An Experimentation System for Bus Route Planning and Testing Metaheuristics Algorithms

Krzysztof Golonka, Leszek Koszalka, Iwona Pozniak-Koszalka, Andrzej Kasprzak
Dept. of Systems and Computer Networks, Wroclaw University of Technology
Wroclaw, Poland
e-mail: leszek.koszalka@pwr.wroc.pl

***Abstract*—In this paper, we present an experimentation system for school bus route planning and testing various algorithms to solve such an optimization problem. It is a crucial social issue that concerns faster and more comfortable transport. Moreover, the route optimization allows decreasing the ticket price by maximizing the profit of the provider. Since the problem belongs to hard optimization problems, thus, we considered four meta-heuristic algorithms: three adapted, including Tabu Search, Simulated Annealing, Genetic Algorithm, and algorithm invented by the authors called Constructor. The efficiency of algorithms was tested and compared to that found by Complete Overview using the designed and implemented experimentation system. The investigations made on various problem instances, allowed to emerge the most efficient algorithm.**

Keywords-experimentation system; metaheuristic algorithms; route planning; optimization; efficiency

I. INTRODUCTION

Since banking crisis from 2008 many companies were forced to cut expenses and look for more savings. Moreover, nowadays a lot of pressure is put on being green – environmentally - friendly, especially when it comes to industry or transport e.g. [1]. A lot of effort must be put in analysis and planning process to come across these challenges. This paper focuses on optimization of a school bus route and proposes the direction of searching optimal solutions. The main goal is to find the most profitable route (e.g. the shortest path).

This optimization belongs to non-polynomial problem and has a huge solution space, meaning we can not find the best solution in polynomial time. For small instances it is easy to search through the whole solution space but when instance begins to grow, the required time may become unacceptable. The only one reasonable way to solve this is to use meta-heuristic algorithms that were invented to struggle with such problems. An idea to consider such algorithms based on artificial intelligence like Tabu Search (e.g. described in [2], [3], and [4]), Simulated Annealing (e.g. explained in [5], [6], and [7]) and Genetic Algorithm (e.g. illustrated in [8], [9], and [10]) seems to be promising.

We decided to adapt all three mentioned ideas for implementing algorithms to find an efficient solution to bus route problem. In addition, we tried to invent on our own a new algorithm and we created the algorithm called Constructor which is described in this paper. To determine the shortest path from the starting point to the ending point through all possible bus routes we implemented Bellman-Ford Algorithm (e.g. [11]).

Moreover, we implemented Complete Overview (CO) algorithm to be able to calculate differences between maximum and optimum found using meta-heuristics (unfortunately, CO may be used only for instances smaller than 20 bus routes because of its complexity and the required time). Finally, we had to introduce an evaluating function as the measure of efficiency for the considered algorithms.

To assess algorithms' efficiency we implemented an experimentation system that allows user to perform series of tests along with multistage experiment design ideas presented in [12].

The rest of paper is organized as follows: Section II defines the problem to be solved. Section III describes the considered algorithms and their roles in optimization process. Section IV shortly presents the implemented experimentation system. The results of the research appear in Section V. Section VI provides some final remarks and conclusion.

II. TYPE STYLE AND FONTS

There is given a certain urban area (Fig. 1), that consists of links and Bus Stops (BSs). Lines represent links, numbers next to them represent their lengths and red dots symbolize Bus Stops. The beginning of the route is marked by a green rectangle, but blue ending point (rectangle) represents the location of the school.

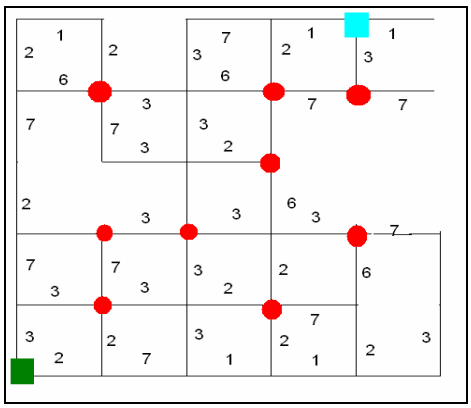

Figure 1. An example of an instance of bus route planning problem.

It is necessary to determine the most profitable route of a school bus to maximize profits of a bus provider.

 ISBN:978-1-61208-114-4

This means that the route may consist only BSs at which the number of students (pupils) waiting for a bus is sufficient not to make loses. Once the route is planned the bus may omit some BSs which are not on this specified route.

The basis for making decision on which BS should stop is the observation of statistics that deliver the number of students (pupils) waiting at a BS on a given time. These values are represented by a matrix, in which each row corresponds to the next hour in bus transit and the columns show the number of pupils (Table 1). The problem parameters are listed in Table 2.

