applied sciences

MDPI

Article

A Data-Driven Based Dynamic Rebalancing Methodology for Bike Sharing Systems

Marco Cipriano, Luca Colomba * and Paolo Garza

Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, 10129 Torino, Italy; marco.cipriano@studenti.polito.it (M.C.); paolo.garza@polito.it (P.G.)
* Correspondence: luca.colomba@polito.it (L.C.)

Abstract: Mobility in cities is a fundamental asset and opens several problems in decision making and the creation of new services for citizens. In the last years, transportation sharing systems have been continuously growing. Among these, bike sharing systems became commonly adopted. There exist two different categories of bike sharing systems: station-based systems and free-floating services. In this paper, we concentrate our analyses on station-based systems. Such systems require periodic rebalancing operations to guarantee good quality of service and system usability by moving bicycles from full stations to empty stations. In particular, in this paper, we propose a dynamic bicycle rebalancing methodology based on frequent pattern mining and its implementation. The extracted patterns represent frequent unbalanced situations among nearby stations. They are used to predict upcoming critical statuses and plan the most effective rebalancing operations using an entirely data-driven approach. Experiments performed on real data of the Barcelona bike sharing system show the effectiveness of the proposed approach.

Keywords: bike sharing systems; dynamic rebalancing; itemset mining; data mining; machine learning; smart mobility; decision support system

Citation: Cipriano, M.; Colomba, L.; Garza, P. A Data-Driven Based Dynamic Rebalancing Methodology for Bike Sharing Systems. *Appl. Sci.* **2021**, *11*, 6967. https://doi.org/10.3390/app11156967

Received: 6 July 2021
Accepted: 27 July 2021
Published: 28 July 2021

1. Introduction

Bike sharing systems are public transportation systems adopted by many different municipalities to reduce traffic congestion and pollution and provide more sustainable means of transport within smart cities. Additionally, compared to the use of private cars, the adoption of bicycles combined with other means of public transport (bike-and-ride) offer environmental benefits and a number of social benefits [1]. Such systems are widely adopted: more than 700 systems are active world-wide [2,3]. In the context of smart cities and Sustainable Development Goals (SDG), understanding the key factors towards large scale adoption by users is fundamental to reduce air pollution and improve quality of life in cities [4–6], according to SDG 11 and 13. In fact, additionally to environmental benefits and a reduced amount of pollution, Ref. [7] quantified health and economical benefits which arose in European cities with large adoption of bicycles. Despite the benefits, such systems require investments from companies and municipalities to be effective and successful. The documented case of the city of Seattle [8] underlines the importance of proper design and the need of maintenance to provide good quality of service. In this context, we highlight the relevance of our work in developing a framework to solve the bike rebalancing problem in bike sharing systems, which can be categorized into two main types: (i) free-floating and (ii) station-based systems (also referred to as dock-based systems). In free-floating systems users can find and drop bicycles at almost any location within a delimited operational area in the city, whereas station-based systems require the user to pick up and drop bicycles in docking stations, specifically located within the administrative boundaries of the municipality. The main advantage of free-floating services is that users can freely drop bicycles everywhere, especially in case their destination is in an area far from points of interests and in outskirts of the city, but the disadvantage is

that when the user needs to pick up a bicycle, none may be available in his/her vicinity. Station-based bike sharing systems may instead provide more reliability since there are several stations spread across the city in the most relevant and popular locations. The main disadvantage of such systems is that upon the arrival at the designated destination, the station may be full, thus requiring the user to reach another (nearby) station to drop the rented bicycle and eventually incurring in higher costs due to the additional trip, or the station from which the trip starts may be empty, requiring the user to walk to another station within walking distance or to use other means of transport.

Station-based bike sharing systems thus require rebalancing operations to maintain users' satisfaction and limit the inabilities to use a station, either for pick-up or drop of bicycles. However, station-based bike sharing services have a great advantage compared to free-floating competitors: easier maintenance operation planning due to the presence of a limited number of stations in fixed locations.

The Bike sharing Rebalancing Problem (BRP) consists of using a set of vans to move bicycles across different stations for rebalancing purposes, avoiding situations in which stations cannot be used as starting or destination points due to them being empty or full, respectively. The BRP problem can be tackled in two different ways, either via static rebalancing or dynamic rebalancing. Static rebalancing strategies are performed when the system is offline or when the system is barely used, i.e., during night time, whereas the dynamic rebalancing strategy consists of moving bicycles across the stations when the system is operating and is used by the users. Dynamic rebalancing operations need to be performed in a short amount of time to avoid large amount of bicycles being unavailable to users due to operators moving them across different stations. For this reason, bicycles are usually moved among nearby stations (from full to empty ones).

In this paper, we propose a fully data-driven and configurable methodology that addresses the dynamic rebalancing problem, leveraging on past data of system's usage and data mining techniques to identify frequent critical situations and plan quickly effective rebalancing operations. We validated the proposed approach in a real scenario by applying the corrective actions generated by our methodology on the Barcelona Bike sharing system. The generated rebalancing actions for compensating system's critical situations are supported by strong interpretable patterns which provide also an explanation of the behaviour of the system under analysis. Specifically, we apply association rules, a well-known data mining technique, to find frequent and relevant patterns concerning unbalanced critical situations among nearby stations. The proposed approach extracts association rules that correspond to recurrent situations in past data in which combinations of almost empty or almost full nearby stations are involved. Based on the extracted rules, we plan in advance local corrective rebalancing actions to compensate for possible future critical situations. To extract such patterns, we apply the state-of-the-art FP-Growth [9] algorithm on a transactional database generated using past stations' status. An example of an extracted rule is $+s_4, +s_7 \rightarrow -s_{11}, sup = 60\%, conf = 80\%$, which means that in 80% of the analysed historical data when the nearby stations s_4 and s_7 are positively critical (i.e., they have significantly more bicycles than their neighbours) then station s_{11} is negatively critical (i.e., it has significantly fewer bicycles than its neighbours). This rule, which is also highly frequent (it occurs in 60% of the historical data), highlights a frequent critical unbalanced behaviour and can be used to plan rebalancing operations among the nearby stations s_4, s_7, and s_{11} moving bicycles from stations s_4 and s_7 to s_{11}.

The contributions of this paper are as follows:

- Characterization and identification of historical critical unbalanced situations among nearby stations through frequent geospatial patterns, leveraging on the definitions of positively and negatively critical station status.
- Proposal of a rebalancing approach, based on the extracted patterns, to address the dynamic bike sharing rebalancing problem. The proposed rebalancing approach (i) trains contextualised models offline and (ii) plans the rebalancing operations in few seconds online when rebalancing actions are needed.

- Contextualised rebalancing operations.

This paper is structured as follows. Section 2 introduces the problem statement and preliminaries, while Section 3 describes the related work and compares previous work with our methodology. Then, Section 4 explains the proposed methodology adopted to solve the online BRP problem, while Section 5 describes the Barcelona bike sharing dataset on which we evaluated our methodology, describing the preprocessing steps adopted and the results we achieved. Finally, Section 6 draws conclusions and presents ideas for future work.

2. Problem Statement and Preliminaries

In this work, we propose a rebalancing methodology that reduces the situations in which bike sharing stations are unusable because of critical status. Specifically, we aim to plan online rebalancing operations subject to the constraint that such actions are performed in a short amount of time to limit any new critical situation or station unavailability. The rebalancing operations are planned by analysing historical data about the occupancy of the stations under analysis. In the following paragraphs, we formalize the addressed problem.

Let $\mathcal{S}$ be the set of stations of the bike sharing system under analysis. Each station $s \in \mathcal{S}$ is characterized by its geographical position.

Let $\mathcal{D}$ be a structured dataset containing the historical information about the number of bicycles and free slots available for each station $s \in \mathcal{S}$. The schema of $\mathcal{D}$ is $\{t, s, used\_slots, free\_slots\}$, where t is a timestamp, $s \in \mathcal{S}$ is a station, $used\_slots \in \mathbb{N}$ is a natural number that represents the number of bicycles available in s at time t, and $free\_slots \in \mathbb{N}$ is the number of free slots available in s at time t.

Let $\mathcal{B}$ be a structured batch of records, with the same schema of $\mathcal{D}$, containing the information about the number of bicycles and free slots available for each station $s \in \mathcal{S}$ at the new timestamp t_{new}.

Let t_{reb} be the time needed for a truck to reach the set of stations we need to rebalance and move bicycles from them to apply the planned rebalance action.

Problem statement. Given the input data $\mathcal{S}$, $\mathcal{D}$, and $\mathcal{B}$, the problem addressed in this work consists of planning at time t_{new} which rebalancing actions should be made within at most t_{reb} minutes from time t_{new} to minimize the number of stations in a critical situation at time $t_{new} + t_{reb}$.

