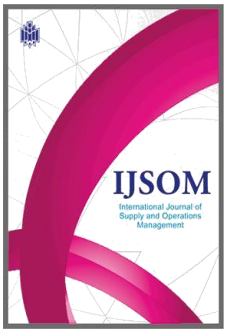
**International Journal of Supply and Operations Management**

# IJSOM

**February 2017, Volume 4, Issue 1, pp. 1- 14**
**ISSN-Print: 2383-1359**
**ISSN-Online: 2383-2525**
**www.ijsom.com**




## Iterated Local Search Algorithm with Strategic Oscillation for School Bus Routing Problem with Bus Stop Selection

Mohammad Saied Fallah Niasar [a, *], Luca Talarico [a], Mehdi Sajadifar [b] and Amir Hosein Tayebi [a]

[a] *Department of Engineering Management, Faculty of Applied Economics, University of Antwerp,* Belgium
[b] Department of Industrial Engineering, University of Science and Culture, Tehran, Iran

**Abstract**

The school bus routing problem (SBRP) represents a variant of the well-known vehicle routing problem. The main goal of this study is to pick up students allocated to some bus stops and generate routes, including the selected stops, in order to carry students to school. In this paper, we have proposed a simple but effective metaheuristic approach that employs two features: first, it utilizes large neighborhood structures for a deeper exploration of the search space; second, the proposed heuristic executes an efficient transition between the feasible and infeasible portions of the search space. Exploration of the infeasible area is controlled by a dynamic penalty function to convert the unfeasible solution into a feasible one. Two metaheuristics, called N-ILS (a variant of the Nearest Neighbourhood with Iterated Local Search algorithm) and I-ILS (a variant of Insertion with Iterated Local Search algorithm) are proposed to solve SBRP. Our experimental procedure is based on the two data sets. The results show that N-ILS is able to obtain better solutions in shorter computing times. Additionally, N-ILS appears to be very competitive in comparison with the best existing metaheuristics suggested for SBRP.

**Keywords:**
School Bus Routing Problem; Combinatorial Optimization; Iterated Local Search Algorithm; Strategic Oscillation.

### 1. Introduction

Transporting students to and from a school is a challenging problem for any local government trying to optimize its budget. This requires an efficient coordination and planning of an urban transportation network. The problem of student transportation is qualitatively different from traditional vehicle routing problems, since "students are not simple packages, as in the case of pick-up and delivery of goods, and because these services are provided through the public sector (Bowerman & Calamai, 1995)". There is a large body of research conducted on this problem, called School Bus Routing Problem (SBR). In the general definition, SBRP seeks to transport students to and from school in a safe and convenient way. Considering real-life applications, SBR needs to take into account additional constraints as well as factors linked to the usability of the transportation network. For these reasons, SBRP has a more complex formulation and, in general, it requires more complex solution strategy than Vehicle Routing Problem (VRP). Overall, traditional VRP aims at finding a set of optimal routes for vehicle, in which each vehicle (the bus for SBRP) travels from a given depot (the school in the SBRP), serves a given set of customers (the stops where students wait for the bus to reach the school in the SBRP), and returns to the same depot (the school in the SBRP) at the end of the route. Among the many variants that have been made in connection with school bus routing problem, this paper deals just with a single school without time window. The SBRP under consideration consists of the following decisions. A set of potential bus stops needs to be created, such that they present at least a maximum distance from the locations where students actually live collecting students. Afterward, the definition of optimized bus routes consists of the selection of these bus stops to guarantee that all students are able to reach the school while satisfying the maximum capacity of the buses. Therefore, SBRP can be decomposed into three sub-problems: (1) the selection of the minimal set of bus stops to be included in

*Corresponding author email address: fallah.pasco@gmail.com

each route, which allows the collection of all students; (2) the allocation of students to stops such that the bus capacity is not exceeded, and (3) the definition of bus routes, which includes the selected bus stops so that the total length of routes is minimized. This paper addresses these three sub-problems, embedding them into one single optimization procedure. In previous studies, significant efforts have been made to develop heuristics that explore only the feasible search space. However, a variety of problems can be efficiently solved if the solution algorithm considers both feasible and infeasible solutions in a wider search area. For instance, Brandao (2006) presented a strategy that implemented efficient switching between feasible and infeasible search areas. This approach has been called Strategic Oscillation. It is obvious that carrying out strategic oscillation requires an efficient use of large neighborhood operators. It has to be noted that the main component of this algorithm is referred to as oscillating local search (OLS) strategy that relies on the ability of infeasible portions of the search space. As mentioned, the aim of the oscillation is to continually move from the feasible to the infeasible space and vice versa, having at the same time a span to control the number of movements between infeasible and/or feasible states. This transition is set by considering a dynamically adjusted penalty which is applied to infeasible solutions. To the best of our knowledge, the application of strategic oscillation in an SBRP context is new. In this paper, we develop an iterated local search algorithm that uses a strategic oscillation for transition between feasible and infeasible solutions. This approach embeds six neighborhood structures in a variable neighborhood descent methodology.

The two heuristics of insertion iterated local search and the nearest neighborhood iterated local search have been abbreviated as I-ILS and N-ILS, respectively. To test and validate the proposed metaheuristics, we used sets of instances for SBRP. The remainder of the paper is organized as follows. In Section 2, the literature concerning SBRP and related problems is briefly discussed. In Section 3, the problem is described and its mathematical formulation is given. Section 4 illustrates the solution approach and metaheuristic configuration. Computational experiments are presented in Section 5. Finally, Section 6 concludes the paper and provides some suggestions for future research.

## 2. Literature review

Many variants of SBRP have been suggested by researchers in recent decades. A comprehensive description of school bus routing problem and a relatively broad survey could be found in Park and Kim (2010). In their study, they categorized the existing works in the literature on SBRP in five categories: data preparation, bus stop selection, bus route generation, school bell time, and bus scheduling.

Considering the objective functions used in SBRP, the majority of researchers aim at minimizing the number of buses used in the solution and the total distances traveled by buses. Only a few SBRP studies have considered load balancing and maximum route length as an objective function. Some applications could be found in Angel et al. (1972), Newton and Thomas (1974), Verderber (1974), Gavish and Shlifer (1979), Bodin and Berman (1979), Dulac et al. (1980), Desrosiers et al. (1981), and Swersey and Ballard (1984). Also, a number of authors have attempted to incorporate some typical features of SBRP into the objective function. For instance, Bowerman et al. (1995) considered the minimization of total walking distance of students.

Regarding the solution approach point of view, it has to be pointed out that the most efficient heuristic methods for SBRP recently published are Tabu search (Pacheco et al., 2013) and approximation algorithm (Bock et al., 2011).

Multi vehicle covering route problem and capacitated m-ring star problem are similar to SBRP. In the former case, the total route length and the number of stops to be contained in the route are limited (Hachicha et al., 2000). Capacitated m-ring star problem (Baldacci et al., 2007) differs from SBRP in that it supposes customers and transition points to represent students and stops, respectively. Besides, in the capacitated m-ring star, there is no limitation concerning which student can be allocated to which stop. Lastly, in this problem, the number of rings (routes) is predefined.

