} \end{cases}$
- $o_k = \begin{cases} 1 & \text{if the bus } k \text{ is used} \\ 0 & \text{otherwise.} \end{cases}$
- $y_{ijk}$: number of students transported throw arc $(i, j)$ by bus $k$.

and the following set of parameters:

- $d_{ij}$: distance between arc $(i, j)$
- $c_k$: capacity of bus $k$.
- $a_j$: number of students at stop $j$. 

![Fig. 4. 2-Exchange Local Search.](image)

![Fig. 5. Shift Local Search.](image)
Formulation. The objective function (1) consists of minimizing the sum of distances traveled by buses. Constraints 2-4 ensure the conservation of flow along all buses' tour. Constraint 5 assures the same number of buses departing and arriving at stops and 65 students. For the 67 stops and 221 students solution found by solver CPLEX is not necessarily optimal. This gap represents the maximum distance, in percentage, between the optimal solution and the solution found by solver CPLEX.

As shown in Table 1, the mixed integer linear formulation M was able to find optimal solution for instances up to 26 stops and 65 students. For the 67 stops and 221 students instance the solver CPLEX was not able to find an integer solution in 4 hours using M formulation.

Table II presents the GRASP algorithm results and comparison was made for them. This section presents the results of the exact approach and the heuristic one for the route generation of the SBRP.

GRASP algorithm used the following parameters: \( \alpha = 0.1 \) and 20,000 iterations. We use CPLEX with default parameters to compute the linear relaxation solution and the optimal integer solution. We imposed CPLEX a limit of 4hs to compute the solution.

\[
\text{min} \sum_{i \in P} \sum_{j \in P - \{i\}} \sum_{k \in B} d_{ijk} x_{ijk}
\]

Subject to:

\[
\sum_{i \in P} y_{ijk} = 0 \quad \forall k \in B \quad (2)
\]

\[
\sum_{i \in P - \{j\}} y_{ijk} - \sum_{i \in P - \{j\}} y_{ijk} = a_{jk} y_{ijk} \quad \forall j, k \mid j \in P - \{0\}, \quad k \in B \quad (3)
\]

\[
\sum_{i \in P - \{0\}} y_{ijk} \leq c_{ik} a_{jk} \quad \forall k \in B \quad (4)
\]

\[
\sum_{j \in P - \{0\}} v_{jk} = 1 \quad \forall j \in P - \{0\} \quad (5)
\]

\[
\sum_{i \in P - \{0\}} x_{ijk} = 1 \quad \forall i \in P - \{0\} \quad (6)
\]

\[
\sum_{k \in B} \sum_{j \in P - \{0\}} x_{ijk} = 1 \quad \forall i \in P - \{0\} \quad (7)
\]

\[
y_{ijk} \leq c_{k} x_{ijk} \quad \forall i, j, k \mid i, j \in P \quad (8)
\]

\[
\sum_{k \in B} \sum_{j \in P - \{0\}} x_{0jk} = \sum_{k \in B} a_{0k} \quad (9)
\]

\[
\sum_{k \in B} \sum_{j \in P - \{0\}} x_{jk} = \sum_{k \in B} a_{0k} \quad (10)
\]

\[
\sum_{i \in P - \{0\}} x_{ijk} = \sum_{i \in P - \{0\}} x_{ijk} \quad \forall j, k \mid j \in P, \quad k \in B \quad (11)
\]

\[
y_{ijk} \geq 0 \quad \forall i, j, k \mid i, j \in P \quad (12)
\]

\[
\sum_{j \in P, k \in B, i \neq j} d_{ijk} x_{ijk} \leq m \quad \forall k \in B \quad (13)
\]

Table I specifies the major issues of twelve instances solved using the two approaches: GRASP algorithm and solver CPLEX. Our sample consisting of 944 students using the rural bus school transportation and 23 schools. However, some schools have just a few students using the rural bus school transportation. In this work we will just present the results for the 12 most significative schools of the database.

These schools are responsible for 868 rural students. We also assume that all students are in the same shift. Field ID represents the school (instance). Fields \( \text{Stops} \) and \( \text{Students} \) represent the number of stops and the number of students associated with the school. \( \text{CPLEX Time} \) (s) represents the time, in seconds, that the solver CPLEX spend to find solution. \( \text{CPLEX solution (m)} \) shows the distance (meters) necessary to pickup all students. \( \text{CPLEX Gap ( %)} \) computes the gap given by the solver CPLEX. When this gap is zero means that CPLEX found the optimal solution. When CPLEX gap is different from zero means that the solution found by solver CPLEX is not necessarily optimal. This gap represents the maximum distance, in percentage, between the optimal solution and the solution found by solver CPLEX.

As shown in Table 1, the mixed integer linear formulation M was able to find optimal solution for instances up to 26 stops and 65 students. For the 67 stops and 221 students instance the solver CPLEX was not able to find an integer solution in 4 hours using M formulation.

