Learning to optimize timetables for efficient transfers in public transportation systems

Renzo Massobrio[a,b,*], Sergio Nesmachnow[b,*], Jonathan Muraña[b], Bernabé Dorronsoro[a]

[a] *Universidad de Cádiz, Spain*
[b] *Universidad de la República, Uruguay*

Abstract

This article presents the application of a learning-based optimization method to solve the Bus Synchronization Problem, a relevant problem in public transportation systems. The problem consists in synchronizing the timetable of buses to optimize the transfer of passengers between bus lines. A new problem model is proposed, extending previous formulations in the literature, and solved using Virtual Savant. Virtual Savant is a novel soft computing method inspired by the Savant Syndrome that combines machine learning and optimization to solve complex real-world problems in a massively parallel way. The proposed methodology is validated and evaluated over a set of synthetic and realistic instances based on the public transportation system of Montevideo, Uruguay, and compared against a reference evolutionary algorithm and the current solution defined by the city authorities. The main results indicate that Virtual Savant is able to compute accurate solutions and outperform baseline results in eleven out of fifteen realistic instances. This is the first reported research applying Virtual Savant to a problem with a high synergy between its decision variables. The obtained results suggest that it is a suitable tool for solving this kind of optimization problems.

Keywords: Virtual Savant, Bus Synchronization Problem, machine learning,

*Corresponding author
Email addresses: renzom@fing.edu.uy (Renzo Massobrio), sergion@fing.edu.uy (Sergio Nesmachnow), jmurana@fing.edu.uy (Jonathan Muraña), bernabe.dorronsoro@uca.es (Bernabé Dorronsoro)

1. Introduction

Transportation systems play a major role in urban scenarios and are a key element of modern smart cities [1, 2]. In this context, public transportation allows reducing car dependency and is the most efficient and environmentally-friendly mean of transportation for citizens. However, the ability of public transportation systems to provide good quality of service depends on several relevant problems, including the proper planning of routes, timetabling, buses, and drivers [3].

Most public transportation systems operate along fixed routes and a predefined timetable, with services running "every x minutes" (the value of x is the *headway* defined for the line). Timetable synchronization has been recognized as one of the most difficult problems for public transportation planning and optimization [4]. For this reason, timetable synchronization problems have often been addressed intuitively, assuming that experienced operators are able to take proper decisions that account for appropriate quality of service.

Complex optimization problems, such as timetable synchronization, require algorithms that demand large computing resources [5]. The current growing interest in machine learning techniques comes at hand to deal with this problem. The fields of optimization and machine learning are closely related. However, the vast majority of research has explored one direction of this relationship, i.e., optimization applied to machine learning techniques (e.g., parameter optimization in machine learning models, feature selection problems) [6]. The inverse, i.e., applying machine learning to solve optimization problems, while explored [7, 8], still has plenty of room for contribution. In this direction, Virtual Savant (VS) is a novel paradigm that applies machine learning and parallel computing to address complex optimization problems [9]. VS is inspired by the Savant Syndrome, a mental condition where patients excel at certain abilities far above the average. In analogy to the Savant Syndrome, VS uses machine learning to

find patterns that allow solving the problem at hand, which are learned from a set of previously-solved instances of the problem. Due to its design, VS can be executed in massively-parallel computing architectures, significantly reducing execution times and effectively scaling in the problem instance.

In this line of work, this article proposes applying the soft computing learning-based optimization approach of VS to solve the Bus Synchronization Problem (BSP), a relevant problem in public transportation. BSP proposes finding a set of headways of bus lines to maximize the number of synchronized transfers in a given set of synchronization points, according to maximum waiting times the passengers are willing to accept for an appropriate quality of service of the public transportation system. The BSP is modeled as an NP-hard combinatorial optimization problem; thus, non-exact methods are needed to solve realistic instances in reasonable execution times. The main focus of the research is studying the capabilities of the learning/optimization approach of VS to solve a complex real-world problem with a high synergy between problem variables, since synchronizing two lines in a given synchronization point notably impacts on the ability of synchronizing those lines with other ones, in other synchronization points. This article presents the first reported research applying VS to a problem with this inherent complexity. VS was able to compute accurate results, outperforming the reference Evolutionary Algorithm (EA) in eleven out of fifteen realistic instances.

The main contributions of the reported research include: i) a new model for public transportation timetable synchronization, which allows bus transfers at any given bus stop; ii) the application of VS to solve the problem, by learning patterns from the results of the reference EA; and iii) the experimental evaluation of the proposed VS over a set of synthetic and realistic problem instances.

The article is organized as follows. Section 2 introduces the problem model and its mathematical formulation. Section 3 reviews related works about timetable synchronization to optimize transfers and learning/optimization approaches for solving combinatorial optimization problems. The proposed approach applying VS to the BSP is described in Section 4. The experimental evaluation over

synthetic and real problem instances is reported in Section 5. Finally, the conclusions and main lines for future work are formulated in Section 6.

2. The Bus Synchronization Problem

This section describes the BSP model and its mathematical formulation.

2.1. Overview of the problem model

Public transportation planners often prefer network topologies comprised of few, short, and densely interconnected bus lines. This design is good from an operational point of view, because it allows operating higher bus frequencies with smaller vehicle fleets. However, it requires passengers to make more bus transfers instead of traveling directly from origin to destination. Passengers dislike transfers: studies have shown that the perceived time when waiting for a bus or when walking between bus stops can be up to 2.5 times larger than the real time spent [10]. Consequently, reducing waiting times for passengers when transferring between buses is a desirable goal from the point of view of citizens.

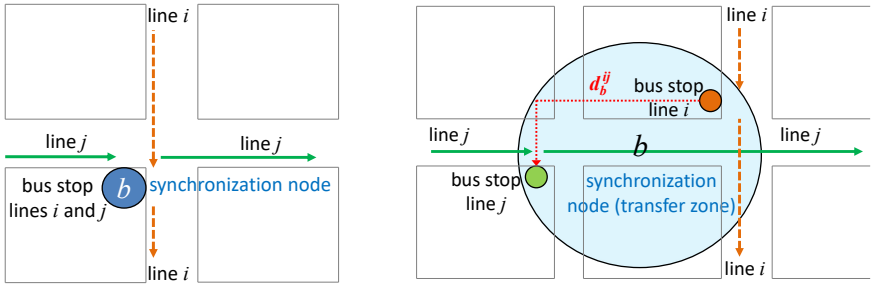
Two main approaches are applied for bus synchronization: i) timetable-based methods propose defining fixed departure times for trips of each bus line, and ii) headway management/operation methods allow changing the headway values to properly deal with dynamic situations that can affect the quality of service (e.g., traffic jams, forced detours, unexpected demands).

The considered problem model follows the timetable scheduling approach. The main goal is to determine headway values to maximize successful synchronizations for transfer trips. Possible headway values are within a range, defined considering both the quality of service offered to citizens (e.g., avoiding large times between bus trips for each line) and the economic viability of the bus service (i.e., avoiding very frequent departures, which in turn may cause bus bunching). Once defined, headways do not change during the bus service operation.

The BSP consists in finding the headways of each bus line in a public transportation system, which allow maximizing the number of synchronized bus

transfers. A bus transfer is considered synchronized when the waiting time experienced at the transfer bus stop does not exceed a given threshold, defined according to the maximum time passengers are willing to wait for the transfer.

Nowadays, some public transportation fare schemes do not impose limitations on the number of transfers for passengers. Furthermore, some transportation systems consider all bus stops in the network as possible transfer stops, providing flexibility to passengers when planning their routes. In this scenario, the corresponding synchronization problem is more complex, as the transfer may require the passenger to walk between bus stops. Thus, a new version of the BSP is proposed in this article, extending previous models in literature [11, 12]. In the proposed BSP model, nodes are not just fixed bus stops, but *transfer zones* that can include flexible combination of bus stops for lines. In addition, the synchronization criteria considers the walk time in the transfer zone. Figure 1 shows a graphical description of both previous and proposed models, considering two generic lines i (orange) and j (green). In the new model, the distance between the bus stops for lines i and j, $d_b^{i,j}$, is drawn in red.



(a) Previous BSP model (b) Extended BSP model

Figure 1: Comparison between the previous BSP model with predefined synchronization nodes (from the related literature) and the one proposed in this article, with flexible synchronization nodes (expanded to transfer zones)

2.2. Mathematical formulation

The BSP accounts for the main goals of modern transportation systems: providing a fast and reliable way for the movement of citizens while maintaining

reasonable operational costs. The problem model focuses on the quality of service provided to the users, i.e., a better traveling experience with limited waiting times when transferring between buses.

The mathematical formulation for BSP considers the following elements:

- A set of bus lines $I = \{i_1, \ldots, i_n\}$ and a set of lines $J(i)$ that may synchronize with line i (in a synchronization node, see next item). Buses that operate each line have a maximum capacity of C transfers, i.e., passengers that board the bus in the second leg of a transfer trip.

- A set of synchronization nodes $B = \{b_1, \ldots, b_m\}$. Each node $b \in B$ is a triplet $<i, j, d_b^{i,j}>$ indicating that lines i (inbound line) and j (outbound line) may synchronize in b, and that the bus stops for lines i and j are separated by a distance $d_b^{i,j}$.