TABLE 1. Students (pupils) statistics - an example.

Bus Stop / Time	#1	#2	#3	#4
8:00am	5	5	16	9
8:45am	6	1	3	11
9:30am	10	22	4	9
10:15am	0	2	0	0

TABLE 2. Input parameters of the problem.

Sign	Parameter
SBS	Set of potential BSs
(x_i,y_i)	BS_i coordinates
$L_{i,j}$	Link length between i and *j* BS
$P_{k,j}$	Pupils at k BS at j transit
T	Ticket price
DC	Driver's cost
BC	Bus exploitation cost
J	Number of transits

A. Basic Terms

Bus Stop (BS) is a point on a map where pupils wait for a bus to school. Each BS has its coordinates *x* and *y* on a map.

$$BS = [x, y] \tag{1}$$

Set of Potential BSs (SBS) is a collection of all BSs on a map, where SBS(*i*) may be defined by (2).

$$SBS(i) = \begin{bmatrix} BS_1 \\ BS_2 \\ \text{M} \\ BS_i \end{bmatrix} = \begin{bmatrix} x_1, y_1 \\ x_2, y_2 \\ \text{M} \\ x_i, y_i \end{bmatrix} \tag{2}$$

Link (L) is a matrix defined by (3) that describes lengths of links between BS_i and BS_j. Some BSs may not be directly linked to others but each BS must have at least one link.

$$L(i, j) = \begin{bmatrix} l_{1,1} & \Lambda & l_{1,j} \\ \text{M} & \text{O} & \text{M} \\ l_{i,1} & \Lambda & l_{i,j} \end{bmatrix} \tag{3}$$

$l_{i,j}$ is defined as:

$$l_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4}$$

Route (R) is a path consisting starting point, ending point and going through BSs chosen from SBS. A Route may contain smaller amount of BSs than SBS.

$$R = SBS(k) \qquad \text{where } k \le i \tag{5}$$

Ticket Price (T) informs how much each pupil must pay for taking a bus.

Driver's Cost (DC) equals money paid to a driver for driving one unit of a length of a Route.

Bus Exploitation Cost (BC) equals expenses for fuel used by a bus after driving one unit of a length of a Route.

Number of Transits (J) says how many times the bus is going a route per a day.

Route Length (RL) is a length of a shortest path.

B. Evaluating Function

The evaluating function introduced by us is called the Balance (6) interpreted as a daily balance – obtained after one day of work. If *Q(R)* is less than 0 that the provider gets loses, if greater than 0 that the provider gets profits.

$$Q(R) = \sum_{j=1}^{J} \left(\sum_{k=1}^{K} P_{k,j} * T - RL * (DC + BC) \right) \tag{6}$$

Moreover, to make sure that the bus will not go from starting point right to the destination point, we introduce a constraint. The constraint describes the minimal percentage number of all pupils from the statistics (Table 1) that should be delivered to the school (7).

$$100\ \% * \frac{\sum_{j=1}^{J} \sum_{k=1}^{K} P_{k,j}}{\sum_{j=1}^{J} \sum_{i=1}^{I} P_{k,i}} \ge P_{\%} \tag{7}$$

III. THE ALGORITHMS

A. Basic Ideas

In the paper, we consider four meta-heuristic algorithms as the main algorithms, including three known algorithms (but specially adopted) : TS - Tabu Search, SA - Simulated Annealing, GA - Genetic Algorithm, and the Constructor (originally proposed by the authors of the paper). The first two of them are described, e.g. in [13]. Our adaptations of GA as well as the Constructor are explained below. The

Route Length is calculated by using Bellman-Ford algorithm ([11]). TS, SA, and GA perform calculations for a certain number of iterations, processing on a solution and its neighborhood. A solution is a RL and the neighborhood is a set of Routes with one different BS, without one BS or with additional one BS.

The performance of each meta-heuristic algorithm is affected by a few factors such as an instance parameters (the size of SBS) and algorithms inner parameters.

B. *Simulated Annealing (SA)*

In any iteration the solution is replaced by a new one randomly chosen from the neighborhood if the new one is better. If the new solution is worse it has 50% of chances to replace the previous one. In this particular implementation we do not have such thing as temperature that changes the probability of replacing solutions. Here probability is constant. This is the only one difference between our SA and the one described in [7].