To limit the amount of time during which some bicycles are unavailable because they are on a truck that is moving them from one station to another, the rebalancing problem is subject to the constraint that only local rebalancing operations can be planned, where each local rebalancing operation moves bicycles only among neighbouring stations, i.e., among stations at a maximum distance of d meters from each other.

Before describing the proposed methodology, we provide preliminary definitions.

2.1. Neighbourhood of a Station

As stated previously, the planned rebalancing operations must be local and redistribute bicycles among nearby stations which distance is not greater than a maximum distance threshold d. Thus, we define the neighbourhood of a station $N(s)$ as the set of stations within a range of d meters from its position. The station s itself belongs to its neighbourhood. More formally, given a set of stations S, the neighbourhood of a station $s_i \in S$ is defined as:

$$N(s_i) = \{s_j \in S | dist(s_i, s_j) \leq d\}$$

The distance is computed using the haversine distance and thus does not take into account streets and pedestrian areas.

2.2. Station Neighbourhood and Criticality

Rebalancing operations are fundamental to manage or potentially avoid in advance critical situations in which some stations are critically empty whereas some others are critically full. The entire framework leverages on the definition of critical station that takes

into account the occupancy status of neighbouring stations. To this aim, we first define the occupancy rate OR of a station s at a given timestamp t as:

$$OR_s(t) = \frac{used\_slots_s(t)}{used\_slots_s(t) + free\_slots_s(t)}$$

A station is critical if its occupancy rate is either "much higher" or "much lower" compared to the average occupancy rate of its neighbouring stations. A station in one of those two critical situations can be, or quickly become, full or empty and hence unavailable. To identify stations in a critical situation, we define the critical rate of a station s at timestamp t and we compare it with a user defined threshold cr. The critical rate of a station s at timestamp t is defined as follows:

$$CR_s(t) = OR_s(t) - \frac{1}{|N(s)|} \sum_{s_i \in N(s)} OR_{s_i}(t)$$

As can be seen from the previous equation, the critical rate $CR_s(t)$ can be either positive or negative, depending on the fact that the occupancy rate of station s at time t is higher or lower than the average occupancy rate of its neighbourhood, respectively. Given a criticality threshold cr, if $CR_s(t) > cr$ we say that the station, at time t, is positively critical (i.e., it hosts, with respect to its capacity, "too many" bicycles compared to its neighbouring stations), otherwise if $CR_s(t) < -cr$ we say that the station is negatively critical (i.e., "too few" bicycles are present in the station compared to the average occupancy rate of its neighbourhood). The criticality threshold was introduced to let the system identify critical situations and compensate/react before the bike stations become either completely full or completely empty, i.e., the system performs a sort of "predictive maintenance" operation and tries to act when the stations are already in a critical situation but before the stations become completely unusable.

As we will describe in Section 4, the proposed methodology addresses positively and negatively critical situations and applies rules to balance the number of bicycles stored in those stations, trying to flatten the local occupancy rate using a single vehicle for each unbalanced neighbourhood such that all the critical stations belonging to a subset of a neighbourhood achieve (almost) the same occupancy rate and hence either completely full or completely empty statuses are avoided or at least limited.

2.3. Frequent Itemsets and Association Rules

In this paper, we will use an established data mining technique, called association rule mining, that describes the frequent co-occurrence of sets of items in a large amount of collected data. Association rules have been initially exploited to identify correlations among items in the market basket data analysis context. However, they have been used also in several other contexts such as network traffic data analysis, functional diagnosis, and medical data analysis. The input data, in the association rule mining context, is a dataset $\mathcal{TR}$ composed of a set of transactions tr of arbitrary length, where each transaction $tr \in \mathcal{TR}$ is a set of items. Given an arbitrary dataset $\mathcal{TR}$ as input, the association rule mining problem consists of mining the set of rules of the form $X \rightarrow Y$ that are frequent in $\mathcal{TR}$, i.e., it consists of mining the most frequent correlations among items in $\mathcal{TR}$ and the "direction" of the extracted implications. More formally, an association rule r is represented as an implication in the form $X \rightarrow Y$, where both X and Y are arbitrary sets of items, called antecedent and consequent of the rule, respectively. X and Y are disjoint sets, i.e., $X \cap Y = \emptyset$.

The quality of an association rule is usually measured by means of support and confidence. The support measure corresponds to the frequency of the set $X \cup Y$ in $\mathcal{TR}$ (i.e., the percentage of transactions in $\mathcal{TR}$ that contain both X and Y) while the confidence measure corresponds to the estimation of the conditional probability of finding Y in $\mathcal{TR}$, having found X, and is given by $\frac{sup(X \cup Y)}{sup(X)}$ (i.e., the confidence of $X \rightarrow Y$ is the percentage

of transactions containing both X and Y among those transactions containing X). Only the association rules with a support higher than a minimal support threshold min_sup are mined in order to extract only statistically significant rules.

In our context, each pair $(station, critical\ status)$ is an item that represents a specific critical status of a station. Hence, in our application, each transaction $tr \in \mathcal{TR}$ is a set of pairs $(station, critical\ status)$ and represents the set of stations that are in a critical status in a specific timestamp t. $\mathcal{TR}$ contains one transaction for each timestamp of the time period under analysis. The rules mined by our approach represent frequent co-occurrences among stations that are simultaneously critical.

3. Related Work

In this section, the state-of-the-art techniques adopted for both the static and dynamic BRP problem and the association rules extraction algorithms are discussed.

3.1. The BRP Problem

Designing a proper rebalancing strategy for a station-based bike sharing system is not a trivial task: the operation's planning depends on the city, the number of people which are actively using the system, the season, the elevation, the weather conditions, and the number of trucks available for the rebalancing operation. Considering for example a city with a hilly territory [10] and stations at higher altitude, it is very likely that the number of bicycles taken from such station is much higher than the number of bicycles users dropped throughout the day. Such situation would not appear in a city in which all the stations are placed in a flat area [11]. Different research activities analysed bike sharing systems with different point of views and focusing on different aspects: strategic planning, socio-demographic analyses, predictive modelling, operational research, urban planning, demand and economical analyses [4].

The solution to the BRP problem must consider different aspects of the system and the city, analyse the stations' behaviour and plan the rebalancing policy according to the usage of the system and eventual offline periods. It is important to underline the difficulty of properly comparing different rebalancing techniques, which often minimize or maximize different criteria, such as the travel cost or the users' satisfaction.

3.1.1. Static Rebalancing

The Static Bike Rebalancing (SBR) problem consists of performing maintenance and rebalancing operations when the system is unavailable or rarely used by users, e.g., during the night. Static rebalancing techniques have the advantage to guarantee perfectly balanced stations the moment the system restarts, but does not compensate any anomaly at day time. Furthermore, static rebalancing operations can usually be performed in a larger time span compared to dynamic rebalancing strategies in which the operations need to be performed as fast as possible to not cause disservices to customers. In the literature, previous research papers analysed the static bike rebalancing problem as NP-complete [12] optimization problems known as the Travelling Salesman Problem (TSP) [13,14] or the NP-hard [15] Vehicle Routing with Pickup and Delivery Problem [16–18]. In general, the bike sharing system is modelled as a complete undirected and weighted graph in which the stations and the depot represent the nodes and each edge between node i and node j represents the travelling cost from node i to node j. Since bicycles represent a discrete quantity, such problem is often solved as an integer or mixed integer programming problem, in which one or multiple paths for each vehicle need to be identified to relieve unbalanced bike distribution. To solve such problems, different heuristics and metaheuristics such as the branch and cut method [19] are adopted.

Dell'Amico et al. [11] formulates the static bike rebalancing problem as a special case of the one-commodity pickup and delivery problem, in which a number of vehicles used for rebalancing operations needs to rebalance the stations, minimizing the travelling cost. In the formulation, multiple visits are allowed. The problem is presented with 4

different mathematical formulations which are solved with branch and cut algorithms. The same author analysed the SBR problem with different approaches: in [2], a Destroy and Repair (D&R) metaheuristic algorithm is adapted to solve the routing problem with maximum duration constraint. The D&R algorithm starts from a greedy solution [20] and modifies it using a local search procedure. Then, destroy and repair procedures [21] are applied iteratively until a stop criterion is met. Similarly, Ref. [22] proposed two mixed integer programming formulations, improving existing solutions proposed by Dell'Amico et al. The optimization problem, which includes depot inventory costs, is solved by implementing an improved general variable neighbourhood search algorithm. Dell'Amico et al. [23] also proposed a stochastic programming model to solve the SBR problem, modelling stations' demands and allowing some of stations' requests to not be satisfied [24] by adding a penalty term.