Table II presents the GRASP algorithm results and compare the GRASP solution with CPLEX solution. ID represents the instance (school). \( \text{GRASP Time (s)} \) shows the total time that GRASP algorithm takes to compute the 20,000 iterations. \( \text{GRASP Best Iteration} \) shows the iteration that the best solution was found. \( \text{GRASP Time}^* \) (s) shows the time
spent by GRASP algorithm to find the best solution. **GRASP Solution** shows the total distance found by algorithm GRASP. **Gap (%)** **CPLEX GRASP** shows the difference between the GRASP solution and CPLEX solution. When this gap is zero means that both strategies have reached the same solution.

Our GRASP algorithm finds competitive solutions in very short computer time when compared to CPLEX. Our experiments also demonstrate that CPLEX is unable to find feasible solutions for very large instances.

In this work we also analyze the convergence of both strategies (CPLEX and GRASP) according to the number of iterations and computer time. In order to present our analysis we have selected a single school at random (school #9).

<table>
<thead>
<tr>
<th>ID</th>
<th>GRASP Time (s)</th>
<th>Best Iteration</th>
<th>GRASP Time* (s)</th>
<th>GRASP Solution (meters)</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.76</td>
<td>3</td>
<td>0.0005</td>
<td>31140.10</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>3.68</td>
<td>2</td>
<td>0.0006</td>
<td>130531.0</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>2.88</td>
<td>29</td>
<td>0.0045</td>
<td>104573.0</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>3.33</td>
<td>14</td>
<td>0.0028</td>
<td>201236.0</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>3.88</td>
<td>3</td>
<td>0.0008</td>
<td>135690.0</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>2.74</td>
<td>1973</td>
<td>0.2702</td>
<td>639603.0</td>
<td>1.10</td>
</tr>
<tr>
<td>7</td>
<td>7.87</td>
<td>1224</td>
<td>0.4997</td>
<td>244444.0</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>15.74</td>
<td>3729</td>
<td>2.917</td>
<td>149976.0</td>
<td>0.90</td>
</tr>
<tr>
<td>9</td>
<td>17.76</td>
<td>13375</td>
<td>11.87</td>
<td>227042.0</td>
<td>3.51</td>
</tr>
<tr>
<td>10</td>
<td>13.02</td>
<td>737</td>
<td>0.4698</td>
<td>752178.0</td>
<td>2.88</td>
</tr>
<tr>
<td>11</td>
<td>15.45</td>
<td>17960</td>
<td>13.87</td>
<td>510369.0</td>
<td>7.38</td>
</tr>
<tr>
<td>12</td>
<td>61.82</td>
<td>19601</td>
<td>60.60</td>
<td>1067050</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table II**

**GRASP x CPLEX RESULTS**

Figure 6 presents convergence of GRASP approach according to the number of iterations. In the first iterations the algorithm GRASP converges fast, while the last iteration are less productive.

Figure 7 presents the convergence of CPLEX algorithm according to the number of iterations. The gap between 'lower bound' and 'upper bound' are tight from iteration 400,000 to 17,000,000, i.e., CPLEX takes approximately 95% of the total number of iterations just to prove optimality.

Figure 8 indicates the convergence of 'lower bound' and 'upper bound' in CPLEX according to time.

Figure 9 indicates the computer time of both strategies to find near optimum routes for a given school: while our Grasp approach requires one minute to find a near optimal solution, CPLEX takes 17 minutes (a difference of three orders of magnitude).

Figure 10 shows the calculated routes for a given school by GRASP Algorithm. The red triangle shows the school position and the yellow circles represents the stops.

**V. CONCLUSION REMARKS**

In this paper we present School Bus Routing Problem (SBRP) applied to the rural area of a brazilian city. We have used a real georeferenced database with schools, roads network and a sample of 944 students. We assume heterogeneous fleet with multiple schools in a single load version. Because we are targeting real deployment of our solution, we have implemented two constraints to model economic/social aspects:

- 'maximum walking distance' for home to bus stop;
- 'maximum travel distance' for students tripping.

Our algorithm solves the first three steps of the SBRP (data preparation, bus stop selection, bus route generation). For the bus stop selection we propose an algorithm based on the dominant set in order to assign students to stops. The routes generation use two approaches: exact, and heuristic. Our exact approach is a mixed integer linear formulation solved by CPLEX. The heuristic approach is
a GRASP based algorithm that finds good solutions in a extremely competitive computational time. Besides, the GRASP algorithm is able to find good quality solutions for all instances. On the other hand, solver CPLEX was not able to find an integer solution for very large instances. As our results demonstrate the gap between GRASP solution and CPLEX is never above 10%. Our next steps on SBRP concentrate on improving the system focusing on the deployment of our solution to entire country. It is possible to add some valid cuts in the model to speed up the convergence of the solver CPLEX. It is also possible to implement a branch-and-cut algorithm to solve this problem.

ACKNOWLEDGMENT

This work was partially supported by CNPQ, Laboratory NucleTrans, and MEC/FNDE Project.

REFERENCES