Later studies have addressed the composite nature of SBRP which can be solved through different sub-problems by using two different solution approaches: Allocation–Routing-Location (ARL) and Location- Allocation- Routing (LAR). In the LAR strategy, bus stops are determined in the first phase, and then students are assigned to those stops. Afterward, bus routes are generated. Thus, both bus stops selection and student allocation to the stops are independent of route generation, resulting in drawbacks such as an exceeding number of routes. For more details, the reader is referred to Bodin and Berman, 1979; Dulac et al., 1980; Desrosiers et al., 1981 and 1986a.

To overcome this problem, a variant of ARL strategy is used that considers students who are grouped in clusters. These latter are designed based on one single route which satisfies the capacity constraint. Subsequently, for each cluster, bus stops selection and route generation to visit these stops will occur. Lastly, students in the cluster are allocated to the bus stops. This way the method can surmount some problems associated with LAR strategy. Chapleau et al. (1985) and Bowerman et al. (1995) proposed some heuristic algorithms using the ARL strategy.

Efficient application of ARL approach can be found in Bowerman et al. (1995), where a multi-objective formulation of the urban school bus routing problem is presented. The problem involves minimizing several objectives such as the number of routes, the total route length, a variety of students assigned per each route, the variation in route length, and the total walking distance covered by students from their homes to bus stops. The advantage of applying ARL strategy is that it allows an efficient load balance in allocating students to each cluster. Moreover, the number of bus routes can be kept to the minimum level because both objectives mentioned above (minimization of the number of routes and load balancing) are independent of both bus stop locations and the routes generated to serve these stops. The only drawback concerns balancing the route length. In fact, due to the potential dispersion of students within a cluster, ARL strategy cannot efficiently cope with this objective. From another point of view, ARL and LAR strategies have so far represented two alternative approaches for "bus route generation." Location, allocation, and routing denote sub-problems (1), (2), and (3) discussed in the introduction.

Concerning the vehicle routing problem, most strategies employed for the generation of routes use either first-route-then-cluster (or ARL) or first-cluster-then-route (or LAR) approaches. Both of these strategies possess a heuristic nature. Bodin and Berman (1979) implemented "first-route–then-cluster "approach. In this method, a long route visiting all the nodes in the network is first generated by solving a traveling salesman problem heuristic. Secondly, the long route is partitioned into smaller routes to satisfy some predefined constraints (e.g. bus capacity, the maximum travel time of students, and the maximum number of students who could be allocated to an allowable bus stop).

In the strategy known as first-cluster–then-route, initially students are grouped in the clusters. Subsequently, the number of stops is determined for each cluster, and a route is generated for each cluster while satisfying predefined constraints. Several studies investigating first-cluster–second-route strategy have been carried out (Dulac et al., 1980; Chapleau et al., 1985; Bowerman et al., 1995). All researchers have adopted a two-step strategy, namely ARL and LAR in order to deal with SBRP by solving its sub-problems separately. However, both approaches offer a weak global optimization solution method. An interesting area of research, which has not received much attention so far, concerns the idea of considering bus stop selection and route generation simultaneously. Based on the argument mentioned above, it is evident that a solution method which combines the advantages of both bus stop selection the route generation could be very valuable to SBRP. This integrated approach has been examined by Schittekat et al. (2013), Riera-Ledesma. (2013), and Kinable et al. (2014). Application of strategic oscillation is not new in the literature on vehicle routing problem; however, no study has been conducted in the area of SBRP. In this paper, we take a closer look at investigating SBRP problem, proposing two effective metaheuristic approaches. The objective of our problem is to minimize total bus travel distance. Our contribution is fivefold: 1) Solving SBRP by using an iterated local search heuristic that makes use of a strategic oscillation. 2) Suggesting two metaheuristics, called N-ILS (a variant of the Nearest Neighbourhood with Iterated Local Search algorithm) and I-ILS (a variant of Insertion with Iterated Local Search algorithm). Both metaheuristics are based on three stages: **constructive** stage, **intensification** stage and **diversification** stage.3) Optimal solutions for small instances of the problem are found using CPLEX solver. 4) The key components of each metaheuristic are investigated and tuned. 5) Once the best parameter setting for each of the metaheuristics is obtained, we will compare our solution methods (I-ILS and N-ILS) with the best-known solution existing in the literature, proposed by Schittekat et al.(2013).

## 3. Problem definitions

The SBRP studied here is a generalization of the well-known vehicle routing problem (VRP), where a single school, one type of students, and identical buses with a fixed capacity are employed. The goal is to optimize the total traveling distance as in the traditional VRP. Since SBRP extends VRP, it could be considered NP-hard as well. The following table summarizes the symbols included in the model.

**Table 1.** Symbol used in mathematical model

| Table 1 Symbol used in mathematical model | |
|---|---|
| Ddata | |
| $C$ | Bus capacity |
| $V$ | Set of potential stops with $\|V\| = n$ |
| $S$ | Set of students |
| $B$ | Set of buses |
| $C_{ij}$ | Travel cost from stop i to stop j |
| $s_{il}$ | Binary variable equal to 1 if student $l$ can reach stop $i$ ,0 otherwise |
| $i = 0$ | Index for school |
| Decision variable | |
| $x_{ijk}$ | 1 if bus $k$ traverses the arc from stop $i$ to $j$ , 0 otherwise |
| $y_{ik}$ | 1 if the bus $k$ meets stop $i$ , 0 otherwise |
| $z_{ilk}$ | 1 if student $l$ is picked up by bus $k$ at stop $i$ , 0 otherwise |

The following model is built based on the formulation of Toth and Vigo (2002). A formulation of SBRP is shown below. (To conduct the formulation, we have referred to Schittekat et al., 2013).

$$\min \sum_{i \in V} \sum_{j \in V} C_{ij} \sum_{k=1}^{n} x_{ijk} \quad (1)$$

s.t.