Ref. [25] presented an intractable formulation of the single vehicle route identification problem and proposed two relaxations, solved through branch and cut. The same variant of the problem was dealt by [26,27]. Cruz et al. [27] identifies a solution using an iterated local search-based heuristic. The developed search procedure starts from an initial greedy solution, modified via local search and a perturbation mechanism to escape from local optimal solutions.

An additional and explicit state information to the graph model is introduced by [28], modelling the vehicle's location and the status of all stations. By defining adjacent states and moves between adjacent states, Benchimol et al. formulates the SBR problem as the identification of the sequence of least-cost moves to transit from an initial state to a desired balanced state. The proposed solution adapts the Chalasani–Motwani algorithm [29], developing a 9.5-approximation and a 2-approximation algorithms.

A different approach was proposed by [30]: stations are grouped into several clusters and the original routing problem is decomposed into many separate single-vehicle routing problems. Each cluster represents a group of self-sufficient stations, i.e., bike rebalancing operations are performed within stations of the same cluster. The authors model the stochastic demand of each station and determine target inventory bounds, i.e., service level requirements. The solution is identified by means of mixed integer programming techniques.

3.1.2. Dynamic Rebalancing

The dynamic bike rebalancing problem requires the system administrators to perform the bike repositioning operations to avoid critical situations in certain part of the city when the system is actively used by users. For such reasons, operations need to be timely performed. Compared to the static rebalancing problem, the dynamic counterpart presents more difficulties. In general, it is necessary to predict either the future status of stations, identify common groups of unbalanced stations or determine possible unsatisfied users' requests by modelling future demands. Then, in case rebalancing operations are scheduled to rebalance multiple stations at once, heuristics and meta-heuristics are generally adopted for path planning.

Ref. [31] reformulates the dynamic rebalancing problem as an optimization problem on the complete directed graph associated to the system. More in depth, considering such graph and the number of vehicles adopted for rebalancing purposes, three different mathematical formulations are provided based on space-time networks, arc-flow formulation and the Dantzig–Wolf [32] and Benders [33] decompositions to handle medium and large systems. The problem is formulated as a minimization problem over the total unmet demand with different constraints.

Another approach is analysed by [34], solving the 1-vehicle dynamic regulation problem by dividing the urban area into several regions, one per each rebalancing truck. The authors built a theoretical framework to reduce the imbalance throughout the day by modelling users' demands as a Poisson process. The dynamic bike rebalancing problem is analysed and evaluated in terms of user's satisfaction: the main goal is to identify rebalanc-

ing tours by analysing the complete directed graph and maximize the probability that the first m users find a bike at the designated stations. Different 1- and 2-step heuristics are formulated considering future demands prediction and rebalancing missions are consequently assigned. At most, operators know the next two stations to visit at every mission. Such approach continuously reassign new missions to trucks to keep the system balanced. Contrarily, in our proposed approach the path for each operator and the maximum number of stations involved in the rebalancing process is known and fixed.

Other approaches are based on demand prediction, such as [35]. They trained neural network models to predict future demands and estimate the number of lost and waiting users and the waiting time at each station. The problem is formulated as an optimization problem, taking into account relocation costs and unmet demands costs. Similarly, Ref. [36] builds a demand prediction model and evaluates the importance of each station to prioritize rebalancing actions. Several criteria are analysed and weighted differently by an entropy-based technique to improve rebalancing operations policies. Ref. [37] formulated a mathematical model to solve the rebalancing problem, considering route optimization and users satisfaction and developed a priority-based evolutionary algorithm. Priority evaluation is performed by considering stations' inventory and safe intervals.

An interesting and alternative formulation is given by Chiariotti et al. [38] where, instead of modelling the stochastic users' demand, they use Birth–Death processes [39] to model stations' occupancy and analyse the amount of time a station is self-sufficient, i.e., the amount of time the station is able to satisfy users' requests autonomously. To minimize system failures and rebalancing costs, the system maximizes the survival time for each station and plans rebalancing operations accordingly. The aforementioned survival time is modelled via a Markov–Modulated Poisson process [40], whereas the rebalancing path identification problem is formulated as an optimization problem which takes into account the time remaining till the next predicted system failure. Other works analyse the active users participation in automatic system rebalance through users incentives [41], eventually allowing hybrid approaches with intervention of operators [42].

Hulot et al. [43] include in their analyses weather information in addition to stations' status and focus on the prediction of the number of expected trips per station. In particular, the authors leverage on machine learning models to analyse and predict the hourly arrivals and departures at each station. The predictions are performed on a simplified problem and at test time, the simplified predictions are used to estimate hourly arrivals and departures for each station and determine inventory intervals, i.e., number of bicycles needed. The intervals are used to assign rebalancing operations: a single truck is used to instantaneously compensate an imbalanced station whenever such station does not satisfy the inventory intervals, potentially leading to higher costs associated to rebalancing operations. On the contrary, the framework developed in this paper tries to reduce such costs by moving bicycles between neighbouring stations.

Indeed, many of the previously cited works make some assumptions on the demand or self-sufficiency distribution, eventually leading to lower performances in case such hypotheses do not hold on real systems. Instead, the proposed approach presented in this paper leverages on data mining techniques to identify common critical patterns among the system and thus does not make any prior assumption on data distribution.

3.2. Association Rule Mining

As we introduced in Section 2.3, association rule mining is one of the most used data mining technique to discover relevant and common patterns among data.

An association rule is a fully interpretable pattern identified among the input data with a given strength.

Over the years, different association rule mining algorithm were proposed [44–47]. Two popular choices are the Apriori algorithm [45] and the Frequent Pattern Growth (FP-Growth) algorithm [9]. Apriori is the first algorithm that has been proposed to address the

association rule problem, while FP-Growth is the state-of-the-art one in terms of efficiency. To extract the rules quickly, we used FP-Growth in our approach.

4. Methodology and Methods

To address the problem described in Section 2, we propose a methodology that (i) identifies frequent unbalanced sets of nearby stations in critical situations, representing recurrent critical patterns, analysing past data $\mathcal{D}$ offline and then (ii) plans rebalancing actions to apply online (in at most t_{reb} minutes) leveraging on the mined historical patterns.

The general methodology proposed in this work is fully configurable according to system administrators' needs and can be adapted through time by performing offline periodic updates to find potential new patterns that could emerge due to changes in the city assets or people's habits.

The proposed methodology is composed of two main phases: (i) pattern-based model training on historical data and (ii) online planning of rebalancing operations and their application. The first phase extracts frequent patterns (association rules) that represent sets of recurrent critical situations among nearby stations. Each of the considered pattern contains both positively and negatively critical stations and can be used to plan a rebalancing operation. Those patterns model recurrent critical situations in the analysed data. Based on the patterns extracted from historical data, rebalancing actions can be planned and applied to dynamically manage critical stations status.

For instance, suppose that during the fist phase the frequent pattern $+s_4, +s_7 \rightarrow -s_{11}, sup = 60\%, conf = 80\%$ is mined. This rule highlights a frequent critical unbalanced behaviour and can be used to plan at time t_{new} a rebalancing operation among the nearby stations s_4, s_7, and s_{11} moving bicycles from stations s_4 and s_7 to s_{11}, supposing s_{11} is negatively critical while s_4 and s_7 are positively critical at time t_{new}.

Since different time periods are characterized by different behaviours, the proposed methodology trains and applies a set of contextualised models, each one tailored for a specific time period.

4.1. Pattern-Based Model Training

Given the input dataset $\mathcal{D}$ containing information about the historical stations' statuses and the set $\mathcal{S}$ containing the stations and their geographical locations, the pattern-based model training phase (depicted in Figure 1) is based on the following steps:

Neighbourhood identification. Given the stations' geographical locations and the neighbourhood radius d, the neighbourhood $N(s)$ for each station $s \in \mathcal{S}$ is computed.

Occupation rate computation. Given the input dataset $\mathcal{D}$ and the neighbourhood $N(s)$ for each station $s \in \mathcal{S}$, the occupation rate $OR_s(t)$ for all stations $s \in \mathcal{S}$ and all timestamps $t \in \mathcal{D}$ is computed, i.e., the occupation is computed for each pair $(s, t) \in \mathcal{D}$.

Identification of critical stations. Given the criticality threshold cr, the occupation rate for each pair $(s, t) \in \mathcal{D}$ and the identified neighbourhoods, the critical rate for all the pairs (s, t) is computed. Then, only the pairs (s, t) associated with either a positively or a negatively critical situations are selected and stored in the dataset $\mathcal{D}_{\mathcal{CR}}$, enriched with the critical status (positive or negative).

Contextualised data partitioning. Given a contextualised partitioning schema based on timestamp, $\mathcal{D}_{\mathcal{CR}}$ is split into N non-overlapping partitions P_i. A partition P_i is a logical group defined on input data, related to a specific temporal context, on which we are interested in training a tailored model, e.g., if we are interested in a contextualised model for each of day of the week, $\mathcal{D}_{\mathcal{CR}}$ is split in seven partitions (one for each day of the week).