- A planning period $[0, T]$, expressed in time units, and the number of trips of each line i, f_i, needed to fulfill the transfer demands for each line in that period.

- A *traveling time function* $TT : I \times B \to \mathbb{Z}$, where $\mathbb{Z}$ represents the set of integer numbers. $TT_b^i = TT(i, b)$ indicates the time for buses of line i to reach the synchronization node b (measured from the origin of the line). In general, this value depends on several features, including bus type, bus speed, traffic in roads, etc.

- A walking time function $WT : I \times I \times B \to \mathbb{N}$, $\mathbb{N}$ representing the set of natural numbers. $WT_b^{i,j} = WT(i, j, b) = d_b^{i,j}/ws$ indicates the time needed for a pedestrian to walk between bus stops at the synchronization node according to a given walking speed ws.

- A *demand function* $P : I \times I \times B \to \mathbb{Z}$. $P_b^{i,j} = P(i, j, b)$ indicates the number of passengers that transfer from line i to line j in synchronization node b, within the planning period.

- A maximum waiting time $W_b^{i,j}$ for each synchronization node, stating the maximum time that passengers are willing to wait for line j, after alighting from line i and walking to the stop of line j, in synchronization node b.

- A valid range of headways, which define the separation (measured in time units), between consecutive trips of the same line. The range of valid headways for bus line i is defined by an interval $[h^i, H^i]$, where values of h^i and H^i are usually enforced by public transportation administrators. Headway values can be either constant (useful on assumption of uniformly distributed transfer demands scenarios) or variable (to take into consideration specific variations of transfer demands within the planning period).

The BSP proposes finding appropriate values of headways for each bus line to guarantee the best synchronization for all lines with transfer demands in the planning period T. The mathematical model is formulated in Equation 1. The departure time of trip r of bus line i is represented by an integer variable X_r^i. A binary variable $Z_{r,s,b}^{i,j}$ indicates whether trip r of line i and trip s of line j are synchronized or not in node b. The proposed objective function weights synchronizations according to the number of passengers that transfer in the planning period, thus giving priority to synchronization nodes with larger transfer demands.

$$\text{maximize} \quad \sum_{b\in B}\sum_{i\in I}\sum_{j\in J(i)}\sum_{r=1}^{f_i}\sum_{s=1}^{f_j} Z_{r,s,b}^{i,j} \times \min(\frac{P_b^{i,j}}{f_i}, C) \tag{1a}$$

$$\text{subject to} \quad X_1^i \leq H^i \tag{1b}$$

$$T - H^i \leq X_{f_i}^i \leq T \tag{1c}$$

$$h^i \leq X_{r+1}^i - X_r^i \leq H^i \tag{1d}$$

$$(X_s^j + TT_b^j) - (X_r^i + TT_b^i) > WT_b^{i,j} \text{ if } Z_{r,s,b}^{i,j} = 1 \tag{1e}$$

$$(X_s^j + TT_b^j) - (X_r^i + TT_b^i) \leq W_b^{i,j} + WT_b^{i,j} \text{ if } Z_{r,s,b}^{i,j} = 1 \tag{1f}$$

$$X_r^i - X_{r-1}^i = X_s^i - X_{s-1}^i \forall r, s, r > 1, s > 1 \tag{1g}$$

$$X_r^i \in \{0, \ldots, T\}, Z_{r,s,b}^{i,j} \in \{0, 1\} \tag{1h}$$

The objective function (Equation 1a) proposes maximizing the number of synchronized transfers, weighted by the transfer demand for each trip in each synchronization node. The demand is split uniformly among the f_i trips of line i. This is a realistic assumption for planning periods where demand does not vary significantly. The number of passengers considered on each synchronization node is bounded by the transfer capacity for buses C.

Regarding constraints, Equation 1b assures that the first trip of each line starts before the upper bound of headways for that line. Equation 1c forces the last trip of each line to end before T; thus, it must start after T minus the upper bound of headways for that line. Equation 1d guarantees that the computed headways of each line are bounded to the range of valid headways for that line. Equations 1e and 1f state that trip r of line i and trip s of line j are synchronized at node b if passengers are able to transfer, considering the time needed to walk between the bus stops $WT_b^{i,j}$ and the maximum time that users are willing to wait for the second bus in the transfer to arrive, $W_b^{i,j}$. Equation 1g states that all headways for a given bus line are constant in the planning period. This constraint is consistent with the assumption of uniformly distributed transfer demands. This constraint can be relaxed to model a more complex variant of the BSP, allowing for bus headways of a given line to change within the planning period. This is a distinction of the proposed model, considering that related works usually split the problem in multiperiod timetabling planning, forcing to solve several chained problems with fixed headways [13]. Finally, Equation 1h defines the domain for the decision variables of the problem.

Figure 2 presents an example of the BSP for a scenario with three bus lines and three synchronization points (b_1, b_2, and b_3). Each synchronization point in this example is assumed to be on the same physical bus stop, so the corresponding walking time is zero, i.e., $WT_b^{i,j} = 0$. The maximum waiting time is set to two minutes for all synchronization points and bus lines, i.e., $W_b^{i,j} = 2$. In the example, the headway of line 1 is set to four minutes ($X_1^1 = 4$) and the headway of line 2 is set to three minutes ($X_1^2 = 3$). Considering the travel time of each bus line to reach synchronization node b_2 ($TT_2^1 = TT_2^2 = 12$), the first

trips of both lines are synchronized at node b_2, since the difference in arrival times is within the maximum waiting time threshold. The diagram shows how setting different values for the headway of bus line 3 affects the synchronization with the other two lines in the example. Setting the headway of bus line 3 to two minutes allows synchronizing the first trip of this line with the first trip of bus line 1 at node b_1, but no synchronization is possible between lines 2 and 3. In contrast, setting the headway of bus line 3 to five minutes results in two synchronized trips of lines 1 and 3 at node b_1 (i.e., the second trip of line 1 with the first trip of line 3 and the third trip of line 1 with the second trip of line 3) and also synchronizes the first trip of bus line 2 with the third trip of bus line 3 at node b_3. Furthermore, no synchronization is possible between the first trip of lines 1 and 3, i.e., $Z_{111}^{13} = 0 \ \forall X_1^3$, because of the headway value already selected for line 1. However, a different configuration setting $X_1^1 = 7$ would allow synchronizing the first trips of both lines. This setting could also affect the synchronization of other trips of the lines considered in the example since the initial conditions are changed.

The BSP is within the $\mathcal{NP}$-hard complexity problem class. The basic model proposed by Ceder et al. [14] is $\mathcal{NP}$-hard according to the formulations presented by Domschke [15] and Daduna and Voß [16] (Quadratic Semi-Assignment Problem formulation). Ibarra and Ríos [11] proved the NP-hardness of the BSP by applying a polynomial reduction from a variant of the Boolean satisfiability (3SAT) problem. The model considered by Ibarra and Ríos assumed a unitary demand function (i.e., $P_b(i,j) = 1 \forall b \in B$). The BSP formulation proposed in this article is also $\mathcal{NP}$-hard, since it extends previous models by considering transfer demands defined by a function P_b determined by the input data from a real Intelligent Transportation System.

3. Related Work

This section presents a review of related works about timetable synchronization to optimize transfers and the application of learning approaches for solving

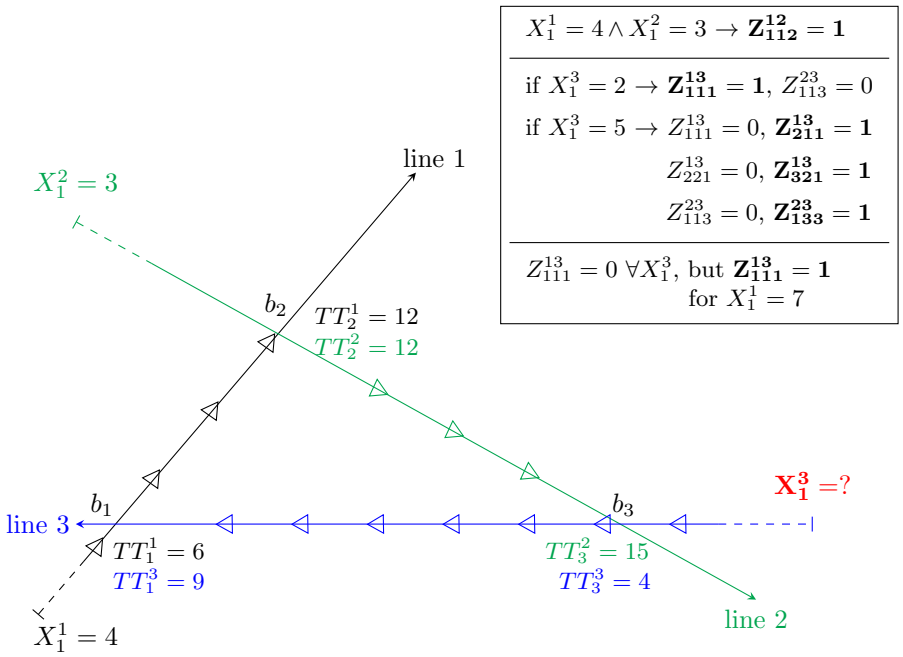


Figure 2: A simple example of the BSP for an scenario with three lines and three synchronization points, $WT_b^{i,j} = 0$, and $W_b^{i,j} = 2$.

combinatorial optimization problem.