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \qquad \forall i \in V, \mathrm{k} = 1, \ldots\ldots\ldots, \mathrm{n} \quad (2)$$

$$\sum_{k=1}^{n} y_{ik} \leq 1 \qquad \forall i \in V \setminus \{0\} \quad (3)$$

$$\sum_{k=1}^{n} z_{ilk} \leq s_{il} \qquad \forall l \in S, \forall i \in V \quad (4)$$

$$\sum_{i \in V} \sum_{l \in S} z_{ilk} \leq C \qquad k = 1, \ldots\ldots\ldots, n \quad (5)$$

$$z_{ilk} \leq y_{ik} \qquad \forall i, l, k \quad (6)$$

$$\sum_{i \in V} \sum_{k=1}^{n} z_{ilk} = 1 \qquad \forall l \in S \quad (7)$$

$$\sum_{i,j \in Q} x_{ijk} \leq |Q| - 1 \qquad \forall Q \subseteq V \setminus \{\mathrm{v}_0\}, \forall \mathrm{k} \quad (8)$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in V, \mathrm{k} = 1, \ldots\ldots\ldots, \mathrm{n} \quad (9)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i, j \in V, \mathrm{i} \neq \mathrm{j}, \mathrm{k} = 1, \ldots\ldots, \mathrm{n} \quad (10)$$

$$z_{ilk} \in \{0,1\} \qquad \forall i, j \in V, \mathrm{i} \neq \mathrm{j}, l \in S, \mathrm{k} = 1, \ldots\ldots, \mathrm{n} \quad (11)$$

The objective function (1) minimizes the total travel distance handled by all buses. Due to constraint (2), for each stop i the number of arcs entering it is exactly the same as the number of arcs going out from it. Constraint (3) guarantees that each stop is visited only once, except for stop 0 which is associated with the school. Constraint (4) enforces that each student is to be picked up at the stop where he/she walks to. Constraint (5) guarantees that the capacity of buses is not to be exceeded. Inequalities (6) impose that picking up a student in a non-visited stop by bus k is not possible. Constraint (7) states that each student is picked up only once. Constraint (8) forces the connectivity of the route performed by bus k. This constraint shows sub-tour elimination constraints. Finally, constraints (9), (10), and (11) define the domain of the decision variables which are all binary. The MIP formulation addressed in this section has been solved using CPLEX solver in GAMS software and tested on a set of small instances (Appendix A). When the instance size increases, the computing time rises, and the exact method is only able to solve 43 instances, up to 10 stops and 200 students, to optimality in less than one hour. Thus, CPLEX solver is impractical to be employed for larger instances. For this reason, we have developed a metaheuristic approach described in the next section.

## 4. Metaheuristic Configurations

Our results demonstrate that the exact method is only able to solve the easiest 43 instances in less than one hour and is not practical in the case of larger instances. Therefore, a fast and robust solution mechanism ought to be used. Furthermore, the quality of solutions, i.e. determining the minimum travel distance taken by bus, is an imperative aspect which must be guaranteed. Hence, two metaheuristics, named N-ILS and I-ILS, are developed in this paper in order to solve SBRP in a reasonable computation time and to reach near optimal solutions. The configurations N-ILS and the I-ILS are associated with the class of iterated local search (ILS) metaheuristic (see Lourenco et al., (2010)). Both metaheuristics are based on three stages: **constructive** stage, **intensification** stage, and **diversification** stage. Note that the proposed metaheuristics differ only in the **constructive** stage which is used to generate initial solutions for SBRP (see Algorithm 1). Intensification and diversification stages are the same in both metaheuristics. The **constructive** stage used in the N-ILS is based on the nearest neighborhood heuristic, while that of I-ILS uses an insertion heuristic. The solutions obtained through these heuristics serve as inputs for the intensification stage, which is based on an oscillating local search (OLS) heuristic. The latter consists of two levels: (1) improvement level by a variable neighborhood descent

(VND) heuristic, which is a variant of variable neighborhood search (VNS) (Hansen and Mladenovic, 1999) and (2) re-optimization level, which uses remove and redistribution operators. These two stages are executed sequentially (See section 4.3). The VND heuristic used during the improvement phase includes six local search operators: three intra-route operators and three inter-route operators. Intra- route operators attempt to improve the solution by changing a single route at a time.On the other hand, inter-route operators change more than one route simultaneously. The VND heuristic stop once local optima are reached. In order to further re-optimize the search process, the results achieved during the first stage are used as inputs for the second step (called re-optimization stage) which consists of two types of operations: remove and redistribution heuristics. In order to investigate unexplored areas of the solution space and escape from local optima, two diversification strategies are used (see Section 4.5). The former perturbs the current solution by partially destroying and rebuilding a restricted number of routes, while the second one applies double -swap move (1, 1) (Subramanian & Drummond, 2010). In short, two swap movements are executed in sequence randomly. In our metaheuristic, the capacity constraint is relaxed to allow an oscillation between feasible and infeasible solutions. In the remaining sections, the main components of the metaheuristic are described. Furthermore, we discuss the important components affecting the performance of the algorithm. In this section, we also focus on the analysis of metaheuristic configuration by describing the application of each element separately. In Algorithm 1, ILS metaheuristic uses the construction phase only once and iterates the local search a number of times, starting from a perturbed solution. The ILS metaheuristic terminates when a maximum number of iterations ( $\varphi$ ) is reached.

**Algorithm 1. Iterated local search (ILS)**

$S_o \leftarrow$ Generate Initial Solution; // {*Constructive phase*}

$S_{best} \leftarrow OLS(S_o)$ ; // *Improve solution using "OLS" heuristic*

***While*** ( $\varphi$ *is not reached*) ***do***

$S_{perturb} \leftarrow Perturb(S_{best})$ ; // *Perturb the best solution found so far* $S_{best}$

$S_{ols} \leftarrow OLS(S_{perturb})$ ; // *Improve solution using "OLS" heuristic*

***If*** $Cost(S_{ols}) < Cost(S_{best})$ ***then***

$S_{best} \leftarrow S_{ols}$ ;

***End if***

***End While***

Report best solution $S_{best}$ found;

### 4.1. Nearest Neighbourhood with Greedy Randomized Selection Mechanism (NNg)

A nearest neighbourhood heuristic with a greedy randomized selection process represents the first constructive heuristic developed in this paper (see Talarico, L. et al., 2015). At the beginning of the algorithm, a student allocation problem (see section 4.4) is solved for each stop. After implementing student allocation heuristic, we applied a variant of the nearest neighbourhood constructive heuristic. This heuristic is modified as follows: a greedy randomized selection mechanism is employed instead of a greedy selection process; consequently, the next stop in the current route is randomly chosen from a restricted candidate list (RCL) containing the $\alpha$ first closest non-visited stops. It should be noted that the allocation of students to stops is not executed every time that a stop is added to the current solution. After non-visited stop are selected, the capacity constraint is checked. If a feasible solution for student allocation problem is found, the addition of a new stop in the current route is evaluated. If it is not possible to include additional stops, the current route is closed,and a new route is constructed from the school to new non-visited stops.

### 4.2. Insertion Heuristic with Greedy Randomized Selection Mechanism (Ig)

The second constructive heuristic is an insertion one, similar to that presented by Campbell and Savelsbergh (2004). Like the modified nearest neighbourhood heuristic, a student allocation problem is solved for each stop at the beginning of the algorithm. After student allocation heuristic is implemented, our version of the insertion heuristic is modified in two stages. 1) A greedy randomized selection mechanism is incorporated instead of a greedy heuristic mechanism. It determines a restricted candidate list of $\alpha$ least-cost positions for inserting each non-visited stop into one of the current routes. 2) With regard to the capacity constraint, feasibility check is performed every time that the non-visited stop is considered for insertion. However, feasibility check is terminated when no feasible insertion position is found for the remaining non-visited stops. In this case, insertion happens by the cheapest possible position without considering feasibility check. It means that only stage 1 is considered. This shows that the result of the insertion heuristic may produce either a feasible solution or one with less violated capacity while the order of insertions does not lead to building a feasible solution.