C. *Tabu Search (TS)*

The implemented Tabu Search is more complex algorithm than SA because of applying the searching procedure through the whole neighborhood of a recent solution and choosing the best one unlike SA. Moreover, TS is more resistant to loops thanks to the taboo list. The best found solution may replace the previous one only if it differs from all the records in a taboo list more than of a certain percentage value. The length of the taboo list is limited and when the list is full, the old records are overwritten.

D. *Genetic Algorithm (GA)*

This algorithm is based on evolutionary mechanisms. The main idea is to create a population of a constant size and observe its evolution meanwhile registering the best ones. The most interesting here is a cross-over process. It requires two individuals and eventually gives two children. DNA chain is represented in this situation as a single solution. The crossover is described below on an example:

parent no1: 1001|1101
parent no2: 1100|0111
child no1: 1001|0111
child no2: 1100|0111

All the solutions have a chance to hand over gens but the higher the Price function value of the solution, the higher possibility of being picked as a parent. Moreover, each child may mutate with probability 50% that leads to changing only one randomly picked chromosome. As the population size is constant, the newborns must replace the old solutions regardless of their breeding history.

The algorithm implemented in our specified problem may be described as follows:

Step 0. Initial population is picked randomly.
Step 1. Pick parents randomly from existing population.
Step 2. Perform breeding process.
Step 3. Choose solutions to extinct.
Step 4. Add children solutions to the rest of solutions in existing population.
Step 5. Check whether the evaluating function for each solution in current population is not the best global optimum from already explored solution space.
Step 6. Go back to step 1 if stop condition (defined by parameter Iterations) is not fulfilled.

E. *The Constructor*

The algorithm named Constructor was invented by us – it is based on the decomposition concept - we assumed that splitting a big instance to smaller ones, solving them separately and joining all together may be effective.

At the beginning the Constructor splits the whole instance - containing two BSs, the beginning BS and the last BS. Next, it searches through a neighborhood of each small instance and modify them. After that, the algorithm combines in pairs small instances integrating them as obtaining the initially considered instance.

Step 1. For specified short route that consists of beginning BS, ending BS and temporary amount of BSs find new possible routs picked from the short route's neighborhood.
Step 2. Check the optimization function for solutions, which include new route. Find the best neighbor.
Step 3. Modify current solution by including that best neighbor as a part of the solution.
Step 4. Pick next temporary amount of BSs and go back to step 1, unless all of BSs has already been picked.
Step 5. Double the temporary amount of BSs and go back to step 1, unless the temporary amount of BSs exceeded amount of all BSs.

IV. EXPERIMENTATION SYSTEM

The application was designed, mainly in order to visualize the tested algorithms. It was created using Visual Studio 2008. The implementation language was C#. In Fig. 2, a screenshot of the application window is shown.

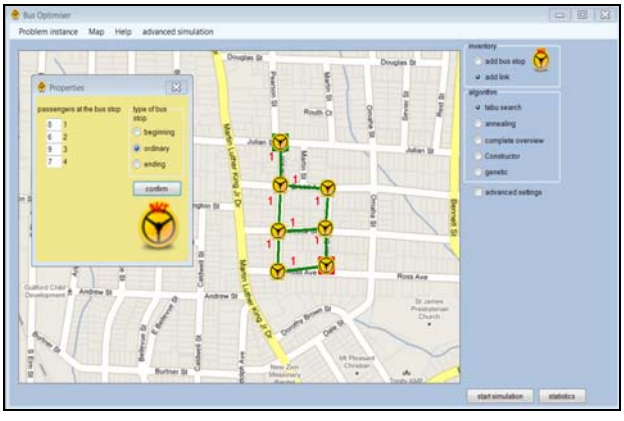

Fig. 2. Application window.

In the beginning the user defines an instance of problem by putting BSs on the map and creating links between them.

 ISBN:978-1-61208-114-4

Next, the beginning and the ending points of the potential route are precised and pupil statistics (by clicking and selecting properties) is determined. Finally, the user selects the considered algorithm and fixes its parameters. After clicking on "start simulation" button the application is searching for an optimal solution. There is a possibility to see an animation of a transit, and some statistics presented also on plots (e.g. served BSs, Balance, the percentage of served pupils) that allows observing current results.

V. INVESTIGATIONS.

A. *Calibrating Algorithms - Concept*

The first part of research refers to finding the values of the best inner parameters for two algorithms:

- Tabu Search,
- Genetic Algorithm

For each new set of parameters, a new simulation was made. The parameters of the problem for the considered instances are shown in Table 3. A single series of experiment meant that 10 different instances were tested; moreover, each instance was repeated 10 times. In Table 4 and Table 5 are the averages values over a given set of series of experiment.

TABLE 3. Parameters – set 1.