3.1. Timetable Synchronization to Optimize Transfers

Daduna and Voß [16] studied the timetable synchronization problem on bus networks to minimize the waiting time of passengers. Different objectives were formulated, including a weighted sum considering transfers and the maximum waiting time. Simulated Annealing and Tabu Search were analyzed for simple versions of the problem over randomly generated examples based on the Berlin Underground network and three real-world cases from different German cities.

Ceder et al. [14] studied the problem of maximizing synchronization events between bus lines at shared stops. A greedy heuristic was proposed to solve the problem by defining custom timetables. The work focused on simultaneous bus arrivals and was evaluated on small instances with few nodes and few bus lines. Later, Fleurent et al. [17] considered a synchronization metric including

expert knowledge to minimize vehicle scheduling costs. A heuristic algorithm was proposed and evaluated on two small scenarios from Montréal, Canada.

Ibarra and Ríos [11] studied the BSP in Monterrey, Mexico. A flexible formulation of the problem was proposed, considering a time window between travel times to account for traffic congestion and other situations. A Multi-start Iterated Local Search (MILS) was evaluated over eight instances modeling the bus network in Monterrey (15 to 200 bus lines and 3 to 40 synchronization points). The method was able to compute efficient solutions for medium-size instances in less than one minute, but the gaps of MILS did not scale properly. Later, Ibarra et al. [13] solved the multiperiod bus synchronization problem, optimizing multiple trips of a given set of lines. MILS, Variable Neighborhood Search, and a simple population-based approach were proposed to solve the problem. All methods computed solutions with similar quality to an exact approach over synthetic instances with few synchronization points.

Timetable synchronization methods may be complemented with other techniques at the operation/management level to account for unexpected events, detours, and demand changes. In this regard, Hall et al. [18], Delgado et al. [19], and Van Oort et al. [20] studied techniques based on holding buses in stops (or transfer stations) to optimize the synchronization and the public transportation reliability.

We presented in [21] the BSP formulation outlined in Section 2, together with a highly specialized EA to solve it. Candidate solutions to the problem were modeled in the EA using integer vectors, each value representing the headway (in time units) of a bus line. The initial population of the EA was comprised of randomly generated solutions that satisfied the problem constraints as well as seed solutions corresponding to the current reality defined by the city authorities and from greedy approaches for the problem. Evolutionary operators included a traditional tournament selection, a two-point crossover operator, and a Gaussian mutation to modify the headway of the lines. The fitness function was defined considering the number of synchronized trips and their corresponding demands, according to the problem formulation. Problem instances based on real data

from the public transportation system in Montevideo, Uruguay were used for the experimental evaluation. Results showed that the proposed evolutionary approach computed accurate solutions, providing improvements of up to 13.33% in the fitness values and up to 24.20% in the waiting times, when compared to the current real timetable in Montevideo.

The EA proposed in the previous article [21] is used as the reference algorithm for the VS implementation for BSP presented in this article. The proposed EA was demonstrated to be highly accurate for a simpler version of BSP, as reported in our previous article [22]. In the addressed simpler version of BSP the headways are fixed, set as the real timetable values, and the control variables are the offset of the departing time of the first trip of each bus line. Under these assumptions, the BSP focuses on computing solutions with the same number of trips for each bus line than the real solution, thus maintaining the same monetary operation cost of the real timetable. However, this version restricts the domain of the problem, and the computed solutions are biased towards the real timetable configuration. The simpler version of BSP can be solved by an exact approach applying Integer Linear Programming (ILP). Thus, it allows evaluating the accuracy of the proposed EA by comparison to the optimal solution. The EA outperformed real timetables by 20% in average, and the gaps with the optimal solution were below 5%, demonstrating the accuracy of timetables computed by the proposed EA. This is a significant result, since the ILP method cannot be applied to compute exact solution for more complex versions of BSP, such as the one presented in Section 2, due to the curse of dimensionality associated to NP-hard optimization problems.

The model considered in our article includes additional features to the one proposed by Ibarra and Ríos [11]: scenarios where every pair of bus stops are possible transfer zones to synchronize and real transfer demand in each possible transfer zone.

3.2. Machine learning for optimization

The idea of applying machine learning techniques to solve optimization problems has been addressed in the literature, as reviewed in the recent article by Bengio et al. [23], which also outlined a methodology for further integrating both fields of research, and enumerated the main challenges in this regard. A brief review of works in this area of study is presented next.

Many real-world optimization problems follow the "predict then optimize" paradigm, which consists of sequentially predicting a set of unknown parameters or variables of the problem instance [24, 25]. Elmachtoub et al. [26] [unpublished work] proposed Smart "Predict, then Optimize" (SPO), a framework for problems that follow the predict-then-optimize paradigm. Instead of focusing on prediction errors, the authors proposed the SPO loss function, which measures the decision error induced by a wrong prediction. Experimental evaluation on synthetic instances of the shortest path and portfolio optimization problems showed that the proposed framework was able to outperform classic predict-then-optimize approaches. Later, Mandi et al. [27] extended the previous work to solve more realistic discrete optimization problems, applying strategies to relax the problem and warm-start of the learning process using previous solutions. Experimental results on small instances of the weighted knapsack and scheduling problems showed that the proposed approach outperformed a traditional two-stage approach that did not consider an optimization-directed loss.

Vlastelica et al. [7] proposed an end-to-end architecture integrating blackbox implementations of combinatorial solvers into neural networks and a novel interpolation technique to compute gradients of the piecewise constant function that characterizes the problem. Results for shortest path finding, Traveling Salesman Problem (TSP), and minimum cost perfect matching, showed high accuracy and generalization capabilities. A similar approach by Berthet et al. [28] rely on stochastically perturbing discrete optimizers with random noise and considering the perturbed solutions to the problem. The proposed strategy outperformed the one by Vlastelica et al. in terms of accuracy and cost ratio.

Many articles have focused on learning optimization solvers for problems

modeled using graphs. Vinyals et al. [8] introduced Pointer Networks (PNs), a model based on Recurrent Neural Networks In PNs, an encoder is used to parse an input graph and a decoder that outputs a probability distribution over its nodes, following an attention mechanism able to handle graphs of arbitrary size. The proposed model was evaluated when solving three discrete combinatorial optimization problems, computing competitive results in problem instances larger than those seen during the training phase. The approach was outperformed by Bello et al. [29] using reinforcement learning with PNs, using as a reward signal the inverse of the tour length. Solutions within 1% of the known optima for TSP (up to 100 cities) and 0/1 Knapsack Problem (up to 200 items) were computed. Hu et al. [30] extended this model to solve the 3D bin packing optimization problem, outperforming baseline heuristics for the problem.

Combinatorial optimization problems not modeled through graphs have also been addressed. Selsam et al. [31] proposed *NeuroSAT*, a solver for the boolean satisfiability problem (SAT) based on Message Passing Neural Networks. Two recent works have applied reinforcement learning to solve scheduling problems in a factory environment [32] and to schedule tasks in the cloud [33].

The paradigm used in this article to address the BSP was previously applied to the Next Release Problem [34, 35], an optimization problem arising in software engineering, and to the Heterogeneous Computing Scheduling Problem [9, 36] for planning tasks execution in large computing infrastructures.

The analysis of related works shows an increasing interest in applying machine learning to solve optimization problems. Some of the reviewed techniques involve iteratively executing the solver that is being learned. This constitutes a major obstacle when addressing large instances of complex optimization problems. The fact that VS can learn from a benchmark of previously-solved instances is a major advantage over these methods. Moreover, many approaches in the literature only learn from exact solvers. In this regard, the design of VS presents a major advantage over these approaches, since it can learn from instances solved by both exact and approximate algorithms.

4. Virtual Savant for the Bus Synchronization Problem

This section describes the proposed learning/optimization approach applying VS to solve the BSP.

4.1. Overview of VS

VS is a novel technique that aims to learn how to solve a given optimization problem [9] . It is inspired by the Savant Syndrome, a rare mental condition in which patients with significant mental disabilities develop certain abilities far above what would be considered average [37]. Patients with Savant Syndrome—known as *savants*—usually excel at a single specific activity, generally related to memory, rapid calculation, or artistic abilities. The main hypotheses state that savants learn through pattern recognition [38].

As an analogy to the Savant Syndrome, VS proposes using machine learning techniques to find patterns that allow solving the problem at hand. These patterns are learned using machine learning from a set of previously-solved instances of the problem. Solutions used for training are computed by one (or several) reference algorithm(s) for the problem. VS does not require knowing the code of the reference algorithms it learns from, in the same way that real-life savants are unaware of the underlying principles related to their skill. The training of VS involves partitioning the problem instance and, like savants, VS can derive global solutions by combining smaller pieces. Once the training phase is completed, VS can solve unseen and larger problem instances, without the need of any further retraining. Similarly to real-life savants, which are thought to have parallel processing capabilities [39], VS can also take advantage of multiple computing resources. Due to its design, each phase in VS can be executed in a massively-parallel fashion. Thanks to parallelism, VS can reduce execution times significantly, find better solutions due to an improved exploration of the search space, and effectively scale in the problem dimension.