### 4.3. Oscillating Local Search (LOS) Heuristic

The aim of strategic oscillation is to allow temporary exploration of the infeasible part of the solution space (see Toth & Vigo, 2003). The strategy adopts a procedure to find new promising feasible search spaces. It is important to note that

the idea of an excursion in the infeasible area is controlled by a dynamically adjusted penalty function which is multiple to the cost of the infeasible solution. The cost function described in this paper, has been introduced by Brandao (2006) and consists of two terms:

$$Cost(s,\lambda) = Cost(s) + \lambda d(s) \quad (12)$$

The first term denotes the total travel distance covered by all buses (objective function), while the second term $d(s)$ is obtained by the sum of all excess loads traveled by each bus (violations of bus capacity) and multiplied by the value of a penalty $\lambda$ .This latter is dynamically updated throughout the exploration process. If all of the bus routes are satisfied with respect to the capacity constraint, the second term is set to zero. The value of the penalty $\lambda$ emphasizes the effect of capacity violation on the total solution cost. The pseudo-code of the oscillating local search strategy is shown in Algorithm 2. OLS takes as input parameters the initial penalty $\lambda_0$ , an initial solution $S_o$ , and penalty increasing factor $\beta$ . The initial solution $S_o$ is generated by the constructive phase (described in section 4.1and 4.2) or diversification phase (outlined in section 4.5).

In the first level of Algorithm 2, OLS heuristic initially attempts to employ the improvement level performed by a variable neighbourhood descent heuristic using 6 well-known operators: three intra-route operators and three inter-route operators. Relocate, Exchange and Two-Opt as intra-route operators and Relocate, Exchange, and Two-Opt as inter-routes. All these operators are applied sequentially. This improvement procedure is repeated until a local optima solution is found (line 13 of pseudo-code). After exploring the first level, one has to evaluate the feasibility of locally optimal solution $S_{act}$ (line 14 of pseudo-code). If the locally optimal solution is not feasible, the value of $\lambda$ rises in order to better explore the feasible region (Lines 26 and 27 of pseudo-code). On the other hand, if a feasible solution is found, the algorithm works as follows. The solution obtained from the first stage (VND heuristic) serves as input for the second stage (re optimization level) for further improvement. The second phase consists of two operators: remove and redistribution operators (described in section 4.4). After the second stage is explored, verification is executed whether the solution of the second stage $S_{act}^{*\prime}$ is better than the best feasible solution found so far (Lines 18 and 19 of pseudo-code). If so, the new solution $S_{act}^{*\prime}$ is accepted and the value of $\lambda$ is set back to $\lambda_0$ (line 21 of pseudo-code). But if $S_{act}^{*\prime}$ is worse than the best feasible solution found so far, OLS heuristic stops its execution (Line 23 of pseudo-code). This is mainly because exploration has been oriented around the same feasible local optimal solution (i.e., cycling around the same feasible solution), or that the algorithm is searching around an unpromising region of the solution space which contains a weak feasible solution).

To conclude, while the algorithm explores the feasible segment of the solution space for a relatively high number of iterations, the value of $\lambda$ tends to restrict a deeper exploration of infeasible areas of the solution space. In contrast, if the algorithm explores infeasible segments of the solution space for a larger number of iterations, the value of $\lambda$ rises with the intention of guiding the exploration to a feasible region. Thus, the value of $\lambda$ could help to investigate the effectiveness of exploration in the solution space. Additionally, in OLS, the value of $\lambda$ rises when there is no feasible local optimal solution, and this value is set back to $\lambda_0$ when the best feasible solution is reached. The values of $\lambda$ and $\beta$ are two important parameters that significantly affect the performance of OLS. The value of $\lambda$ represents the capability of the heuristic to explore infeasible portions of the solution space. In other words, this value is associated with a maximum violation of the bus capacity constraint in OSL heuristic. As shown in Algorithm 2, a larger value of $\lambda$ drives the algorithm back into the feasible solution space. Hence, a smaller capability of exploring the infeasible segment of the solution space. The value of $\lambda$ is initially set to $\lambda_0$ and then systematically multiplied by the value of $\beta$ . The term $\beta$ denotes the transition speed from the infeasible to the feasible space, and it influences the capability to find feasible regions with a better solution. A high value of $\beta$ allows the penalty $\lambda$ to increase; thereby, OLS is pushed to reach a feasible solution in a shorter time.

Algorithm 2. Oscillating local search (OLS) heuristic

**Input:** initial solution $s_o$, lower end point for penalty value $\lambda_0$ ( initial penalty value $\lambda_0$ ), upper end point for penalty value $\lambda_u$, Penalty increase factor $\beta$
  If $s_o$ is feasible then
   $s_{best-feasible} = s_o$
   $cost_{best-feasible} = cost(s_o)$
  Else
   $cost_{best-feasible} = \infty$
  End if
  $\lambda = \lambda_0$
  $s_{act} = s_o$
 Stop criterion=false
    While (stop_criterion) do
 // Improvement in first level
        $S_{act} =$ Applying VND $(s_{act}, \lambda)$
          If $S_{act}$ is feasible
 // Re-optimize in second level
            $S^*_{act} =$ Remove operator $(S_{act})$
             $S^{*\prime}_{act} =$ Redistribution operator $(S^*_{act})$
                If $cost(S^{*\prime}_{act}) \le cost_{best-feasible}$ then
                $cost_{best-feasible} = cost(S^{*\prime}_{act})$
               $s_{best-feasible} = S^{*\prime}_{act}$
                $\lambda = \lambda_0$
                Else
                stop_criterion = true
                End if
          Else
            If ( $\lambda < \lambda_u$ )then
            $\lambda = \beta \times \lambda$
            Else
            $s_{act}$ =Restore operator $(s_{act}, \lambda)$
              End if
          End if
  End while
 Output: $S_{best-feasible}$

It might happen during the course of the algorithm that the value of $\lambda$ is enlarged so that it reaches the predefined value of $\lambda_u$. In this situation, a point can be reached where the current solution becomes far from a feasible case. In order to deal this problem, a restore operator is implemented to return back feasible solution in two sates. In the first stage, this procedure attempts to allocate students to the allowable stops in other routes, when possible. This process continues until the whole eligible stops as well as routes are investigated. At the end of the first stage, if there is no violated route, a feasible solution is found. If it is not the case, the solution of the first step enters the second step, called split procedure.

### 4.4. Allocation Heuristic Sub-Problem

We consider student allocation in two positions: first, in the constructive phase when the stops are selected, the routes are built and the feasibility of the solution in term of student allocation is preserved; second, during the second level of the oscillating strategy stage where the current solution is re-optimized. Before applying remove and redistribution operators, feasibility check or student re-allocation must take place. Due to the nature of our problem, it is not necessary to check feasibility during the first stage of OLS.