Generation of transactional datasets. Given the partitions $\mathcal{P}_i$, a transactional dataset $\mathcal{TR}_i$ that encodes the critical stations in each timestamp t is built from $\mathcal{P}_i$. Each transaction $tr_t \in \mathcal{TR}_i$ includes the set of stations that are positively or negatively critical at timestamp t and their status (positive or negative).

Rule extraction. Finally, the association rules are mined from each transactional dataset $\mathcal{TR}_i$ to extract for each context the set of frequent patterns $\mathcal{R}_i$ representing recurrent critical situations among nearby stations.

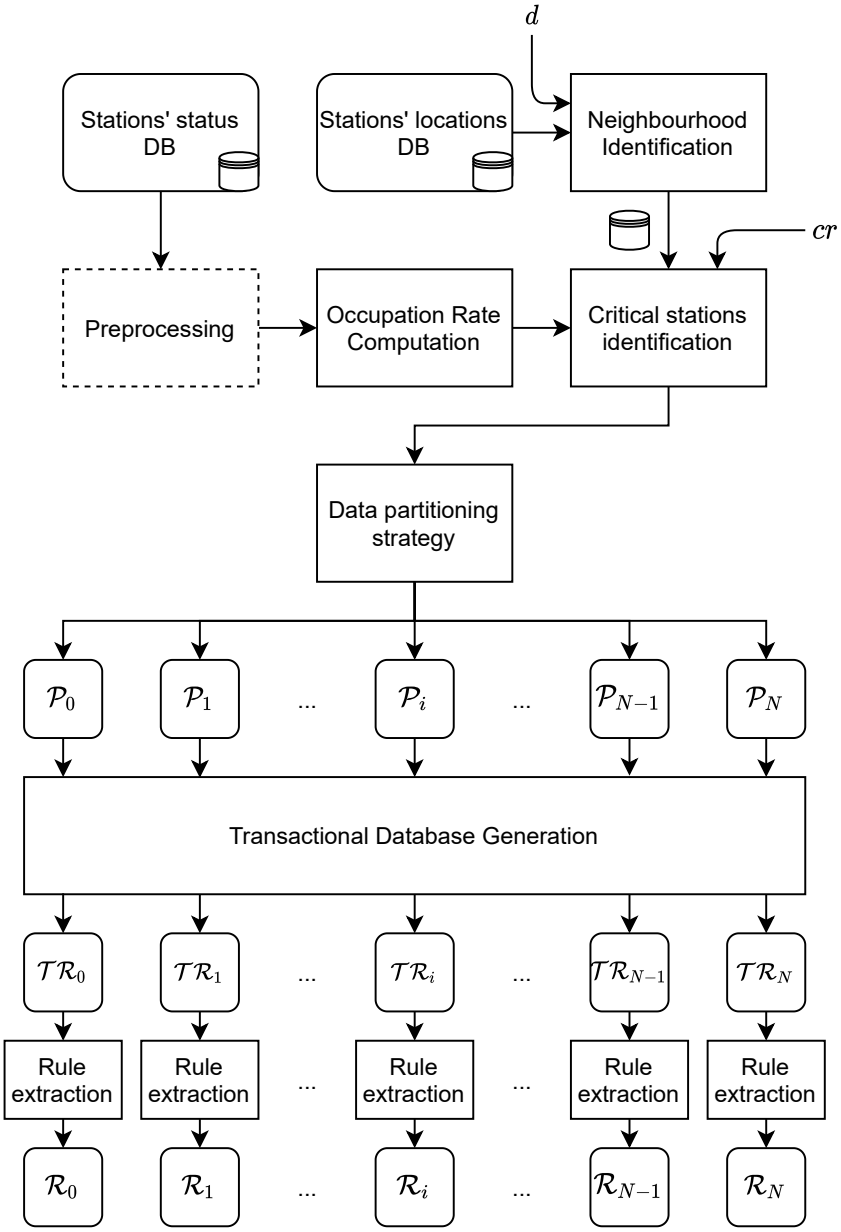


Figure 1. Framework structure at training time.

The set of extracted association rules represents the training model that is used to plan online rebalancing actions using the approach described in Section 4.2.

In the following subsections we provide more details about the contextualised data partition step and the rule mining one, which are the building blocks of the proposed methodology.

4.1.1. Contextualised Data Partitioning and Models

The rebalancing framework proposes different contextualised data partitioning strategies to extract meaningful insights at different time partitioning groups and to let the framework adapt according to different habits of users and contexts. Each partition includes the data used to train a tailored model for a specific temporal context. The proposed contextualised partition-based approach allows to model more precisely the characteristics

of each context and hence plan more effective rebalancing strategies, as shown in the experimental section.

The data partitioning component of our framework receives an input dataset $\mathcal{D}_{\mathcal{CR}}$ with the timestamp t information associated to each record and divides the dataset into N non-overlapping partitions $\mathcal{P}_i$, based on the value of t, such that

$$\bigcup_i \mathcal{P}_i = \mathcal{D}$$

It is useful to notice that, given a contextualised temporal partition strategy, a specific timestamp t belongs to only one single partition $\mathcal{P}_i$.

Different partitioning strategies lead to different frequent patterns and different contextualised models. Using different time partitioning strategies, our framework is able to collect meaningful information about users' usage patterns in different moments of the day, week, and month and allows us to understand how the usage behaviour changes in different temporal contexts. Considering different partitioning strategies, we can understand if finer or coarser temporal contexts should be used in the bike sharing domain. Finer partitioning strategies should provide more tailored and precise models but overfitting could occur with a higher probability. Conversely, coarser partitioning strategies avoid overfitting but could be too general and do not perform well for all contexts.

We proposed and evaluated the following temporal contextualised data partitioning strategies:

Per month partitioning. Data belonging to the same month are kept together and monthly models are trained.

Per day of the week partitioning. Data belonging to the same day of the week are included in the same partition and analysed together. By doing this, the mined patterns are able to get insights about critical stations within the same day of the week. A total number of seven groups $\mathcal{P}_1, ..., \mathcal{P}_7$ are generated.

Per time slot partitioning. Three timeslots are defined: 5:00–13:00, 13:00–21:00, and 21:00–05:00. One partition for each timeslot is defined. Association rules in this case gather insights about frequent critical stations in certain time slots of the day, independently of the day of the week.

Per day of the week and time slot partitioning. This approach combines together the latter two partitioning approaches, defining one partition for each combination (timeslot, day of the week).

We consider the time information the most relevant context in this domain and for this reason we decided to partition data based on the timestamp dimension. However, the proposed methodology can be easily adapted also to other contextual dimensions.

4.1.2. Transactional Dataset Generation and Rule Extraction

The proposed methodology uses the association rules mined from historical data as a model of recurrent behaviours to plan the rebalancing operations. However, since the itemset and association rule mining algorithms operate on transactional datasets, the original data needs to be manipulated to obtain the input in a transactional format. Each partition $\mathcal{P}_i$ must be mapped to a transactional dataset $\mathcal{TR}_i$. Specifically, for each partition $\mathcal{P}_i$ and for each distinct timestamp t in $\mathcal{P}_i$, we generate a transaction tr_t that is stored in the transactional dataset $\mathcal{TR}_i$. The transaction tr_t contains all the stations that are in a critical status at time t and their critical status. In particular, we associate the plus sign $(+)$ to station s if the station is positively critical at time t and the minus sign $(-)$ if it is negatively critical. A single transaction in $\mathcal{TR}_i$ thus represents the list of positively and negatively critical stations present in partition $\mathcal{P}_i$ at timestamp t.

For instance, suppose that at timestamp t_1 the stations s_4 and s_7 are positively critical, station s_{11} is negatively critical, and all the other stations in $\mathcal{S}$ are not in a critical status. It follows that the following transaction $tr_{t_1} = \{+s_4, +s_7, -s_{11}\}$ will be inserted in the transactional dataset.

Given the N transactional datasets, the FP-growth algorithm is used to extract frequent itemsets and the set of association rules, given a minimum support threshold *minSupport* and a minimum confidence threshold *minConfidence*. Specifically, a set $\mathcal{R}_i$ of association rules is extracted from each transactional dataset $\mathcal{TR}_i$, i.e., a set of contextualised rules is mined for each (temporal) context.

The extracted rules are characterized by a set of stations, with the associated critical status, in the antecedent and one single station, with its critical status, in the consequent. Some examples of neighbourhoods and mined rules are reported in Tables 1 and 2.