4.2. Training of VS for BSP

The process of generating the training set for the BSP is outlined in Figure 3.

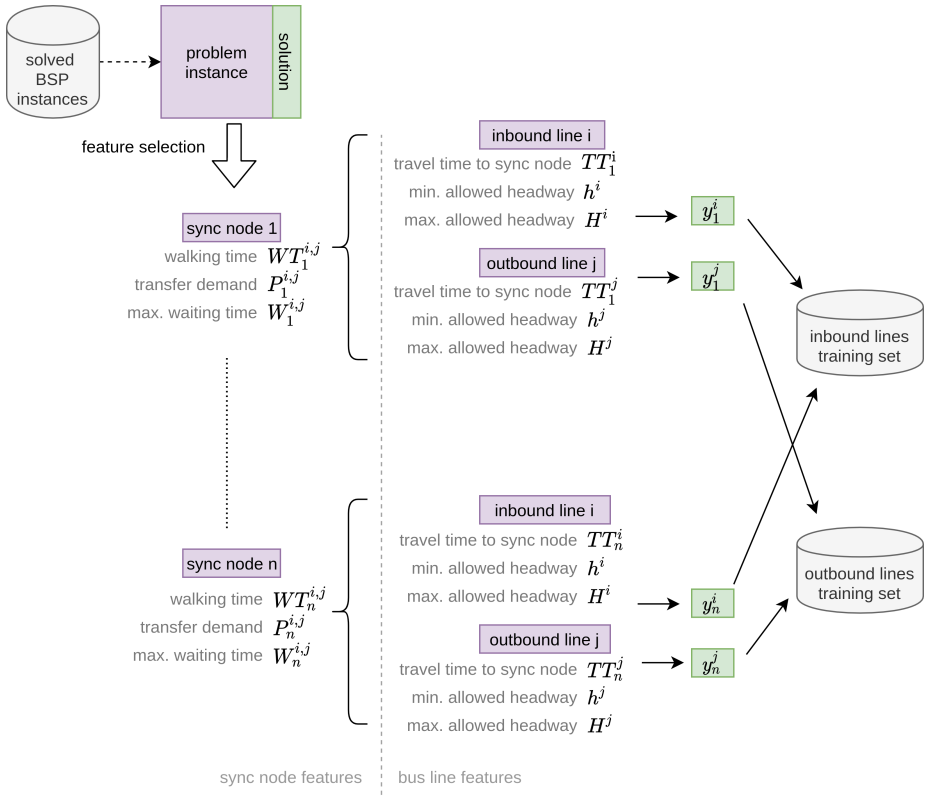


Figure 3: VS training set generation for the BSP.

VS learns from a set of previously-solved instances for the BSP. Experiments presented in Section 5 use the solutions computed by an EA, but other reference algorithms could be used. In the learning process of VS, each synchronization node is considered independently. Different combinations of features can be applied for training. Options considered for each feature vector include:

- *Synchronization node features*: walking time between the pair of bus stops, transfer demand, and maximum waiting time.
- *Inbound line features*: travel time to synchronization node, minimum allowed headway, and maximum allowed headway.
- *Outbound line features*: travel time to synchronization node, minimum allowed headway, and maximum allowed headway.

Two different classifiers are trained to predict the headway of the inbound

line (y_b^i) and the headway of the outbound line (y_b^j). Thus, two different training datasets are built, each one comprised of the same number of training vectors with identical features but with different labels, corresponding to the headways of the inbound and outbound lines as computed by the reference EA. In the proposed method, one BSP instance yields as many training observations to each dataset as the number of synchronization nodes in the instance times the number of independent executions of the reference EA. The scikit-learn implementation of Random Forests was used as a classifier for the implementation of VS [40].

4.3. Execution of VS for BSP

Figure 4 shows the workflow of VS when solving the BSP. VS receives as input the BSP instance, including the features corresponding to synchronization nodes and bus lines. Since the training phase considers each synchronization node independently, the prediction can be highly parallelized. At the finest-grain, the prediction for each pair of bus lines of a synchronization node is computed in parallel using copies of the two trained classifiers. Moreover, the prediction of the headways of the inbound and outbound lines for a given synchronization node can also be parallelized, since it is performed by different classifiers. The output of each classifier is the predicted headway of the inbound or outbound line of the synchronization node ($\hat{y_b^i}$ and $\hat{y_b^j}$).

A given bus line can be part of multiple synchronization nodes, acting as either the inbound or outbound line. Thus, when the predictions of each classifier are gathered, multiple (possibly conflicting) predictions can be made for a given bus line. Those predictions are stored in a list. Additionally, a prediction may be invalid if the predicted headway is not within the range of allowed headways for that line. Since the set of classification labels is comprised of every allowed headway for every bus line, a classification error could lead to predicting a label that corresponds to a valid headway for some line in the training set but not for the bus line being predicted. Consequently, predicted headways are checked in this step and unfeasible predictions are not included in the list of candidate headways for the line. Once all the predictions are gathered, multiple candidate

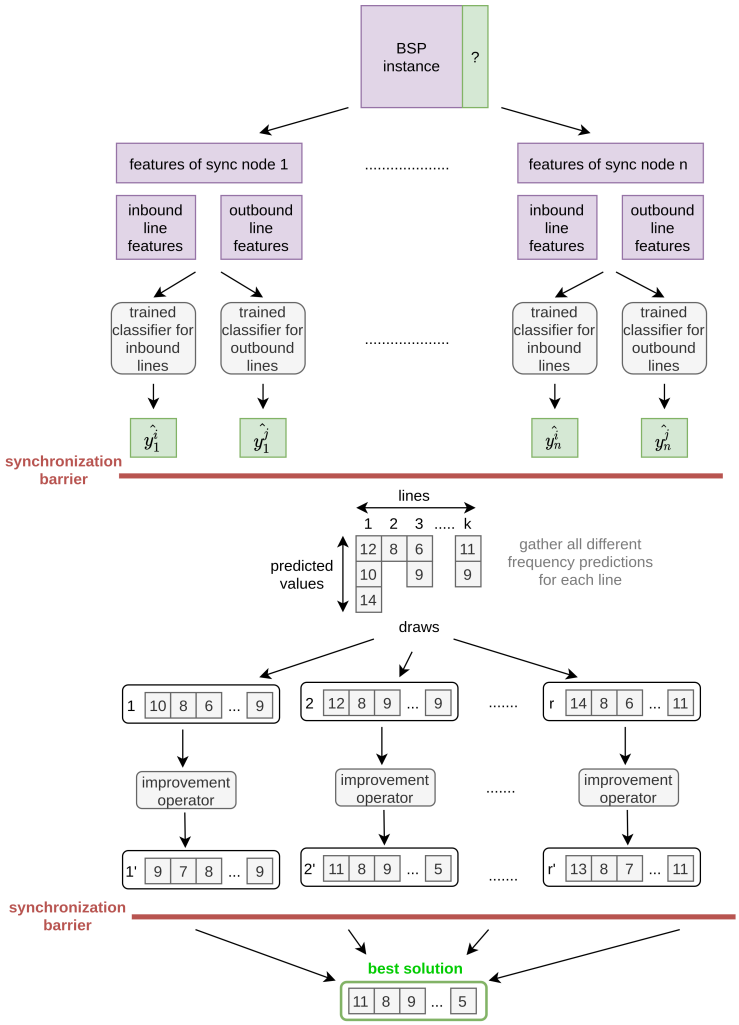


Figure 4: VS workflow for the BSP.

solutions are generated to be further refined through the improvement operator. Candidate solutions are built using the list of headways computed in the prediction phase and according to the following criteria. For a given bus line:

- if only one valid prediction is available for the line, the corresponding headway is fixed for the line in the candidate solution.
- if more than one valid prediction is available for the line, a random headway is selected among the set of valid predictions (according to a uniform probability distribution).

- if no valid predictions are available for the line, a random headway is chosen within the valid range of headways for that line (according to a uniform probability distribution).

Following the described procedure, multiple candidate solutions are generated to be further refined in parallel, using a master-slave approach. The improvement operator is a Local Search (LS) that tries to enhance the solution by iteratively selecting a bus line to randomly change its assigned headway (according to a uniform distribution in the range of valid headways for the line). The change is accepted if the quality of the solution improves, otherwise it is discarded. The quality of the solution is measured through a score function, which reflects the problem formulation, and is used by the reference EA as a fitness function. The score function accounts for the number of synchronized transfers and their corresponding demands and is used as a fitness function by the EA used as reference. The score of a given solution is computed by the procedure described in Algorithm 1.