The heuristic procedure used to assign students to stops during the constructive heuristic works as follows. Each student needs to be allocated to only one potential stop based on stop reachability. To support this decision, a list of stops from which the walking distance is not greater than the maximum walking distance is generated in a preliminary stage. The list of students is then sorted following an increasing order of the number of allowable stops. The heuristic initially starts allocating students from the top of the list to the first available stop that is in a reachable distance. The list of students is explored sequentially. This simple rule allows critical students who have only one or few allowable stops to be assigned first. The drawback of this procedure is that sometimes, during the heuristic, no stop with available capacity is left for some students in the list. If this happens, a repair procedure is put in place to allocate unassigned students. To this end, the congested stops, having no remaining capacity where to host an unassigned student, should be identified.

In order to make room in these congested stops, a list is developed for those students who have already been assigned to these stops, and, consequently, the highest number of non-congested alternative stops where students can theoretically be reallocated is determined. Then, a student is randomly selected from this list and reallocated to another alternative

non-congested stop. Afterward, a room is created to allocate the unassigned student in the congested stop. The heuristic thus continues until the list of unassigned students is emptied.

The student reallocation mechanism employed before applying remove and redistribution operators is a slightly different procedure than the one used to allocate students in the initial solution (constructive phase).

The remove operator tries to remove the stop from the solution, thereby decreasing the total traveling distance. For this reason, before applying the move, a student allocation sub-problem must be solved. If the latter produces a feasible solution, the move is performed.

More specifically, in the remove operator, once a stop is selected to be removed, the students who have ben assigned to it need to be reassigned to a new stop or distributed among the existing routes by solving an allocation sub-problem. The allocation sub-problem for remove operator works as follows: first, the stop to be removed is chosen. A list of students is created so as to be assigned to this stop. The students in the list are successively sorted according to an increasing number of stops among other routes where they can walk to. With these students, the number of stops where they might be potentially reallocated is specified. Then, a student is selected from the top of the list and reallocated to another alternative stop included in another route, without violating the capacity constraint of the routes. This procedure terminates when all students in the list have been investigated. If all students could be successfully assigned to the potential stops, the remove operator is executed; otherwise, it is discarded.

Redistribution operator, as its name suggests, is meant to establish a desirable balance between routes. More precisely, incorporating the set of intra and inter-route operators during the improvement phase leads to the unbalanced distribution of students amongst routes. In other words, some routes might contain a large number of students, and some others could have only a few students. To make a balance in the correct solution, a specific redistribution operator will re-optimize the current capacity distribution among routes. In practice, this heuristic is basically constructed to minimize the corresponding deviation of the loading value of the routes via distributing students among routes in a proportionate manner. It is assumed that the total number of students and occupied capacity of each route are represented with $n_s$ and $c_r$, respectively, while $A_{lr}$ denotes the average loads on all generated routes. Creating a list that would contain routes ranked in decreasing order of occupied capacity is the starting point of the proposed method. This list includes only those routes which feature an occupied capacity greater than $A_{lr}$,. Then, we preced by performing the following step until all routes in the list have been considered. In the beginning, a route is selected from the top of the list for all students. Some students are transferred to another route by considering student allocation sub-problem. This method is repeated and then a student allocation sub-problem is solved as long as $c_r - A_{lr} \leq 0$. If the selected routes do not find an available position for possible students, the next route in the list will be considered. Note that this procedure does not improve our results directly and only affects the desirable distribution of students in the current solution.

### 4.5. Metaheuristic Structures and Diversification Strategies

After the intensification phase, a local optimum is found, and the metaheuristic attempts to escape from this point by destroying part of the current solution. This would increase the chance to reach a global optimum by starting from the perturbed solution and adopting local search. In fact, the primary goal of diversification mechanisms is to provide a good starting point for the intensification stage.

In this paper, two variants of diversification mechanisms are used so that in the case of a deficiency in one operator escaping from the local optima, other operators may result in more efficient outcomes. Therefore, two kinds of diversifications are employed: Repair –Destroy and Double Swap which are applied by a random selection mechanism.

#### 4.5.1 Destroy and Repair Method

This perturbation mechanism, called destroy and repair method, tries to iteratively explores different parts of the solution space by removing stops and reinserting them in other locations. This mechanism takes into account two parameters as input: the percentage of routes to be destroyed (denoted by $\varepsilon$ ) and is the total number of routes contained in the current solution ( $k$ ). During the perturbation heuristic, a destroy-and-repair operator is employed.

In the destroy phase, a random route is selected from the alternative solution; all stops are removed and added to the list of non-visited stops, called U. This step is repeated $\varepsilon * k$ times. During the repair phase, a new solution (x) is generated by creating new routes that include all non-visited stops existing in the U list. These new routes are generated by using the nearest neighbourhood with a greedy randomized selection mechanism described before. In other words, the next stop in the current route is randomly chosen from a restricted candidate list containing the first $\alpha$ closest non-visited stops. This process continues until the list of the non-visited stops is emptied.

#### 4.5.2 Double swap

The method consists of repeating two times a swap movement implemented between different routes which are randomly selected. To do so, at each time, the swap operator exchanges the positions of two stops between different routes. Since double swap operator works on the basis of exchanging the position of two stops between different routes, the results of the solution may lead to an infeasible case. In other words, during double swap operation it may happen that those stops with a large number of students to be exchanged between routes produce an infeasible solution. This can reach the point where neither the constructive nor the diversification heuristics would guarantee providing a feasible solution, something which is expected from the oscillating local search heuristic (see Section 4.3)

## 5. Computational experiments

This section describes the experiments used to test both metaheuristics (N-ILS and I-ILS). Since SBRP has been studied before, our computational experimental is conducted based on the two data sets. The first data set (data set I), containing a set of 104 instances taken from the available benchmark instances proposed by Schittekat (2013), has been employed and grouped in three sets of small, medium, and large instances: set S having instances with 5 to 10 stops, set M containing 20 stops, and set L having 40 up to 80 stops. All these instances present a variety of features such as the number of stops, number of students, bus capacity, and maximum walking distance of students from the selected stops. These instances can be downloaded from http://antor.uantwerpen.be/metaheuristic-for-the-school-bus-routing-problem-with-bus-stop-selection/. The instance names present SSSS-s _u _c _w_ that means a number of stops, students, a bus capacity, and a walking distance of students, respectively.

The second data set (data set II) refers to newly generated data and contains 30 instances. The problem size for this data set includes the number of stops ranging from 6 to 8, the number of student ranging from 30 to 120, and the number of walking distance ranging from 5 to 40. Like data set I, only one school is considered for these instances. To generate data set II, 5 parameters need to be defined per instance in the first stage, including $n_p$ (number of stops), $n_s$ (number of students), $x_d, y_d$ ( x and y coordinates of the school), and $w_{\max}$ (maximum walking distance for each student to reach a bus stop). The x and y coordinates of the school are $(75,75)$ . All instances are randomly generated in the Euclidean square between $(0,0)$ and $(x_{\max}, y_{\max})$ . The values of $(x_{\max}, y_{\max})$ are set to (150, 150). The procedure of creating data set II is like that proposed by Schittekat et al. (2013). Experimental analysis is conducted in three stages. In the first stage, the key components of each metaheuristic, presented in Section 4, are investigated and tuned in such a way that each metaheuristic generates, on average, the best solutions (instances are drawn from data set I). In the second stage, once the best parameter setting for each of the metaheuristics is obtained, we will compare our solution (I-ILS and N-ILS) methods to the best-known solution taken from the literature (i.e., the best solutions found by the metaheuristic as published in Schittekat et al. (2013)). All instances used for the second stage are taken from data set I. In the third stage, we have used 30 instances taken from the dataset II to compare our solution method. As no solution for data set II was referred to in Schittekat et al., (2013), we attempt to compare our solution methods with the exact method found by GAMS software. The first stage of the experiments is presented in sections 5.1 and 5.2, while the second and the third stages are presented in section 5.3.