Bus Stops	15
Test iterations	10
Instances to test	10
[%] passengers	50
Driver's cost	1
Bus exploitation cost	1
Ticket price	3
Number of transits	4

B. *Adjusting Parameters for Genetic Algorithm*

Tests showed that increasing size of the population as well as increasing number of parents does not have a remarkable influence on improvement of solution (only about 5%). But the number of iterations has vast impact on results (Table 4).

TABLE 4. Results given by GA.

Test no.	GA	CO	ΔGA [%]	Population	Parents	Iterations
1	1360	1990	46,32	10	6	20
2	1384	1971	42,41	20	6	20
3	1292	1765	36,61	20	10	20
4	1243	1726	38,86	20	14	20
5	1215	1728	42,22	20	18	20
6	1256	1610	28,18	20	18	30
7	1307	1573	20,35	20	18	40
8	1259	1489	18,27	20	18	60

It may be observed, that for 60 iterations the results differ from the maximum just of approx. 18%. Because of slight differences in results between 60 and 40 iterations, the optimal value of this parameter was taken as equal to 40.

C. *Adjusting Parameters for Tabu Search*

The obtained results of research are shown in Table 5.

TABLE 5. Results given by TS.

Test no.	Tabu Search	CO	ΔTS [%]	Tabu list Length	% Precision	Iterations
1	1328	2255	69,80	4	10	20
2	1361	2074	52,39	4	5	20
3	1263	1571	24,39	4	2	20
4	1416	1592	12,43	4	1	20
5	1489	1595	7,12	4	1	30
6	1329	1352	1,73	4	1	50
7	1185	1213	2,36	8	1	50
8	1445	1505	4,15	16	1	50

As shown in Table 5, changing only two parameters make noticeable difference in precision and Tabu (taboo) list length.

Decreasing precision to 1% causes improvement of results in comparison to parameters from the first test. Apparently, smaller precision allows TS to make smaller but more frequent steps. This means, that TS explores larger solution space and it is obvious that in this situation the probability of encountering better result is higher. The vital parameter is the number of iterations - along with the same as in genetic algorithm property: the more iterations, the better solution.

D. *Comparison of Metaheuristic Algorithms*

The next part of research was to compare to CO all three other algorithms with the best inner parameters. Instances of parameters were the same as in previous part (Table 5) apart from minimal percentage passengers served (% passengers) which is variable in this test (Table 6).

TABLE 6. Parameters – set 2.

TS		SA		GA	
Iteration	50	Iteration	50	Iteration	40
Tabu length	4			Population	20
Precision	1			Parents	18

According to Fig. 3 the smallest inaccuracy was found for TS - from 10% to less than 1% for 90% passengers. The second efficient was GA - from more than 50% to about 10%. SA performed as third and the last place for Constructor. Constructor presents the biggest inaccuracy for 50% passengers (almost 250% inaccuracy) but its performance improves when constraint becomes more strict - 90 % passengers. Despite this surprising result, the rest of results leave a lot to wish, either.

 ISBN:978-1-61208-114-4

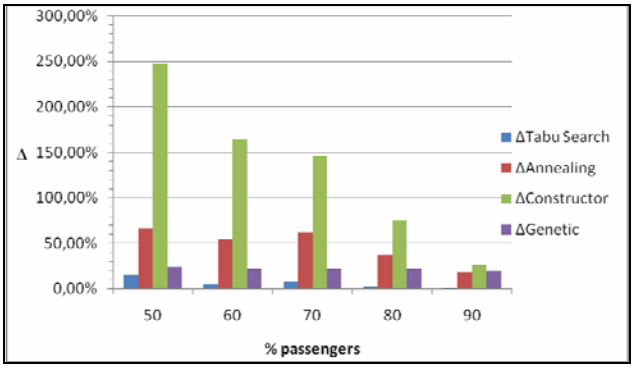

Figure 3. The average inaccuracy.

E. *Comparison – TS vs SA*

The influence of the number of iterations is shown in Fig. 4. The results of this series of experiments justify an observation (rather obvious) that the number of iterations has significant impact on the obtained results. The more iterations, the better results is given by the algorithm. The main useful observation is that TS gives better results than SA regardless of the number of iterations, thus TS algorithm may be recommended for searching the optimal route.