The score function outlined in Algorithm 1 iterates over each synchronization node in the instance (loop in line 2), obtaining all the features of the synchronization node (line 3), i.e., travel time for the inbound and outbound lines (TT_b^i, TT_b^j), walking time between bus stops ($WT_b^{i,j}$), passenger demand ($P_b^{i,j}$), and maximum allowed waiting time ($W_b^{i,j}$). Additionally, the headway for the inbound and outbound lines (y_b^i and y_b^j) are retrieved from the solution being evaluated (line 4). The algorithm then iterates over each pair of trips of the inbound and outbound lines (loops in lines 5–6). The waiting time for a pair of trips is computed as the difference between the time the outbound and the inbound lines arrive at the synchronization node, subtracting the walking time between bus stops (line 7). In the worst case, passengers have to wait for the full headway of the outbound line (lines 8–9). A pair of trips is synchronized if the computed waiting time is below the threshold $W_b^{i,j}$ that passengers are willing to wait for the outbound line (line 10). In this case, the demand is accumulated according to the problem formulation (line 11). The objective function is

Algorithm 1: Score function to evaluate BSP solutions.

```
   input  : solution, sync_nodes, T, C
   output : score
 1 score ← 0
 2 foreach node b in sync_nodes do
 3 |   TT_b^i, TT_b^j, WT_b^{i,j}, P_b^{i,j}, W_b^{i,j} ← get_features(b)
 4 |   y_b^i, y_b^j ← get_headways(solution, b)
 5 |   for m=0 to T step y_b^i do // Iterate over inbound trips
 6 |   |   for n=0 to T step y_b^j do // Iterate over outbound trips
 7 |   |   |   wait_time = (n + TT_b^j) − (m + TT_b^i) − WT_b^{i,j}
 8 |   |   |   if wait_time > y_b^j then
   |   |   |   |   // At most, passengers wait for the full headway
   |   |   |   |   wait_time ← y_b^j
 9 |   |   |   end
10 |   |   |   if wait_time > 0  &  wait_time ≤ W_b^{i,j} then
   |   |   |   |   // multiply objective by T, to avoid division in f_i = T/y_b^i
11 |   |   |   |   score ← score + min(P_b^{i,j} · y_b^i, C · T)
12 |   |   |   |   break
13 |   |   |   end
14 |   |   end
15 |   end
16 end
```

multiplied by the planning period T to avoid the division present in $f_i = T/y_b^i$ in the original formulation. This simple transformation was performed to be able to solve problem instances to optimality using integer linear programming solvers, which is regarded as a line of future work. The break directive in line 12 ensures that each trip of the inbound line is synchronized with only one trip of the outbound line.

Once all candidate solutions are improved results are gathered and the best solution computed is returned.

5. Experimental Analysis

This section describes the experimental evaluation of the proposed VS to solve the BSP. Section 5.1 introduces the problem instances used for the analysis and Section 5.2 presents the baseline solutions used for comparison. Finally, the experimental results for both the synthetic and realistic instances are sum-

marized in sections 5.3 and 5.4, respectively.

5.1. Problem instances

Two sets of problem instances were considered: synthetic instances and realistic instances from the public transportation system of Montevideo, Uruguay. Both types of instances consider minutes as time units and the planning period is $T = 120$ minutes. The methodology for building instances is described next.

5.1.1. Synthetic BSP instances

A set of 130 synthetic BSP instances was built using a grid topology (dimension $M \times N$ blocks), modeling the streets in a city. L lines were generated, of two types: i) $\alpha \cdot L$ regular lines, with start and end points far apart; and ii) $(1-\alpha) \cdot L$ circular lines, with the start and end of the line close to each other. Bus stops for each line were placed at grid intersections, at distance d^* between each other. A set of synchronization nodes was selected over the defined network, considering those lines with nearby bus stops (i.e., distance less than r blocks). Out of the S synchronization nodes, subsets of size $\beta \cdot S$, $\beta \in [0, 1]$ were selected to define different problem instances over the same network topology. The walking speed was ws (in blocks per time unit). Travel times for each line were defined assuming a speed s (in blocks per time unit). h^i values were randomly selected with uniform probability from $[h_{\min}, h_{\max}]$. H^i was defined as $H^i = \lceil c \cdot h^i \rceil$, with c randomly selected with uniform probability from $[c_{\min}, c_{\max}]$. The demand $P_b^{i,j}$ for each synchronization node was randomly selected with uniform probability from $[P_{\min}, P_{\max}]$. The maximum waiting time for each transfer was defined according to the maximum headway of the outbound line: $WT_b^{i,j} = \lambda \cdot H^j$, with $\lambda \in \{0.70, 0.9, 1.0\}$. Smaller values of λ correspond to a tighter time constraint, modeling passengers unwilling to wait for the bus for long periods.

5.1.2. Realistic BSP instances

Realistic BSP instances were built using data from the public transportation system in Montevideo, Uruguay. Different sources of data were used, including open data from the city transportation authorities (bus lines, bus stops, and

timetables), and real transfers information from ticket sales using smartcards in 2015 [41]. The key elements of the generated instances are described next.

Bus trips correspond to working days departing between 12:00 and 14:00 (the peak of ticket sales) [41]. A set of 45 problem instances was defined, 15 for each of the dimensions considered (30, 70, and 110 synchronization nodes). Synchronization nodes were randomly chosen with uniform probability among the 171 most demanded transfers for the considered period, shown in Figure 5.

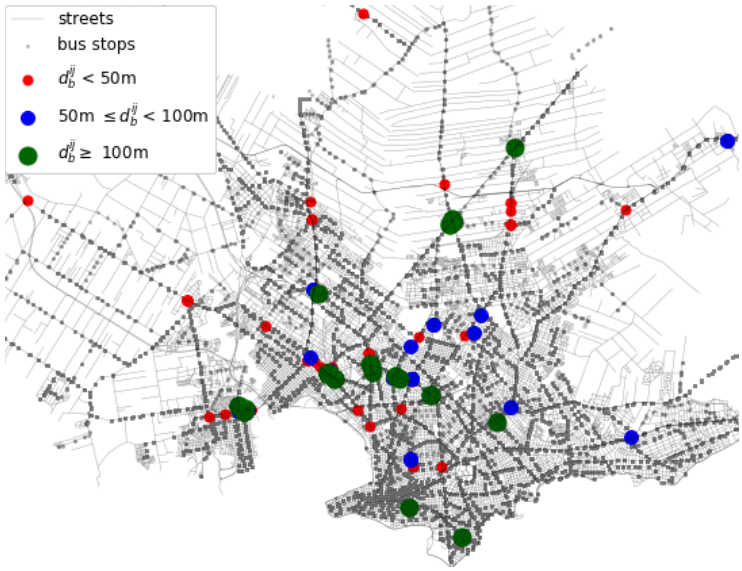


Figure 5: Transfer zones in realistic BSP instances in Montevideo, Uruguay.

Only 53 of the 171 synchronization nodes share the same stop for both lines. The remaining 118 are real transfer zones, as defined by the new BSP model. Of them, in 74 the bus stops are separated at most 50 m (red circles), 49 between 50–100 m (blue circles), and 48 are more than 100 m apart (green circles).

The demand on each synchronization node P_i was assigned according to the real tranfers in May 2015, as registered by smartcard transactions. The bus capacity was fixed to $C = 5$, based on real data and accounting for passengers already on the bus and passengers boarding without transferring. A sensitivity analysis showed that solutions are weakly dependant on the chosen value of C for the studied instances. The walking time between bus stops $WT_b^{i,j}$ was computed according to the distance between the bus stops and assuming a

constant walking speed of 6 km/h. The travel time of the bus lines to the synchronization node were defined using publicly available timetables, averaging the travel times of all trips of each line within the considered planning period. The range of allowed headways for each line $[h^i, H^i]$ was also computed based on the public timetables, considering all trips within the planning period. The maximum waiting time for each transfer was defined according to the maximum headway for the outbound line i.e., $WT_b^{i,j} = \lambda \cdot H^j$, with $\lambda \in \{0.7, 0.9, 1.0\}$.

Table 1 reports the parameter values for both types of instances.

Table 1: Parameter configuration for BSP instances (n/a: not applicable).

parameter	*synthetic instances*	*Montevideo instances*
T	120	120
C	5	5
$M \times N$	$50\times50, 300\times300, 350\times350$ $400\times400, 450\times450$	n/a (city dimensions $\sim$ 37×24,km
L	$50, 300, 350, 400, 450$	94
β	0.5	0.17, 0.41, 0,58
α	0.5	n/a (real data)
d^*	3 blocks	n/a (real data)
r	2 blocks	171 meters (real data)
ws	1 block/time unit	6 km/h
s	4 blocks/time unit	n/a (real timetable)
$(h_{\min},h_{\max})$	$(5, 15)$	$(5, 36)$ (real data)
$(c_{\min}, c_{\max})$	$(3, 5)$	$(1.13, 11.55)$ (real data)
$(P_{\min}, P_{\max})$	$(6, 40)$	(6,56) (real data)
λ	$0.70, 0.9, 1.0$	$0.70, 0.9, 1.0$

5.2. Baseline solutions for results comparison

A set of baseline solutions were considered to evaluate the efficacy of VS:

- *EA*: the best solution computed by the reference EA described in our previous work [21]. The EA works over a population of candidate solutions that includes randomly-generated headway configurations and is seeded using headways configuration from the current reality and greedy approaches. Standard evolutionary operators are applied (i.e., tournament selection, and custom variants of two-point crossover and Gaussian mutation) to preserve specific features of lines already synchronized in

parent solutions, trying to keep useful information in the offspring generation process. Parameters were configured to: population size 100, 1 000 generations, crossover probability 0.9 and mutation probability 0.01.