### 5.1. Parameter configuration

Both metaheuristics, presented in section 4, have several parameters that are essential to be set. The main idea underlying this analysis is to identify the best components that significantly influence both the solution quality and the computing time.

In order to calibrate both metaheuristics, a subset of instances has been used in a full factorial experiment, combining all parameters setting. The testing set was made up by 14 instances (8 instances from set S, 4 instances from set M, and 2 instances from set L). Each of these instances is run five times. A brief description of the parameters that have been tuned as well as the tested values is presented in Table 2.

As mentioned before, two performance measures were considered: the average solution cost and the average computation time. It is worth mentioning that the maximum number of iterations is not included in the list of parameters which have been analyzed, because a large number of iterations will generate better solutions in a longer computational time. Hence, in this stage, the number of iterations has been fixed to 300. In the second phase, we performed some tests to assess the convergence of the algorithm (see section 5.2). The multi-way ANOVA method is used to analyze these runs through SAS software indicating the P-value of F-test (Table 3). The P-value indicates significant effects of the associated parameter on both the solution cost and the computation time. In this context, significant parameters are highlighted in bold. In Table 4, the best parameters setting for both N-ILS and I-ILS are shown.

**Table 2.** Heuristic parameters and the tested levels

| Parameters | Description | Value | No of levels |
|---|---|---|---|
| Repetition | Number of iterations | 300 | 1 |
| N1= Relocate- within | Relocate intra-route operator | On, Off | 2 |
| N2= Relocate - between | Relocate inter-route operator | On, Off | 2 |
| N3=Exchange - within | Exchange intra-route operator | On, Off | 2 |
| N4= Exchange-between | Exchange intra-route operator | On, Off | 2 |
| N5= Two -opt- within | Two -opt intra-route operator | On, Off | 2 |
| N6= Two -opt -between | Two -opt intra-route operator | On, Off | 2 |
| N7=Redistribution | Student transfer operator | On, Off | 2 |
| N8=Remove | Remove operator | On, Off | 2 |
| $\varepsilon$ | Maximum percentage number of routes in best solution found so far to be destroyed | 10%,15%,20% 25%,30%, 40%, 50% | 7 |
| $\lambda_0$ | Initial penalty | 0, 1, 2, 5, 10, 100 | 6 |
| $\beta$ | The multiplicative factor employed to increase the penalty | 1,2,5,10 | 4 |
| $\alpha$ | Size of the restricted candidate list | 1, 2, 3, 4, 5 | 5 |

**Table 3.** P-Values of F-tests

| Parameters | Average solution cost | CPU time |
|---|---|---|
| N1= Relocate- within | **<0.05** | **<0.05** |
| N2= Relocate - between | **<0.05** | **<0.05** |
| N3=Exchange –within | **<0.05** | **<0.05** |
| N4= Exchange-between | **<0.05** | **<0.05** |
| N5= Two-opt - within | **<0.05** | **<0.05** |
| N6= Two-opt –between | **<0.05** | **<0.05** |
| N7=Redistribution | **<0.05** | **<0.05** |
| N8=Remove | **<0.05** | **<0.05** |
| $\varepsilon$ | **<0.05** | **<0.05** |
| $\lambda_0$ | **<0.05** | **<0.05** |
| $\beta$ | 0.3765 | 0.1247 |
| $\alpha$ | 0.6792 | 0.0654 |
| $\lambda_0 * \varepsilon$ | **<0.05** | **<0.05** |

**Table 4.** Best parameter settings for both metaheuristics

| Parameters | I-ILS | N-ILS |
|---|---|---|
| N1=Remove_ Insert -within | On | On |
| N2=Remove _Insert -between | On | On |
| N3=Exchange -within | On | On |
| N4= Exchange-between | On | On |
| N5=2-opt -within | On | On |
| N6=2-opt -between | On | On |
| N7=Redistribution | On | On |
| N8=Remove | On | On |
| $\varepsilon$ | 25% | 30% |
| $\lambda_0$ | 2 | 1 |
| $\beta$ | 5 | 2 |
| $\alpha$ | 3 | 2 |

Based on P-value, it seems that all local search operators, percentage number of routes to be destroyed $(\varepsilon)$, as well as the value of initial penalty $(\lambda_0)$ are important parameters that influence both the solution quality and the computing time. Unexpectedly, the computing time and quality of solution are influenced by neither the multiplicative factor $(\beta)$ nor the size of restricted candidate list $(\alpha)$.

It means that the performance of metaheuristics is insensitive with respect to the values of $\beta$ and $(\alpha)$. Additionally, examination of the results of the ANOVA Table reveals that the interaction of parameters $\varepsilon$ and $\lambda_0$ have a significant impact on both the quality of solution and computing time. Thus, the capability of the algorithm to transfer between the feasible and infeasible portions of the solution space and also the percentage number of routes to be destroyed are key features of our metaheuristics. What matters is the capability to search into the infeasible portion of solution space, not whether it accelerates the speed of transition from the infeasible to a feasible part of solution space or not.

### 5.2. Effect of the number of iterations on the performance of metaheuristics

In this section, we investigate the effect of the number of iterations on the performance of both metaheuristics. To this end, a trade-off between the computing time and quality of the solution has been studied. Two metaheuristics were considered for this analysis (I-ILS and N-ILS). Each metaheuristic was run 10 times on all instances employed in section 5.1, using 6 different number of iterations. For both metaheuristics, the number of iterations $(\phi)$ were equal to 100, 200, 400, 800, 1000, and 1200. The values employed for other parameters are taken from Table 4, featuring the best parameter obtained from the full factorial analysis. For each test set including 12 instances, we report :(1) the percentage gap between the best solutions calculated after 10 runs and the best-known solutions obtained from SBRP instances, which is related to the capacity of the solution to find a better solution; (2) percentage gap between the average cost of the solutions calculated after 10 runs and the best- known solutions obtained from SBRP instances, associated with

robustness analysis. The best-known solutions are taken from Schittekat et al. (2013). The aggregated results of both analyses are shown in Table 5. In Table 5, the second column represents the metaheuristic used. The third column illustrates the percentage of the best gap (% Best Gap), and the next column stands for the average percentage gap (% Avg.Gap). Ultimately, the final column indicates the total computing time required to solve 12 instances. As expected, a larger value of $\phi$ improves the quality of solutions and the robustness of both metaheuristics. However, as the value of $\phi$ increases from 100 to 200, the computational time augments by a factor of approximately 1.9. When the values of $\phi$ increase from 200 to 400, the computational time rises by a factor of 1.43 (for I-ILS metaheuristic) and 1.34 (for N-ILS metaheuristic). This is mainly due to the stopping criterion implemented in the OLS heuristic. This heuristic stops its execution because either exploration has been oriented around the same feasible local optimal solution (i.e., cycling around the same feasible solution), or that the algorithm explores an unpromising region of the solution space that contains a weak feasible solution). In general, we can conclude that a high quality of the initial solution makes OLS heuristic require a small number of iterations to find better results. This can reach the point where a shorter execution time is needed for OLS heuristic when the search explores the most promising region of the solution space (i.e., solution areas close to the good solution). The probability of this exploration rises when a remarkable number of iterations is executed.