Among the extracted rules, only a subset of them can be used to plan local rebalancing operations. Specifically, only the rules (i) that contain only stations belonging to the same neighbourhood and (ii) such that the critical status of the stations in the antecedent of the rule is of opposite sign with respect to the critical status of the station in the consequent of the rule are useful for planning local rebalancing operations. We refer to the rules satisfying the second constraint as discordant rules. The interesting rules must contain only nearby stations because, as we introduced in the problem statement, we want to apply only local rebalancing operations. Moreover, they must be discordant because only if the critical status of the stations is opposite we can plan to move bicycles from positively critical stations to the negatively critical ones to remove the critical situations.

Let us consider the extracted rules reported in Table 2. Rule 2 and 5 are the only discordant rules. Rule 2 has support 60% and confidence 80% and it can be interpreted as "the combination $\{+s_4, +s_7, -s_{11}\}$ is present in 60% of the input transactions and in 80% of the cases when s_4 and s_7 are both positively critical, it follows that s_{11} is negatively critical". Stations s_4, s_7, s_{11} belong to neighbourhood 2 (Table 1). Hence, Rule 2 satisfies also the first condition and it can be used to plan a local rebalancing operation. Rule 5 is very similar to Rule 2, but it is discarded by our approach because a neighbourhood containing the stations s_1, s_4, s_7, s_{11} does not exist. All the other example rules are discarded because they are not discordant rules.

Table 1. Example of neighbourhoods definition.

Id	Neighbourhood
1	$s_{45}, s_{22}, s_6, s_{87}$
2	s_4, s_7, s_{11}, s_{91}
3	s_{103}, s_1, s_{47}
4	s_{31}, s_{72}, s_{134}
5	$s_{10}, s_{40}, s_{91}, s_{52}$

Table 2. Example of extracted association rules.

#	Extracted Rules	Support	Confidence
1	$+s_6, +s_{45}, +s_{22} \rightarrow +s_{87}$	30%	73%
2	$+s_4, +s_7 \rightarrow -s_{11}$	60%	80%
3	$+s_{103}, +s_{47}, +s_{31}, +s_{72} \rightarrow +s_{134}$	23%	90%
4	$-s_{10}, -s_{40}, -s_{91} \rightarrow -s_{52}$	82%	50%
5	$+s_4, +s_7, +s_1 \rightarrow -s_{11}$	31%	60%

4.2. Planning of Rebalancing Operations by Means of Association Rules

In this section, we describe how the mined association rules are used to plan at a given time t_{new} the rebalancing actions to apply in the next t_{reb} time. This phase is composed of two steps: (i) identification of critical stations at time t_{new} and (ii) definition of the rebalancing operations needed to address the identified critical situations.

Let $\mathcal{B}$ be a batch of records containing the (new) online information about the status of all the stations of the system at timestamp t_{new}. $\mathcal{B}$ is gathered using a streaming real-time system. First, for each station $s_i \in \mathcal{B}$, its critical rate at timestamp t_{new} is computed and the

set of positively and negatively critical stations are identified. The set of critical stations at timestamp t_{new}, with the associated critical status, is denoted as $CS_{t_{new}}$. Each element in $CS_{t_{new}}$ is a pair (s, cs) where s is a station and cs is the critical status of s at time t_{new}.

Now we have the set of stations for which a rebalancing operation should be planned at time t_{new} and executed in at most t_{reb} time. To define the rebalancing operations we consider the association rules mined in the training phase. Let $\mathcal{R}_i$ be the sorted list of association rules extracted from partition P_i to which t_{new} is associated (e.g., if we are using the day of the week partitioning strategy, the day of the week of timestamp t_{new} is used to decide which contextualised partition of the historical data and which set of rules must be considered). For each rule $r \in \mathcal{R}_i$, we search whether all the items (i.e., all the pairs (station, critical status)) belonging to the rule are contained in $CS_{t_{new}}$. In such case, the rule r can be used to define a rebalancing operation that fixes the critical situations of the stations in r. A rule that satisfies such constraint is defined as applicable rule. Specifically, rule r represents a recurrent frequent critical pattern: by moving bikes from positively critical to negatively critical stations it is possible to fix critical situations. Movements are allowed only from stations in the antecedent to stations in the consequent or vice versa. The strength of the pattern is numerically represented by the support and confidence values of the extracted rule. For example, suppose stations s_4 and s_7 are positively critical at timestamp t_{new} while station s_{11} is negatively critical at the same timestamp. Suppose that those stations are in the same neighbourhood and the rule $+s_4, +s_7 \rightarrow -s_{11}$ has been extracted from the historical data. This rule can be used to plan a rebalancing operation that fixes the critical situations of stations s_4, s_7, and s_{11}, moving bicycles from s_4 and $s7$ to s_{11}. Since the extracted patterns are frequent, we can suppose that, with a high probability, the stations that are in a critical situation at time t_{new} will still be in the same critical status in the next t_{reb} minutes. Hence, the rebalancing operation planned at time t_{new} would be probably applicable and useful when the truck will actually reach the critical stations. Since each rule is composed solely of neighbouring stations, our methodology performs local rebalancing operations (Figure 2) such that the final occupancy rate of every station present in the rule is the same across stations in the neighbourhood. Thus, the goal of our framework is to flatten the occupancy rate of identified critical neighbourhoods by moving bicycles from positively critical to negatively critical stations or vice versa, fixing critical situations. Each bike station is visited only once during the operation.

Figure 3 shows the framework's structure at reallocation time.

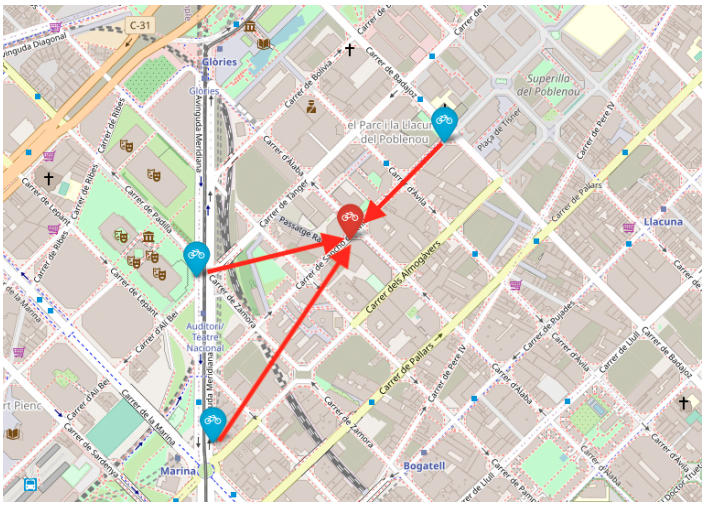


Figure 2. Example of local rebalancing. The extracted rule r contains 4 stations: 3 in the body of the rule and 1 in the head. Blue stations are positively critical; the red station is negatively critical. Once such pattern is detected in the new batch of data $\mathcal{B}$, the frameworks suggests to move bicycles from blue stations to the red stations such that the final occupancy rate of the 4 stations is the same.

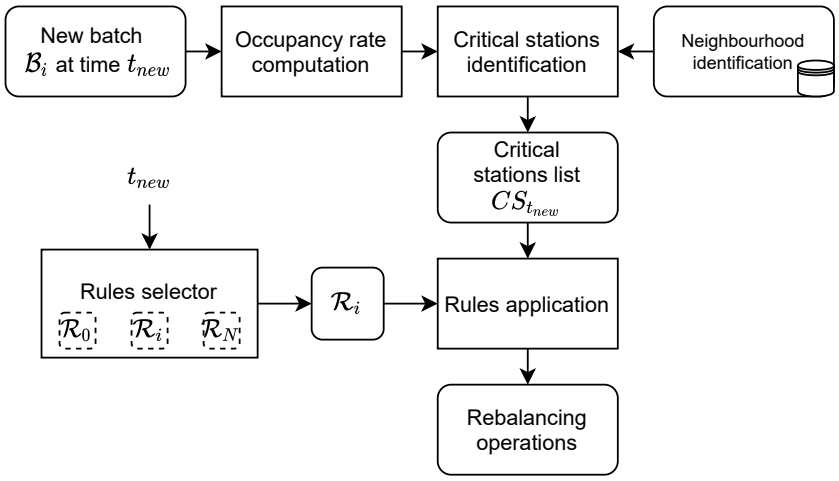


Figure 3. Framework's structure when the system is operating.

The list of applicable rules represents potential rebalancing operations that should be planned at timestamp t_{new} and executed in at most t_{reb} time to fix imbalanced situations across stations. However, due to the limited amount of trucks, operators and time, not all critical stations $CS_{t_{new}}$ can be fixed and thus a priority needs to be defined. Such priority is determined by the quality indices of the mined and applicable association rules. Specifically, we consider the applicable extracted association rules in descending order using the following quality indices: confidence, support, and length of the rule. The higher the confidence, the stronger the pattern is among the training data. Hence, given two rules, the confidence is initially considered and the one with the highest confidence is selected first. Given two rules with equal confidence, the one with higher support (i.e., the rule which is more frequent in the historical data) is considered first. In case also the support value is the same among the two considered rules, the length of the rule is considered. If also the length is the same, the lexicographical order is considered.