- h^i: a solution where the headway of each bus line is assigned to its minimum allowed value (h^i). This solution is too expensive to put into practice, since configuring all lines to operate at minimum headways accounts for a very large number of vehicles and high operating costs.
- *random*: a randomly generated solution where the headway of each bus line is chosen within the valid range of headways for that line (according to a uniform probability distribution).
- *LS (1000)*: a *random* solution followed by 1000 steps of the same LS operator used in the improvement phase of VS.
- *real*: the current solution according to the public timetable available for the transportation system of Montevideo, Uruguay, only used in the experimental evaluation of realistic BSP instances. This solution was computed by assigning to each bus line the real average headway determined by the city authorities for the considered planning period.

Solutions were computed on an Intel Core i7-10510U CPU at 1.80GHz, 16 GB RAM, except for the EA which was executed on a Quad-core Xeon E5430 at 2.66GHz, 64 GB RAM, from National Supercomputing Center (Cluster-UY) [42].

5.3. Results on synthetic instances

This subsection reports the evaluation of VS over synthetic BSP instances.

5.3.1. Training of VS for BSP

The set of 130 problem instances was divided into a training set comprised of 70 instances and a test set comprised of 60 instances, including the larger ones in terms of the number of bus lines. Thirty independent executions of the

reference EA were performed over each instance of the training set and the best solution found on each execution was included in the dataset of solved instances.

Several configurations were evaluated, in terms of accuracy (i.e., the number of correct predictions out of all predictions made), for the feature vector of the classifiers. Configuration C1 considers features of the synchronization node: the bus lines identifiers i and j, the walking time between bus stops $WT_b^{i,j}$, the transfer demand $P_b^{i,j}$ for the node, the maximum allowed waiting time $W_b^{i,j}$, and the traveling times of both bus lines to the synchronization node (TT_b^i and TT_b^j). Configuration C2 uses the same feature vector as C1, but without the bus line identifiers. Configuration C3 adds three features from the problem instance to configuration C2: the number of bus lines L, the number of synchronization nodes S, and the planning period T. Finally, configuration C4 has the same features as C2 but also includes the minimum and maximum allowed headways for the inbound and outbound bus lines. Configurations and their corresponding accuracy are reported in Table 2.

Table 2: Accuracy of different feature configurations for synthetic BSP instances.

configuration	*feature vector*	*accuracy*	
		inbound	*outbound*
C1	$\langle i, j, WT_b^{i,j}, P_b^{i,j}, W_b^{i,j}, TT_b^i, TT_b^j \rangle$	0.17	0.31
C2	$\langle WT_b^{i,j}, P_b^{i,j}, W_b^{i,j}, TT_b^i, TT_b^j \rangle$	0.17	0.29
C3	$\langle L, S, T, WT_b^{i,j}, P_b^{i,j}, W_b^{i,j}, TT_b^i, TT_b^j \rangle$	0.17	0.29
C4	$\langle h^i, H^i, h^j, H^j, WT_b^{i,j}, P_b^{i,j}, W_b^{i,j}, TT_b^i, TT_b^j \rangle$	0.50	0.48

The similar results of C1 and C2 indicate that the bus line identifiers are not useful for accurately predicting headways. Results of C3 suggest that accuracy does not improve when adding global information from the problem instance. Configuration C4 significantly outperformed all the others, with accuracy values of 0.50 and 0.48 for the inbound and outbound lines, respectively. Consequently, configuration C4 was used for the remainder of the experimental evaluation.

5.3.2. Accuracy per instance

The prediction phase of VS was evaluated in terms of accuracy. Table 3 reports statistics of the prediction accuracy of inbound and outbound lines clas-

sifiers for the 60 synthetic BSP instances in the test set. Average (*mean*) and standard deviation (*std*) are reported, along with the minimum (*min*), the maximum (*max*), and the quartiles (Q_1,Q_2,Q_3) of the accuracy values achieved.

The classifier for predicting the headways of inbound lines has 59 classes, i.e., it predicts one headway value from within a set of 59 possible values. The classifier for outbound lines has 68 possible classes. Thus, a random prediction over possible values would render an average accuracy of 0.017 for the inbound classifier and 0.015 for the outbound classifier. Results in Table 3 show a very good prediction accuracy for both classifiers. The maximum prediction accuracy was 0.60 (inbound) and 0.56 (outbound). In median, the accuracy of the inbound classifier 0.49 and the accuracy of the outbound classifier was 0.47.

5.3.3. Number of valid predictions

Since predictions for each synchronization node are independent, many different valid predictions can be made for the same line, acting as either inbound or outbound line of multiple synchronization nodes. Additionally, predictions may not fall within the allowed range of headways for the line, and thus some lines may be left without a valid headway in the prediction phase. With zero valid predictions, the headway is randomly set within the allowed range. When more than one valid prediction is available, one is picked at random to generate a solution. Table 4 reports the number of valid predictions for all instances.

Table 3: Prediction accuracy per instance for synthetic BSP instances.

	accuracy	
	inbound	outbound
mean	0.49	0.47
std	0.04	0.04
min	0.40	0.37
Q_1	0.46	0.44
Q_2	0.49	0.47
Q_3	0.52	0.50
max	0.59	0.56

Table 4: Number of valid predictions per line for synthetic BSP instances.

valid predictions	# lines
0	2
1	14201
2	2484
3	239
4	18

A single headway value was predicted for most lines. Just for two lines no valid prediction was made. In some cases more than one valid headway was

predicted for the same line (at most four values for the same bus line).

5.3.4. Comparison with baseline solutions

Figure 6 compares the best scores achieved by VS and the baseline solutions, (normalized using the EA results). Results correspond to 30 independent executions of each algorithm, except for h^i, which is deterministic. Results for three different configurations of VS are displayed: VS (0) is the prediction phase (without applying the LS improvement operator), VS (1000) and VS (5000) are VS with a 1000-step and 5000-step LS improvement operator, respectively.

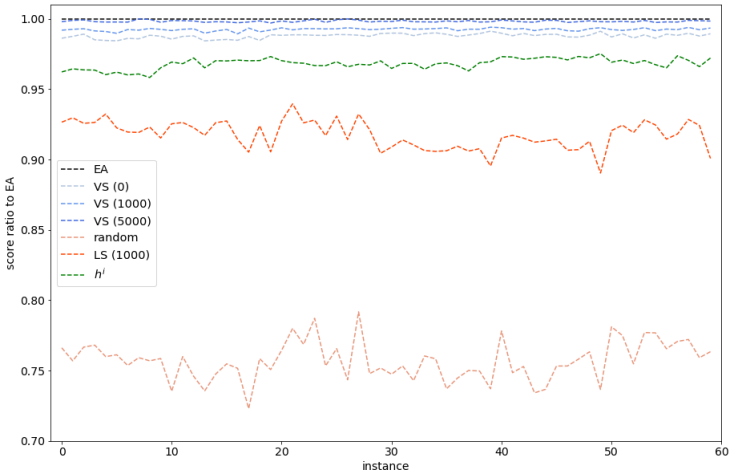


Figure 6: Comparison of VS with baseline solutions for synthetic BSP instances.

VS already computed accurate results in the prediction phase. VS (0) outperformed the random solution by up to 26.4% and the minimum headway solution by up to 3.0%. The prediction phase of VS computed solutions that are up to 99.1% as good as the reference EA (median 98.8%).

The LS operator of VS allowed further improving the computed solutions. Using a 1000-step LS, solutions computed by VS were up to 99.4% as good as the reference EA (median 99.3%). With a 5000-step LS, VS computed solutions 99.9% as good as the reference EA (median 99.8%). The effectiveness of the LS operator is demonstrated by an improvement of up to 19% of LS (1000) over the random solution. Thus, each phase of VS contributes to the overall results.

Instances over grids with dimension 450×450 were totally unseen for VS, which was able to effectively scale in the problem dimension and accurately solve them.

5.4. Results on realistic instances

This section reports the results of VS over the set of realistic BSP instances from the public transportation system in Montevideo, Uruguay.

5.4.1. Training of VS for BSP

The set of instances was divided into a training set (30 instances with 30 and 70 synchronization nodes) and a test set (15 instances with 110 synchronization nodes). The training set was solved using the reference EA and the best solution found on each of 30 independent executions was included in the dataset of solved instances. Each training set had 45 000 vectors, from the 30 independent executions of each of the 15 instances of sizes 30 and 70 synchronization nodes.

The features included in the training vector were those that achieved the best results for the synthetic instances, i.e., the range of allowed headways and travel times for both lines in the synchronization node, the walking time between the bus stops, the maximum allowed waiting time, and the passenger demand. As with the synthetic BSP instances, two separate classifiers were trained to predict the headways of inbound and outbound lines.

5.4.2. Accuracy per instance

Table 5 reports statistics of the accuracy of the inbound and outbound lines headway predictions for each instance in the dataset. Average (*mean*) and standard deviation (*std*) are reported, along with the minimum (*min*), the maximum (*max*), and the quartiles (Q_1,Q_2,Q_3) of the accuracy values achieved.

Results show an overall good prediction accuracy. For the inbound line classifier, the maximum prediction accuracy was 0.77, the minimum was 0.63, and the median was 0.67. For outbound lines, the highest prediction accuracy was 0.79, the minimum was 0.54, and the median was 0.72. These are promising results when considering the number of class labels (i.e., number of possible

Table 5: Prediction accuracy per instance for realistic BSP instances.

	accuracy	
	inbound line	outbound line
mean	0.68	0.71
std	0.04	0.06
min	0.63	0.54
Q_1	0.65	0.70
Q_2	0.67	0.72
Q_3	0.69	0.74
max	0.77	0.79

predictions) of each classifier: 42 for inbound and 27 for outbound lines. Thus, a random prediction over the possible values would render an average accuracy of 0.02 (inbound classifier) and 0.04 (outbound classifier).