**Table 5.** Computational results

| $\rho-\phi$ | Metaheuristic | %Best Gap | %Avg. Gap | Time ( in seconds) |
|---|---|---|---|---|
| 100 | N-ILS | 3.25 | 4.46 | 47 |
| | I-ILS | 3.50 | 4.68 | 51.97 |
| 200 | N-ILS | 2.53 | 3.88 | 90 |
| | I-ILS | 2.59 | 4.11 | 98 |
| 400 | N-ILS | 2.18 | 3.34 | 121 |
| | I-ILS | 2.23 | 3.58 | 140 |
| 800 | N-ILS | 1.98 | 3.11 | 189 |
| | I-ILS | 2.08 | 3.21 | 230.5 |
| 1000 | N-ILS | 1.96 | 3.08 | 245.78 |
| | I-ILS | 1.99 | 3.16 | 312.45 |
| 1200 | N-ILS | 1.91 | 3.01 | 387.89 |
| | I-ILS | 1.96 | 3.13 | 507.56 |

Unsurprisingly, if the number of iterations increases from 800 to 1000, the capacity of both metaheuristics for finding a better solution decreases. In other words, setting the value of iteration greater than 800 makes a little improvement in both % Best Gap and % Avg Gap while computing time increases. Hence, in order to make a trade-off between the computing time and quality of the solution, the number of iterations of 400 seems to be suitable to find the best solutions for the majority of instances and to be fixed for our problem. It can be seen in Table 5 that N-ILS outperforms I-ILS from all points of view. Moreover, regarding the computing time, N-ILS was on average 19% faster than I-ILS.

### 5.3. Comparison of the Metaheuristics On the Basis of Dataset I

Having determined the best parameters setting for each solution approach, we compared both metaheuristics in terms of solution quality and computation time by doing some tests on small, medium, and large instances. The solution approaches have been coded using Java language. To test and compare I-ILS and N-ILS, each metaheuristic was run 10 times on all instances and the results were compared with the best-known solutions found by Schittekat, (2013). As explained before, the experimental analysis was performed on 104 instances consisting of three subsets named, respectively, Set S, Set M, and Set L, where Set S contains 48 instances with a number of stops ranging from 5 to 10, Set M consists of 24 instances with 20 stops, and Set L is comprised of 32 instances with a number of stops ranging from 40 to 80. Also, four maximum values are considered for the walking distances: 5, 10, 20, and 40. In Appendix A, the results of the experiments are reported for each metaheuristic configuration using the parameter setting described in Section 5.1. The aggregated results for each subset of instances are summarized in Table 6 (a); each metaheuristic presents the percentage gap between the best solutions found after 10 runs and the best-known solutions averaged over all instances for each of the three sets of S, M, and L. Table 6 (b) displays the percentage gap between the average cost of solutions calculated after 10 runs and the best-known solutions averaged over all instances for each of the sets of S, M and L. This implies the robustness of each metaheuristic configuration. The best-known solutions are taken from Schittekat et al. (2013). In our approach, to have a fair comparison between competing algorithms, a fixed number of iterations (400) is set for both I-ILS and N-ILS to solve each instance. In Table 6 ((a) and (b)), the first column represents the metaheuristic used, while the next three columns classify the set of instances (from small to large). As it can be observed from Table 6 (a), for small and medium sets, the N-ILS gives on average lower percentage gaps from the best-known solutions, while I-ILS can generate smaller percentage gaps from the best-known solutions for the instances in set L. In terms of robustness, on average N-ILS excelled I-ILS for small, medium, and large sets).On average, N-ILS metaheuristic surpassed I-ILS approach with respect to quality of the solution (the best gap from the best-known solution for N-ILS equals 2.08% and for I-ILS it equals 2.17 %) and robustness (the average gap from the best-known solution

for N-ILS is 3.29 % and for I-ILS it is 3.48 %). With regard to computing time, Table 7 shows that I-ILS was totally 10 % slower than N-ILS.

**Table 6.** Results obtained by solving the instances contained in Sets S, M, and L

| Metaheuristic | SET S (percent) | SET M (percent) | SET L (percent) | Average (percent) |
|---|---|---|---|---|
| (a). Best gap from best-known solutions | | | | |
| N-ILS | 1.83 | 2.09 | 2.31 | 2.08 |
| I-ILS | 2.04 | 2.19 | 2.29 | 2.17 |
| (b). Robustness of each metaheuristic | | | | |
| N-ILS | 3.06 | 3.29 | 3.53 | 3.29 |
| I-ILS | 3.32 | 3.51 | 3.60 | 3.48 |

**Table 7.** Total average computing time of each metaheuristic (per second)

| Metaheuristic | SET S | SET M | Set L | All |
|---|---|---|---|---|
| N-ILS | 347.33 | 1799.57 | 53326.50 | 55474 |
| I-ILS | 369.22 | 1928.52 | 59064.56 | 61362 |

### 5.3.1. Comparison of the Metaheuristics on the Basis of Data Set II

The performance of the proposed heuristic for SBRP has been investigated by considering the newly generated instance (data set II). We solved all the 30 instances using I-ILS and N-ILS metaheuristics. Each metaheuristic was run 10 times on all 30 instances, and the results were compared with the exact solution found by GAMS/CPLEX solver (see Appendix B for further results, Table 10). The aggregated results of instances are summarized in Table 8. It has to be mentioned that the calculation methodology of %Best Gap is like that presented in section 5.3. Observing Table 8 given to the best gaps from the exact solution, one can see that N-ILS generated solutions whose average value is at least 0.29% lower than that of I-ILS heuristic.

Furthermore, N-ILS was able to find optimal solutions for 33% of the instances and obtained relatively small optimality gaps in all other cases, while this number was 7 (23%) for ILS.