We decided to impose the number of rebalancing operations planned at time t_{new} to be equal to the number of trucks N_t available, i.e., among all the extracted association rules, only the top N_t applicable rules with highest priority will be applied. Moreover, since the trucks used for the actual rebalance require time to travel from the deposit to critical neighbourhoods, we distinguish between the operation planning time t_{new} and the actual rebalancing operation time, namely the reallocation time, which must be at most t_{reb} minutes after t_{new}. Specifically, we define as the operation planning time the time t_{new} in which the system analyses the online data $\mathcal{B}$ collected by the stations, computes the occupancy rate and performs the match operations between the extracted rules and identified critical stations to select the top N_t applicable rules. During the bicycles reallocation, due to the system being online, the status of the bike stations involved in the rebalancing operation may change within t_{reb} minutes. For such reason, the system checks whether the planned rebalancing operation is still applicable at reallocation time (i.e., when the truck reaches the stations to rebalance) and in case it is not, the system tries to identify the largest subset of stations for which the rule still holds, i.e., identify a subset of critical stations to rebalance. If no subset is found, the rule is not applied.

Consequently, the association rules extracted at training time are used as an associative classifier to detect at test time recurrent critical neighbourhoods that need to be fixed. Given the set of rules, only the applicable ones are selected by analysing the patterns of critical stations at a new time instant t_{new}. By sorting the applicable rules with the aforementioned criterion, we prioritize certain neighbourhoods with respect to others by selecting only the top N_t rules. Then, the framework generates the set of bicycle movements that need to be performed to flatten the occupancy rate across such problematic neighbourhoods.

Considering a real scenario, it is not feasible to perform a rebalancing operation at every timestamp. Because of such reason, we propose to apply our rebalancing methodol-

ogy a limited number of times per day. The number of times and the timestamps at which such operations are planned and performed are fully configurable and can be decided by systems' administrators according to the city and the population's habits.

4.3. Parallelization, Frameworks, Hardware and Tools

Due to the highly intensive computation required by data mining and association rule mining tasks, the project's workload can be divided twofold: (1) local workloads and (2) distributed jobs. Local workloads include the neighbourhood identification, application of contextualized data partitioning and rule application to assess the performances of the proposed approach. Distributed workloads are represented by data cleaning and preprocessing operations, which are introduced in the experimental section, critical stations identification, transactional database generation and association rules extraction. The second set of operations are more computationally expensive and easily parallelizable using a big data framework such as Spark.

All the local workloads are performed using python and pandas [48,49]. Instead, all the distributed computing were performed using Apache Spark [50] 2.4 on the BigData@PoliTO cluster [51] managed by the SmartData center. Local workloads were executed on a single virtual machine hosted on a master node of the cluster and thus did not take advantage of the distributed environment. The virtual machine has 4 CPU cores and 16 GB of RAM. The distributed cluster has a total of 53 worker nodes equipped with 2 CPUs each, either with Intel Xeon or AMD EPYC processors. The amount of RAM available per node ranges from 96 GB to 512 GB. For more information about the computing facilities, readers can refer to [51].

5. Experiments and Results

In this section, the experimental setup is described, starting from the characteristics of the data used in this paper. Then, the performed experiments and the achieved results are reported and discussed. We analysed the effect of all the parameters on the system and how they affect the rebalancing operations in terms of number of critical stations fixed.

This section describes the results of the following analyses:

- The dataset used to test the developed framework is described and analysed, together with the preprocessing steps made to clean the data.
- The effect of different input parameters are evaluated to properly define the preprocessing pipeline for the data cleaning process. In particular, we analysed the effects of the thresholds f and *varianceThreshold*.
- The effects of the framework's parameters on the rule extraction process are analysed.
- The effects of all the framework's parameters and the number of available vehicles N_t on the rebalancing process are analysed.

For these analyses, all the data partitioning approaches introduced in Section 4.1.1 are considered and compared to determine the most prominent approach for the analysed dataset.

5.1. Dataset Description

The dataset used for the experiments in this paper is the Barcelona Bike Sharing dataset, which contains information about stations' location and their status at a specific point in time. More specifically, the dataset consists of two separate tables. The first table stores information about the ID and geographical location, expressed in terms of longitude and latitude, of each station of the bike sharing system. The second table contains the following columns: station ID, timestamp, used slots, and free slots. The latter can be interpreted as a table containing the logged information for each station at regular time intervals of two minutes: at a specific point in time and for a certain station, the number of used slots (i.e., bikes available for new trips) and the number of free slots (i.e., slots available to park a rented bike) is reported. A sample of the content of the two tables is shown in Tables 3 and 4 respectively. Station information is available from 15 May 2008 to 30 September 2008. The dataset was split into train and test according to the following

criterion: the last 7 days of each month were assigned to the test dataset, the remaining days constitute the train dataset and are used to extract association rules. The test set is used to evaluate the quality of the proposed methodology on unseen data.

Table 3. Example of stations' information.

StationId	Longitude	Latitude	Name
1	2.180019	41.397978	Gran Via Corts Catalanes
2	2.176414	41.394381	Plaça de Tetuan
3	2.181164	41.393750	Ali Bei

Table 4. Random sample of stations' status extracted from the dataset.

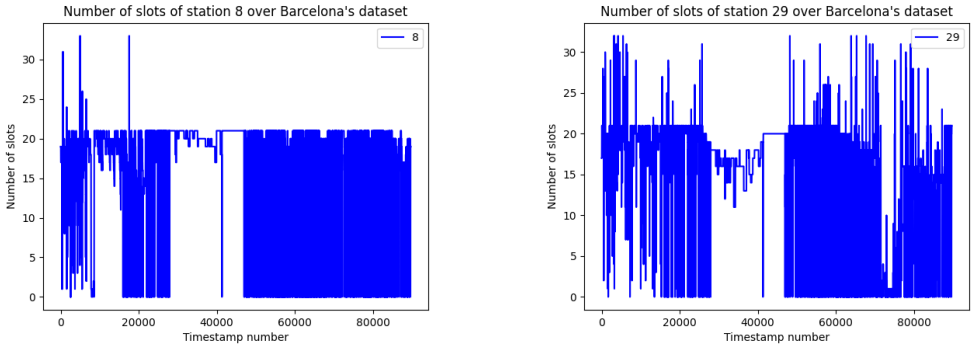
StationId	Timestamp	Used Slots	Free Slots
280	2008-08-24 21:44:00	8	19
223	2008-09-25 04:52:00	5	22
108	2008-06-16 17:20:00	3	24
67	2008-06-26 10:24:00	5	16

5.2. Preprocessing and Data Cleaning

The Barcelona Bike Sharing dataset is a noisy dataset with some anomalies and thus preprocessing is necessary to remove noise. In particular, analysing the stations' status over time, it is possible to notice irregularities among the total number of slots for certain stations. In fact, high oscillations were present in the dataset, having the station a variable amount of available slots, ranging from 0 to a maximum value of 59. Moreover, some information is missing because not all the stations were able to log their status every two minutes. Hence, the following points were addressed by our preprocessing step:

- Removal of data associated to total number of slots for a specific station equal to 0 or 1. Such situations can possibly be associated with the system not working properly or maintenance operations ongoing during the time the data was collected.
- Removal of some bike stations from the dataset due to them being logged rarely. In particular, we removed stations that were not present in at least f% of the available timestamps, being f an input parameter for our framework. We refer to such stations as infrequent stations.
- Removal of some bike sharing stations due to high fluctuation of total number of slots. Specifically, we analysed the variance of the total number of slots for each station and filtered out stations whose variance was higher than a specific threshold denominated as *varianceThreshold*. We refer to such stations as unstable stations.

As representative examples of unstable stations, Figures 4a and 4b show the total number of slots of station 8 and station 29 over the entire time span available, respectively. The total number of slots oscillates between 0 and more than 30 confirming the instability of the information associated with those two representative unstable stations. Unstable stations must be removed in order to extract robust patterns not affected by noisy input data.



(**a**) Station 8. (**b**) Station 29.

Figure 4. Total number of slots of stations over time.

After the removal of the data points associated with a total number of slots equal to 0 and 1, we analyse the impacts of the f and *varianceThreshold* parameters on the cleaning operations. Hence, hereafter we discuss the effects of every parameter on the dataset and the final outcome on the prediction and rebalancing phase, starting from the cleaning operations.

5.3. Analyses of Parameters' Impact on Data Cleaning Operations

Considering all the input data except for those data points associated with total number of slots $\leq$1, we evaluate the number of filtered stations based on the thresholds f and *varianceThreshold*. Table 5 lists the values considered for f and *varianceThreshold*.