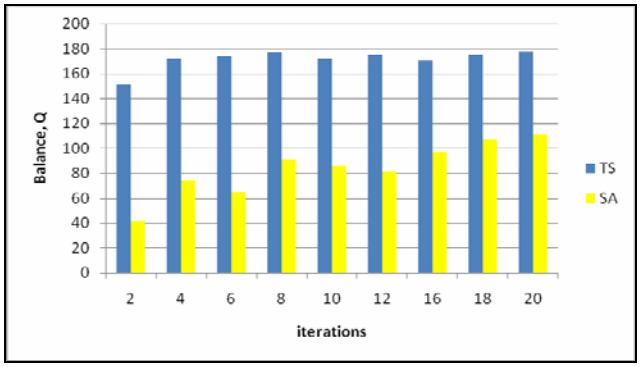

Figure 4. Comparison of alghoritms: TS vs SA.

F. *Comparison – TS vs GA*

The impact of the number of iterations is shown in Fig. 5.

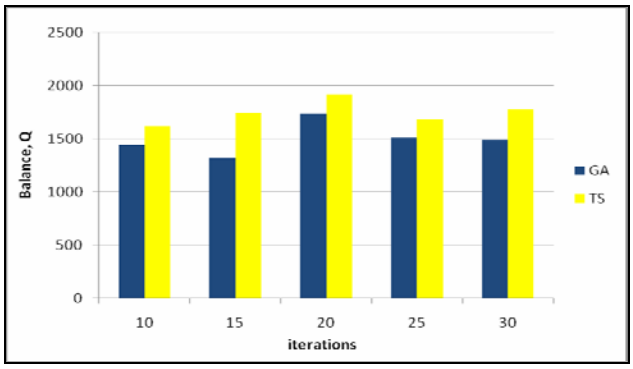

Figure 5. Comparison of algoritms: GA vs TS.

This complex experiment was designed in order to observe differences between the two metaheuristic algorithms: TS and GA, with their best inner parameters apart from parameter - *Iterations* which was a variable. The problem parameters used were such as specified in Table 3.

Similarly to the previous test, TS defeats competitor - GA regardless of iterations number. Although TS wins this competition, the genetic algorithm GA kept pace of TA and the results given by GA were not that bad as in SA case (see Sub-section *E*).

VI. FINAL REMARKS

To sum up, performed research justified the conclusion that TS algorithm gives much better results than SA regardless of defined advanced settings for searching the best solution. SA may give quite good results but much more iterations are needed. The only one algorithm that can compete with TA is GA but the average results of tests show that it would rather never come up with better results than TA. The Constructor algorithm turns out to be the worst and certain improvements are needed to make it somehow useful. Choosing the best algorithm is half the success, however, setting the most appropriate parameters of such algorithm is a vital issue.

According to research presented in this paper, the proposed and recommended algorithm for route planning is Tabu Search (TS).

REFERENCES

[1] J. Skladzien, “Ecological aspects of vehicle transport development”, Opole, 2008 /in Polish/.

[2] M. Gendreau, “An Introduction to Tabu Search”, Universite de Montreal, 2003.K. Elissa, “Title of paper if known,” unpublished.

[3] F. Glover, “Tabu Search – part I”, ORSA Journal on Computing, vol. 1, no. 3, 1997.

[4] F. Glover and G. A. Kochenberger, “Handbook of Metaheuristics”, Springer, Heidelberg, New York, 2002.

[5] V. Granville, M. Krivanek, and J. P. Rasson, “Simulated Annealing: a proof of convergence”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, 1994, pp. 652-656.

[6] S. Kirkpatrick, C. D. Gelatti, and M. P. Vecchi, “Optimization by Simulated Annealing”, Science, vol. 220, 1983, pp. 671-680.

[7] J. M. Laarhoven, H. Emile, and L. Aarts, ”Simulated Annealing: Theory and Applications”, Springer, Berlin, 1987.

[8] L. D. Davies, “Genetic Algorithms and Simulated Annealing”, Morgan Kaufmann Publ., 1987.

[9] H. Youssef and S. M. Sait, “Iterative Computer Algorithms with Applications in Engineering”, Washington., 1997.

[10] D. Ohia, L. Koszalka, and A. Kasprzak, “Evolutionary Algorithm for Congestion Problem in Computer Networks”, Springer, Lecture Notes in Artificial Intelligence, vol. 5711, 2009, pp. 113-122.

[11] A. Kasprzak, “Packet Switching Wide Area Networks”, WPWR, Wroclaw, 1997 /in Polish/.

[12] L. Koszalka, D. Lisowski and I. Pozniak-Koszalka, “Comparison of Allocation Algorithms with Multistage Experiments”, Lecture Notes in Computer Science, vol. 3984, Springer, 2006, pp. 58-67.

[13] P. Wroblewski, “Algorithms: data structure and programming technologies”, WNT, Warsaw, 2003 /in Polish/.

 ISBN:978-1-61208-114-4