5.4.3. Number of valid predictions

Figure 7 presents histograms of the number of valid predictions per line for each studied instance. Results show that most bus lines have a single valid prediction on all problem instances. In these cases, the predicted headways were used directly for generating solutions. Some instances had lines with no valid predictions. In instances #3, #6, and #9, no valid predictions could be made for two of the 78 bus lines in each problem instance. In such cases, headways assigned to that lines were randomly generated within the allowed ranges. Conversely, some lines have more than one valid prediction. The largest number of valid predictions for the same line an all problem instances was four (instances #2 and #4). In these cases, the headway for the line was randomly selected among the valid set of predictions.

5.4.4. Comparison to baseline solutions

Table 6 reports the best results on 30 independent executions of each algorithm (except h^i and real that are deterministic). Three configurations of VS are studied: VS (0), without the improvement operator (i.e., only the prediction phase); VS (1000), applying a 1000-steps LS improvement operator; and VS (5000); applying a 5000-steps LS. Score results are normalized considering

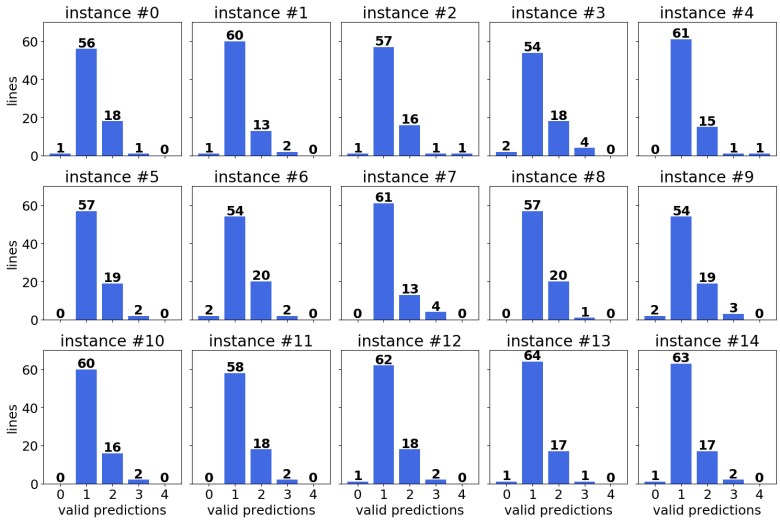


Figure 7: Number of valid predictions per line for realistic BSP instances.

the reference EA. The best result for each instance is highlighted in grey. Figure 8 shows boxplots of the score distribution, compared to the reference EA.

Table 6: Results comparison of VS with baseline solutions for realistic BSP instances.

#I	*real*	h^i	*random*	*LS (1000)*	*EA*	*VS (0)*	*VS (1000)*	*VS (5000)*
0	0.9282	0.9757	0.9168	0.9939	1.0	0.9952	0.9987	0.9989
1	0.8816	0.9689	0.8835	0.9924	1.0	0.9964	0.9999	1.0002
2	0.9290	0.9765	0.9093	0.9927	1.0	0.9965	0.9990	0.9994
3	0.9161	0.9769	0.9090	0.9904	1.0	0.9877	0.9985	0.9994
4	0.9279	0.9749	0.9059	0.9950	1.0	0.9984	1.0003	1.0005
5	0.9170	0.9740	0.9182	0.9923	1.0	0.9984	1.0002	1.0008
6	0.8861	0.9695	0.8639	0.9896	1.0	0.9895	0.9993	1.0009
7	0.8973	0.9670	0.8787	0.9955	1.0	0.9958	0.9996	1.0004
8	0.8817	0.9647	0.8842	0.9907	1.0	0.9951	1.0014	1.0017
9	0.9165	0.9773	0.9027	0.9924	1.0	0.9907	0.9988	0.9996
10	0.9280	0.9750	0.9127	0.9955	1.0	0.9985	1.0004	1.0008
11	0.9170	0.9740	0.9142	0.9931	1.0	0.9976	0.9998	1.0003
12	0.9239	0.9747	0.9268	0.9906	1.0	0.9912	0.9999	1.0006
13	0.8883	0.9683	0.8707	0.9913	1.0	0.9933	1.0003	1.0010
14	0.9238	0.9746	0.9074	0.9921	1.0	0.9954	0.9996	1.0002

VS was able to compute accurate results even when considering only the prediction phase. VS (0) outperformed the real solution by up to 11.5% and the random solution by up to 12.6% and the minimum headway solution by up to 3.0%. In the best case, the solution computed using only the prediction phase of VS was 99.8% as good as the solution computed by the reference EA. In

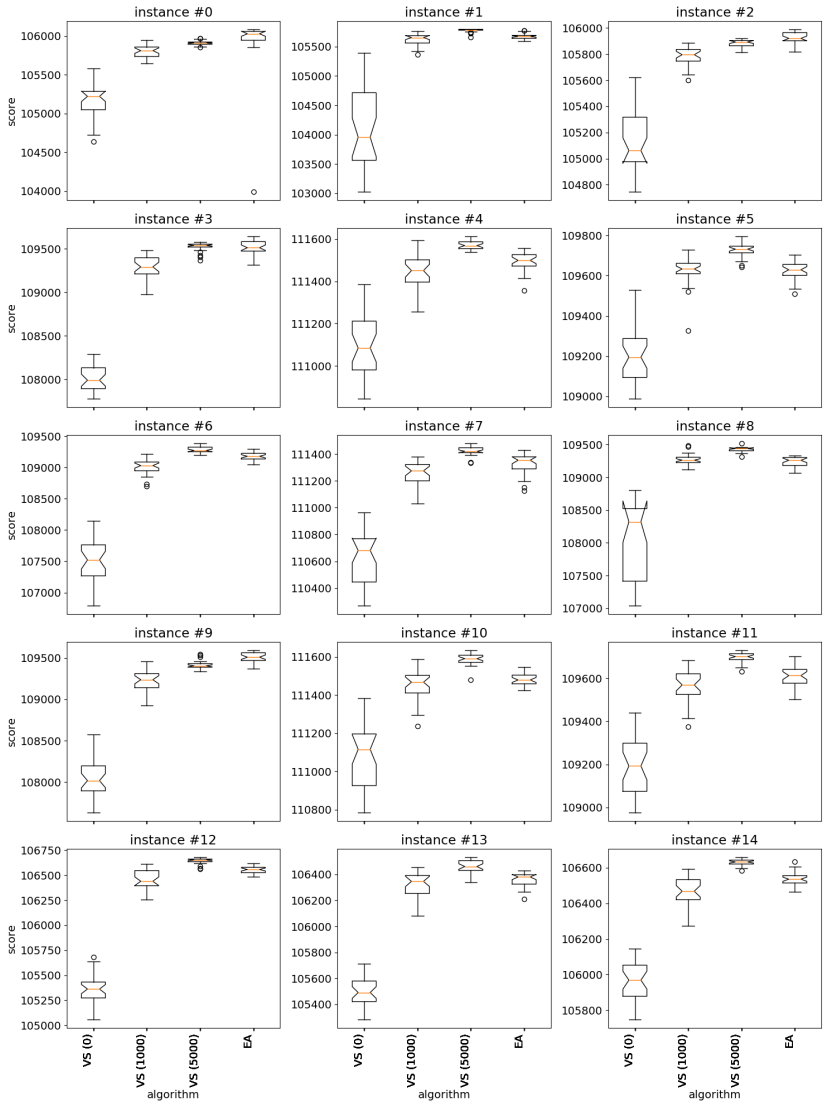


Figure 8: Results comparison of VS vs. EA on realistic BSP instances.

median, the prediction phase computed solutions that were 99.5% as good as the reference solution and, in the worst case, the computed solution was 98.7% of the EA solution. These are highly competitive results, that demonstrate the capability of the prediction phase of VS to compute accurate solutions.

Including the LS operator of VS allowed further improving the computed solutions. In median, adding a 1000-step LS allowed reaching solutions 0.37%

closer to the reference computed by the reference EA. When performing 5000 steps, the improvement increased up to 1.56% and VS outperformed the reference EA in eleven out of fifteen problem instances. Boxplots show that the LS operator was also useful to reduce the variance in the computed results.

VS (0) outperformed the random solution by 8.80% in median, and VS (1000) outperformed LS (1000) on all studied problem instances. These results show that both phases of VS contributed to the overall result.

Furthermore, VS was trained with BSP instances with up to 70 synchronization nodes and evaluated on instances with up to 110 nodes. The experimental results show that VS was able to effectively scale in the problem dimension, solving instances larger than those seen in the training phase.

6. Conclusions and Future Work

This article presented the application of VS to solve the BSP, a complex real-world combinatorial optimization problem.

A new problem formulation was defined, extending previous formulations in the literature by considering transfers at any pair of bus stops in the public transportation system and accounting for the walking time between bus stops. The problem was solved applying the learning/optimization approach in VS, using RF as classifiers. VS was trained using an EA as a reference and several feature configurations were studied to improve prediction accuracy.