Table 5. Parameter list for f and *varianceThreshold*.

Parameter	Values
f	0.1, 0.8, 0.85, 0.9
varianceThreshold	3, 5, 7

5.3.1. Effect of the Frequency Threshold

The effect of the frequency threshold f is limited due to the fact that most of the stations correctly log their status. Given a total number of 284 stations, Table 6 reports the number of infrequent stations (i.e., the stations with many missing data) given the threshold parameter f and a *varianceThreshold* $= 5$ (this is the default value for *varianceThreshold*).

Table 6. Number of filtered infrequent stations with variable f and fixed *varianceThreshold* $= 5$.

	***f* = 0.1**	***f* = 0.8**	***f* = 0.85**	***f* = 0.9**
# infrequent stations	1	4	8	12

Therefore the effect of the frequency threshold is limited. As a consequence of this analysis, we set the frequency threshold f to 0.9 for our framework, which means that only the stations for which their status is available in less than 90% of the timestamps are discarded. In other words, at most 10% of missing log data are allowed for each of the considered station.

5.3.2. Effect of the Variance Threshold

The variance threshold is used to filter unstable stations from the dataset, i.e., stations with an unstable total number of slots over time. The variance parameter is computed for each station on the total number of slots. The number of unstable stations filtered using a fixed value for f of 0.9 is shown in Table 7.

Table 7. Number of filtered unstable stations with variable *varianceThreshold* and fixed f.

	***varianceThreshold* = 3**	***varianceThreshold* = 5**	***varianceThreshold* = 7**
# unstable stations	232	155	114

Setting the threshold to 3, the amount of stations identified as unstable is high, thus reducing the analysis to very few stations and, consequently, limiting the amount of data on which the pattern extraction process is performed. On the contrary, using $varianceThreshold = 7$, an excessive amount of stations with high variability on the total number of slots is present in the dataset. In conclusion, as a good compromise between quantity of data and limited variability of total number of slots, we choose $varianceThreshold = 5$.

In the following section we discuss the effects of neighbourhood radius d and critical threshold cr in the problem definition, since they both influence the way critical stations are determined, as well as the influence of the minimum support and confidence thresholds on the number of extracted association rules.

5.4. Impact of Parameters on Rule Extraction

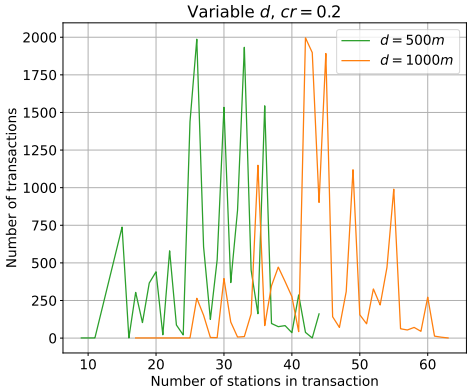
The outcome and the quality of rule extraction process depend on the distance d used to define the stations' neighbourhood, on the critical threshold cr used to mark a station as critical, and on *minSupport* and *minConfidence* thresholds. Whereas the first two parameters are intrinsic of the problem itself, defining the way the stations are critical and the operational range of the rebalancing operations, the latter two parameters are related to the association rule mining algorithm and define the minimum strength of the patterns to be mined. All rules extraction in this subsection are performed by considering Mondays of June using the per day partitioning approach.

We set the neighbourhood radius empirically to 500 m and 1 km, considering it a reasonable distance for users to be travelled on foot/by bicycle in a reasonable amount of time in case the designated station for pick-up is empty or the one designated for drop is full. Moreover, by doing so, the operators in charge of the actual rebalance need a short amount of time to travel between bike stations to move bicycles.

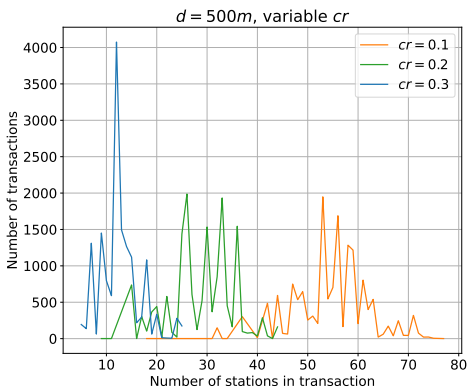
We analysed the performance of the framework with critical threshold cr values of 0.1, 0.2 and 0.3. The higher the cr value, the smaller the number of critical situations because, by increasing cr, a higher absolute difference between the occupancy level of a station and the average occupancy rate of its neighbourhood is needed to highlight a critical status. Considering that almost all stations have a total number of slots in range 25–30, lower or higher threshold values are not meaningful for the detection of critical situations. Too many stations will become critical even if the difference in terms of number of used slots is only one or two slots if cr values lower than 0.1 are considered. Conversely, few or no critical situations would be identified if higher values of cr are considered.

The impact of these two parameters can be evaluated using two metrics: number of critical stations identified per timestamp and number of rules extracted by the framework. Figure 5a represents on the horizontal axis the amount of critical stations and on the vertical axis the amount of timestamps having that number of critical stations. The two curves represent two different values of neighbourhood radius values d with fixed critical threshold cr of 0.2. As can be seen, increasing the radius shifts the curve to the right, with higher amount of critical stations. In fact, by increasing the radius d also the cardinality of the neighbourhood increases and consequently the probability that at least one station has significantly more or less used slots with respect to its neighbours.

Figure 5b uses the same axis notation to show the impact of the critical threshold cr on the number of critical stations per timestamp. The results reported in Figure 5b were computed by setting d to 500 m. The chart shows the expected behaviour. Specifically, as the critical threshold cr shrinks, the amount of critical stations increases (stations that are critical with a threshold of 0.2 are also critical with a threshold of 0.1).



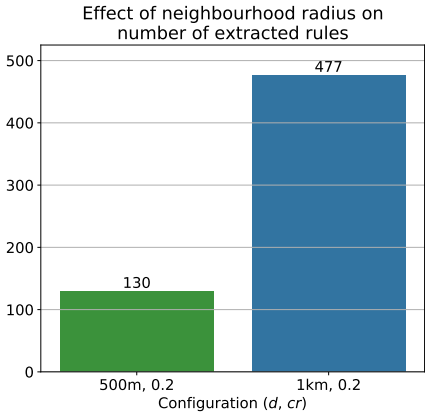
(**a**) Effect of neighbourhood radius on number of critical stations. Critical threshold is fixed at 0.2.



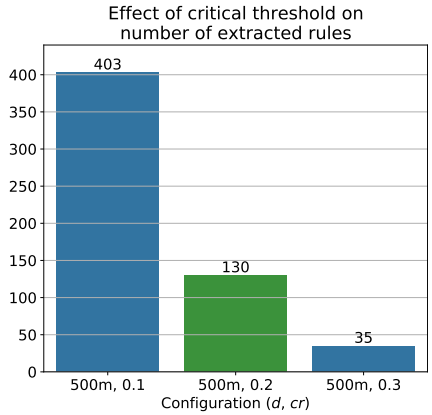
(**b**) Effect of critical threshold on number of critical stations. Neighbourhood radius is fixed at 500 m.

Figure 5. Number of critical stations with varying *d* and *cr*.

A similar analysis can be performed on the number of discordant rules extracted by the framework by varying *d* and *cr*. It is possible to notice in Figure 6a that, by increasing *d* from 500 m to 1 km, the number of extracted rules increases noticeably. A similar trend is observed independently from the applied partitioning strategy and also in Figure 6b as *cr* decreases.



(**a**) Number of extracted rules with variable neighbourhood radius.



(**b**) Number of extracted rules with variable critical threshold.

Figure 6. Number of extracted rule with variable parameters.

The aforementioned analyses about the number of extracted rules confirm the expected behaviour evidenced in Figure 5a,b. By lowering *cr* and/or increasing *d*, more critical stations are detected and more discordant patterns are mined.

The two input parameters of the association rule mining algorithm also impact on the quantity and the quality of the rules extracted. Their effects can be measured by analysing the quantity of the mined rules and the resulting quality of rebalancing operations.

The former effect is analysed in the following paragraph, whereas the latter is discussed in Section 5.5.

As denoted in [9], an exponential decay of number of extracted rules can be observed as the value of *minSupport* increases. As an example, we observed that by increasing the support threshold from 10% to 20% the number of rules drops from 120 to 40. Contrarily, variations in *minConfidence* do not determine a great reduction of number of rules, but usage of rules with extremely high values of confidence is harmful to the system since they represent specific patterns that can be applied only to few critical situations. Considering

the limits imposed by *minSupport* and *minConfidence* parameters in terms of quantity of quality of rules, we set the minimum confidence threshold to 50% and minimum support threshold to 10% as default parameters for our system.