**Table 8**. Results obtained by solving the instances contained in data set II

| Metaheuristic | Best gap from best-known solutions Percent | No. of optimal solution |
|---|---|---|
| N-ILS | 0.91 % | 10/30 |
| I-ILS | 1.20% | 7/30 |

## 6. Conclusion

In this paper, we proposed two metaheuristics to solve the school bus routing problem. The main constituent of the metaheuristics is represented by an oscillating local search that follows three features: during the search process, infeasible segments of the solution space are explored. The second feature is applicable when the value of the violation increases. In this way, restore operator tries to move the exploration back into the feasible portion of the solution space. The last feature is meant to employ a set of large neighbourhoods to better explore a large search space. Experiments were conducted on two data sets: data set I, which is associated with a set of 104 SBRP instances of benchmark introduced by Schittekat et al. (2013) and data set II, which is related to newly generated instances. For data set I, the formulation presented in Section 3 was solved accurately using CPLEX solver in GAMS software. Since only the 43 easiest instances can be solved in this way, two different metaheuristics were proposed to solve small, medium, and large problem instances in a reasonable time. For each metaheuristic, statistical analysis was carried out to obtain the best heuristic parameters setting. Having determined the best parameters setting for each solution approach, we compared both metaheuristics in terms of solution quality, robustness, and computing time in the case of all instances. The results of the computational experiments imply that N-ILS is very competitive in comparison to the best metaheuristic presented by Schittekat et al. (2013). In order to find the number of optimal solutions obtained by each metaheuristic, the results of the computational suggested that N- ILS metaheuristic excelled ILS metaheuristic, and 23 of its instances were matched with the best-known solution. Also, better solutions were found for 2 instances. I-ILS ranked second, as 19 of the instances were matched with the best-known solution. With regard to computing time, N-ILS metaheuristic finds near-optimal solutions in a limited time. As a key result, this research suggests that N-ILS metaheuristic is the best solution approach with respect to the quality of the solution, robustness, and computing time. Regarding data set II, it is noteworthy that both metaheuristics were compared with the 30 newly generated instances. Since no solution for data set II was considered in Schittekat et al., we used an exact approach to compare the results of both metaheuristics. Considering the average performance of the two metaheuristics, the results demonstrated the superiority of N-ILS for the respective instances. Besides, totally 10 instances

(33%) contained in data II could be solved to optimality using N-ILS metaheuristic. As for ILS metaheuristic, this number was 7 (23%).

As far as quality of the solution, robustness, as well as computing time are concerned, the main conclusion of the present study consists of the fact that N-ILS metaheuristic is the better solution approach than I-ILS in both datasets studied here.

Future research could be aimed at addressing additional objective functions, constraints, and features to model real-life situations. In particular, minimization of the walking distance of students from their houses to stops can also be dealt with as either a second objective function or embedded in the current one. A second research line consists of finding better ways to check student allocation feasibility before applying each improvement operator. To this end, further studies may be focused on reducing the computational complexity of the local search operators by, say, testing efficient data structures.

**Reference**

Angel R., Caudle W., Noonan R. and Whinston A. (1972). Computer-assisted school bus scheduling. *Management Science*, Vol. 18, pp. 279-288.

Baldacci R., Dell'amico M. and González J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, Vol. 55, pp. 1147-1162.

Bock A., Grant E., Kinsman J. and Sanità, L. (2011).The school bus problem on trees. *International Symposium on Algorithms and Computation*, Springer, pp. 10-19.

Bodin L. D. and Berman L. (1979). Routing and scheduling of school buses by computer. *Transportation Science*, Vol. 13, pp. 113-129.

Bowerman R., Hall B. and Calamai P. (1995). A multi-objective optimization approach to urban school bus routing: Formulation and solution method. Transportation Research Part A: Policy and Practice, Vol. 29, pp. 107-123.

Brandao J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, Vol. 173, pp. 540-555.

Campbell A. M. and Savelsbergh M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. Transportation science, Vol. 38, pp. 369-378.

Chapleau L., Ferland J. A. and Rousseau J. M. (1985). Clustering for routing in densely populated areas. European Journal of Operational Research, Vol. 20, pp. 48-57.

Desrosiers J., Commerciales É. D. H. É. Transports, U. D. M. C. D. R. S. L. and Opérationnelle, U. D. M. D. D. I. E. D. R. (1986a). TRANSCOL: A multi-period school bus routing and scheduling system, Montréal: Université de Montréal, Centre de recherche sur les tranports.

Desrosiers J., Transports U. D. M. C. D. R. S. L. and Opérationnelle, U. D. M. D. D. I. E. D. R. (1980). An Overview of School Busing System, Montréal: Université de Montréal, Centre de recherche sur les transports.

Dulac G., Ferland J. A. and Forgues P. A. (1980). School bus routes generator in urban surroundings. *Computers & Operations Research*, Vol. 7, pp. 199-213.

Gavish B. and Shlifer E. (1979). An approach for solving a class of transportation scheduling problems. European Journal of Operational Research, Vol. 3, pp. 122-134.

Hansen P. and Mladenović N. (1999). An introduction to variable neighborhood search. In Meta-heuristics, pp. 433-458). Springer US.

Hachicha M., Hodgson M. J., Laporte G. and SEMET F. (2000). Heuristics for the multi-vehicle covering tour problem. Computers & *Operations Research*, Vol. 27, pp. 29-42.

Kinable J., Spieksma F. and Vanden berghe G. (2014). School bus routing—a column generation approach. *International Transactions in Operational Research*, Vol. 21, pp. 453-478.

Lourenço H. R., Martin O. C. and Stützle T. (2010). Iterated local search: Framework and applications. Hand book of Metaheuristics. Springer.

Newton R. M. and Thomas W. H. (1974). Bus routing in a multi-school system. *Computers & Operations Research*, Vol.1, pp. 213-222.

Pacheco J., Caballero R., Laguna M. and Molina J. (2013). Bi-objective bus routing: an application to school buses in rural areas. Transportation Science, Vol.47, pp. 397-411.

Park J. and Kim B. I.( 2010). The school bus routing problem: A review. *European Journal of operational research*, Vol. 202, pp. 311-319.

Riera-ledesma J., Salazar-gonzález J. J. (2013). A column generation approach for a school bus routing problem with resource constraints. *Computers & Operations Research*, Vol. 40, pp. 566-583.

Schittekat P., Kinable J., Sörensen K., Sevaux M., Spieksma F. and Springael J.( 2013). A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, Vol. 229, pp. 518-528.

Subramanian A., Drummond L. M. D. A., Bentes C., Ochi L. S. and Farias R.( 2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research, Vol. 37, pp. 1899-1911.

Swersey A. J. and Ballard W.( 1984). Scheduling school buses. *Management Science*, Vol. 30, pp. 844-853.

Talarico L., Sörensen K.and Springael J. (2015). Metaheuristics for the risk-constrained cash-in-transit vehicle routing problem. *European Journal of Operational Research*, Vol. 244, pp. 457-470.

Toth P. and Vigo D.( 2002). Vehicle Routing Problem SIAM Monographs on Discrete Mathematics and Applications, Vol. 9. SIAM: Philadelphia, PA.

Toth P. and Vigo D.( 2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, Vol. 15, pp. 333-346.

Verderber W. J. (1974). Automated pupil transportation. *Computers & Operations Research*, Vol.1, pp. 235-245.