The experimental analysis was performed over 130 synthetic instances and 45 realistic instances from the public transportation system in Montevideo, Uruguay. VS was able to compute accurate solutions. In the synthetic dataset, VS solutions were (in median) within 1.2% of the reference EA (prediction phase) and within 0.2% when including a 5000-step LS improvement operator. In realistic instances from Montevideo, VS solutions were (in median) 99.5% as good as the reference EA (prediction phase) and outperformed the EA in eleven out of fifteen problem instances when using the 5000-step LS.

Solving the BSP allowed studying the applicability of the VS paradigm to

a real-world optimization problem and evaluating its effectiveness with respect to baseline solutions. The applied problem decomposition involved training two machine learning classifiers. The experimental evaluation was performed over larger instances (in terms of the number of bus lines and synchronization nodes) than those considered in the training phase. The experimental results highlight the scalability properties of VS in terms of the problem dimension. VS was able to outperform an EA specifically tuned for the BSP, while significantly reducing the computational burden in terms of the number of evaluations of the objective function.

The main lines of future work include: comparing VS to other soft computing methods, evaluating the parallel capabilities of VS when solving the BSP, addressing other problem formulations (e.g., multiobjective approaches), and extending the experimental analysis considering historical GPS location data of buses that can be used to obtain more accurate approximations of headways and travel times in the public transportation system. Moreover, dynamic models should be explored to take into account traffic and service disruptions, as well as shift in demands within the planning period.

Acknowledgements

The work of R. Massobrio and S. Nesmachnow was partially funded by ANII and PEDECIBA Uruguay. The work of R. Massobrio was partially funded by Fundación Carolina, Spain, and European Union-NextGenerationEU. The work of B. Dorronsoro was supported by the Spanish Ministerio de Ciencia, Innovación y Universidades and the ERDF under contract RTI2018-100754-B-I00 (iSUN), ERDF under project FEDER-UCA18-108393 (OPTIMALE), and Junta de Andalucía and ERDF (GENIUS – P18-2399).

References

[1] S. Grava, Urban Transportation Systems, McGraw-Hill, 2002.

[2] S. Nesmachnow, S. Baña, R. Massobrio, A distributed platform for big data analysis in smart cities: combining Intelligent Transportation Systems and socioeconomic data for Montevideo, Uruguay, EAI Endorsed Transactions on Smart Cities 2 (5) (2017) 1–18.

[3] A. Ceder, N. Wilson, Bus network design, Transportation Research Part B: Methodological 20 (4) (1986) 331–344.

[4] A. Ceder, O. Tal, Timetable Synchronization for Buses, in: Computer-Aided Transit Scheduling, 1999, pp. 245–258.

[5] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti, M. Protasi, Complexity and approximation: Combinatorial optimization problems and their approximability properties, Springer Science & Business Media, 2012.

[6] S. Sra, S. Nowozin, S. Wright, Optimization for machine learning, MIT Press, 2012.

[7] M. Vlastelica, A. Paulus, V. Musil, G. Martius, M. Rolínek, Differentiation of blackbox combinatorial solvers, in: International Conference on Learning Representations, 2020.

[8] O. Vinyals, M. Fortunato, N. Jaitly, Pointer Networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2692–2700.

[9] F. Pinel, B. Dorronsoro, P. Bouvry, The virtual savant: Automatic generation of parallel solvers, Information Sciences 432 (2018) 411–430.

[10] International Transport Forum, Valuing Convenience in Public Transport, OECD Publishing, 2014.

[11] O. Ibarra, Y. Rios, Synchronization of bus timetabling, Transportation Research Part B: Methodological 46 (5) (2012) 599–614.

[12] G. Saharidis, C. Dimitropoulos, E. Skordilis, Minimizing waiting times at transitional nodes for public bus transportation in Greece, Operational research 14 (3) (2014) 341–359.

[13] O. Ibarra, F. López, Y. Rios, Multiperiod Bus Timetabling, Transportation Science 50 (3) (2016) 805–822.

[14] A. Ceder, B. Golany, O. Tal, Creating bus timetables with maximal synchronization, Transportation Research Part A: Policy and Practice 35 (10) (2001) 913–928.

[15] W. Domschke, Schedule synchronization for public transit networks 11 (1) (1989)

17–24. doi:10.1007/bf01721163.
URL https://doi.org/10.1007/bf01721163

[16] J. Daduna, S. Voß, Practical Experiences in Schedule Synchronization, in: Computer-Aided Transit Scheduling, 1995, pp. 39–55.

[17] C. Fleurent, R. Lessard, L. Séguin, Transit timetable synchronization: Evaluation and optimization, in: Computer-aided Scheduling of Public Transport, 2004.

[18] R. Hall, M. Dessouky, Q. Lu, Optimal holding times at transfer stations, Computers & Industrial Engineering 40 (4) (2001) 379–397.

[19] F. Delgado, N. Contreras, J. C. Munoz, Holding for transfers, in: Transportation Research Board 92nd Annual Meeting, Vol. 2013, 2013.

[20] N. van Oort, N. H. Wilson, R. van Nes, Reliability improvement in short headway transit services: Schedule-and headway-based holding strategies, Transportation research record 2143 (1) (2010) 67–76.

[21] S. Nesmachnow, J. Muraña, G. Goñi, R. Massobrio, A. Tchernykh, Evolutionary Approach for Bus Synchronization, in: High Performance Computing, 2020, pp. 320–336.

[22] S. Nesmachnow, J. Muraña, C. Risso, Exact and metaheuristic approach for bus timetable synchronization to maximize transfers, in: III Ibero-American Congress on Smart Cities, 2020, pp. 1–15.

[23] Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: A methodological tour d'horizon, European Journal of Operational Research 290 (2) (2021) 405 – 421.

[24] D. Grimes, G. Ifrim, B. O'Sullivan, H. Simonis, Analyzing the impact of electricity price forecasting on energy cost-aware scheduling, Sustainable Computing: Informatics and Systems 4 (4) (2014) 276 – 291.

[25] E. Demirović, P. Stuckey, J. Bailey, J. Chan, C. Leckie, K. Ramamohanarao, T. Guns, An investigation into prediction + optimisation for the knapsack problem, in: Integration of Constraint Programming, Artificial Intelligence, and Operations Research, 2019, pp. 241–257.

[26] A. Elmachtoub, P. Grigas, Smart "predict, then optimize", arXiv preprint arXiv:1710.08005 [unpublished].

[27] J. Mandi, P. J. Stuckey, T. Guns, et al., Smart predict-and-optimize for hard com-

binatorial optimization problems, in: AAAI Conference on Artificial Intelligence, 2020, pp. 1603–1610.

[28] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, F. Bach, Learning with differentiable perturbed optimizers, in: 34^{th} Conference on Neural Information Processing Systems, 2020, pp. 1–12.

[29] I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, in: 5th International Conference on Learning Representations, 2017.

[30] H. Hu, X. Zhang, X. Yan, L. Wang, Y. Xu, Solving a New 3D Bin Packing Problem with Deep Reinforcement Learning Method, in: 30^{th} International Joint Conference on Artificial Intelligence, 2017, pp. 1–7.

[31] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, D. L. Dill, Learning a SAT solver from single-bit supervision, in: International Conference on Learning Representations, 2019.

[32] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, A. Kyek, Optimization of global production scheduling with deep reinforcement learning, Procedia CIRP 72 (2018) 1264–1269.

[33] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, H. Xie, Multi-Objective Workflow Scheduling With Deep-Q-Network-Based Multi-Agent Reinforcement Learning, IEEE Access 7 (2019) 39974–39982.

[34] R. Massobrio, B. Dorronsoro, S. Nesmachnow, Virtual Savant for the Knapsack Problem: learning for automatic resource allocation, Proceedings of the ISP RAS 31 (2) (2019) 21–32.

[35] R. Massobrio, S. Nesmachnow, F. Palomo-Lozano, B. Dorronsoro, Virtual savant as a generic learning approach applied to the basic independent next release problem, Applied Soft Computing 108 (2021) 107374.

[36] J. C. de la Torre, R. Massobrio, P. Ruiz, S. Nesmachnow, B. Dorronsoro, Parallel virtual savant for the heterogeneous computing scheduling problem, Journal of Computational Science 39 (2020) 101048.

[37] D. Treffert, Extraordinary People: Understanding Savant Syndrome, iUniverse, 2006.

[38] P. Heaton, G. Wallace, Annotation: The savant syndrome, Journal of child psychology and psychiatry 45 (5) (2004) 899–911.

[39] L. Mottron, M. Dawson, I. Soulières, Enhanced perception in savant syndrome: patterns, structure and creativity, Philosophical Transactions of the Royal Society B: Biological Sciences 364 (1522) (2009) 1385–1391.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[41] R. Massobrio, S. Nesmachnow, Urban Mobility Data Analysis for Public Transportation Systems: A Case Study in Montevideo, Uruguay, Applied Sciences 10 (16) (2020) 1–20.

[42] S. Nesmachnow, S. Iturriaga, Cluster-uy: Collaborative scientific high performance computing in uruguay, in: M. Torres, J. Klapp (Eds.), Supercomputing, Springer International Publishing, Cham, 2019, pp. 188–202.