5.5. Quality of the Rebalancing Operations

Our final goal consists of applying rebalancing operations to fix critical situations. In this section, we will analyse the quality of the proposed methodology with respect to this goal. Evaluation metrics were computed on the test set, i.e., the last 7 days of each month, whereas the remaining data was used to extract the association rules.

To evaluate the performances of the proposed approach on rebalancing operations, it is necessary to consider all the previously mentioned parameters in addition to the difference between the operation planning time and the reallocation time (which must be at most t_{reb} minutes), the data partitioning strategy and the number of trucks N_t adopted for the rebalancing operations. Table 8 contains all the range of values evaluated in our experiments for the mentioned parameters.

We performed the evaluation supposing to rebalance the stations two times per day: at 6 a.m. and at 15 p.m. Those are two moments of the day frequently associated with many critical situations. Hence, rebalancing operations at those moments of the day are crucial to provide a good quality of experience. All experiments have been performed setting t_{reb} to 1 h. Based on the topology of Barcelona, this is the average time we assume a truck takes to go to the neighbourhood of stations to rebalance.

Table 8. Parameters ranges.

Parameter	Values
data partitioning strategy	Per month partitioning, Per day of the week partitioning, Per timeslot partitioning, Per day of the week and timeslot partitioning
minSupport	10%, 20%, 30%
minConfidence	50%
N_t	5, 10, 20

First, we analyse the rebalancing results achieved by the best identified configuration for solving the dynamic rebalancing problem, namely the reference configuration. Second, we analyse the effect of each parameter on the performance of the entire system in terms of number of fixed critical stations and average number of fixed stations per rebalancing operation. In these analyses, we considered the absolute number of fixed critical stations and the number of fixed stations per rebalancing operation, which represent the grade of efficiency of the system when performing corrective actions. The reference configuration is used to suggest a good configuration to be used and for comparison purposes to understand the effect of each of the parameters in taking decisions.

We initially performed the experiments setting d to 500 m and cr to 0.2. These two parameters, as we discussed before, are not associated with the configuration of the proposed methodology. They are two characteristics of the problem. We will analyse their impact on the complexity of the problem later.

5.5.1. Best Performing Configuration

Among all the possible configurations, the chosen reference configuration is the following:

- $minSupport = 10\%$;
- $minConfidence = 50\%$;
- $N_t = 5$;
- *data partitioning strategy* = Per day of the week partitioning.

The default configuration achieves the best result in terms of fixed critical stations/critical status. Specifically, it was able to fix 225 critical station status out of 1377 by performing 151

rebalancing operations. Thus, on average, each rebalancing operation fixed 1.49 stations in a critical status.

5.5.2. Effect of the Contextualised Data Partitioning Strategy

The results obtained varying the data partitioning strategy, and the default configuration for the other parameters, are shown in Figures 7 and 8.

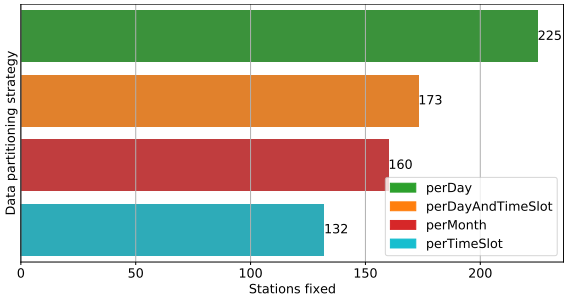


Figure 7. Reference configuration performances in terms of absolute number of station fixed on different data partitioning approaches.

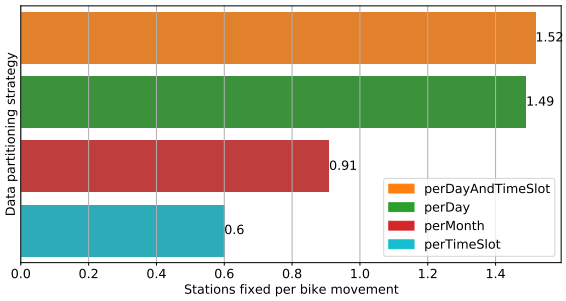


Figure 8. Reference configuration performances in terms of fixed stations per movement on different data partitioning approaches.

Figure 7 shows that the per day of the week partitioning strategy is the best one in terms of absolute number of fixed stations and is the second best one when considering the stations fixed per rebalancing operation metric (see Figure 8). The per day of the week and timeslot partitioning strategy is slightly better than the per day of the week one in terms of stations fixed per rebalancing operation. However, it is significantly worse in terms of total number of fixed stations (173 vs. 225 fixed critical station status). The rules mined considering the per day of the week partition strategy are more general and robust with respect to those mined using the per day of the week and timeslot partitioning strategy. Hence, in the majority of the cases the operations planned at time t_{new} are still useful and can be applied at the reallocation time $t_{new} + t_{reb}$. Differently, when other contextualised partition strategies are considered, the operations planned at time t_{new} are more frequently no longer applicable at time $t_{new} + t_{reb}$ because they are too much specific (per timeslot, per day of the week and timeslot) or too much general (per month partitioning).

5.5.3. Effect of Minimum Support

As mentioned in Section 5.4, a small increase of *minSupport* is likely to cause a significant decrease in the number of rules mined and, consequently, decrease the performance and quality of the rebalancing methodology we proposed because fewer rebalancing operations can be planned. This trend is shown in Figure 9. As the minimum support threshold increases, the total number of fixed stations drops in all the considered data partitioning approaches. It was observed that a small drop of fixed stations, for example in the per day of the week and timeslot strategy, was due to the fact that many of the first rules in the ranking applied by the proposed methodology are also the most frequent ones, which are therefore not affected by the increasing value of *minSupport*. This consequence is confirmed

by Figure 10, in which the fixed stations per movement is plotted. A reduction in the number of rules enforces the framework to apply rules with higher support, which can be associated with stronger patterns. We underline also the fact that the per day of the week and per day of the week and timeslot approaches are the most effective ones. Even though increasing the minimum support threshold increases the quality in terms of fixed stations per rebalancing operation, we opted to prioritize the absolute total number of fixed stations to maintain overall a better quality of service.

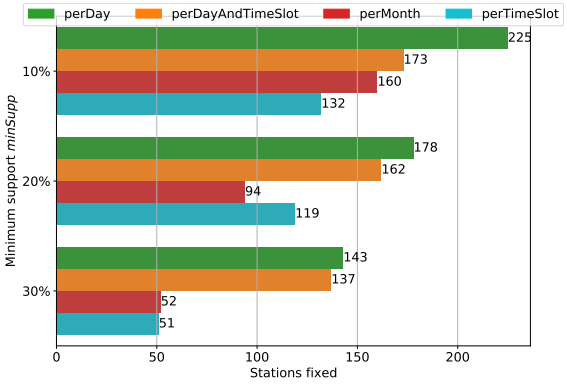


Figure 9. Number of fixed stations with varying *minSupport*.

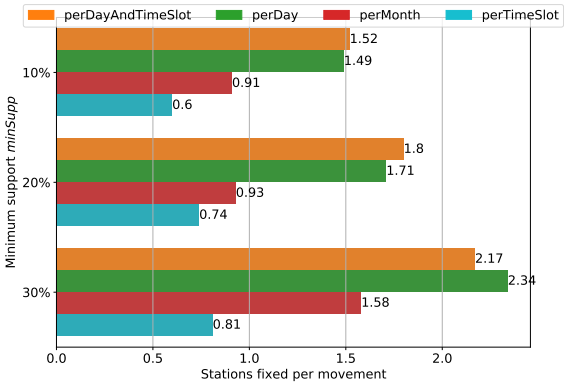


Figure 10. Number of fixed stations per bike movement with varying *minSupport*.

5.5.4. Effect of Number of Trucks N_t

In our analyses we imposed the constraint that a single truck is used to rebalance a single neighbourhood, i.e., to apply a single rebalancing operation. Figures 11 and 12 show the performances obtained for 3 different values of N_t.

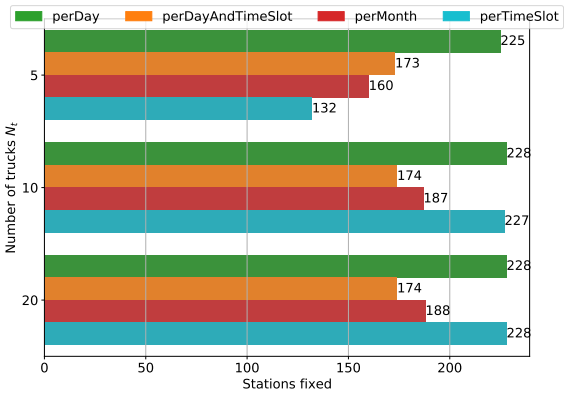


Figure 11. Number of fixed stations with varying N_t.

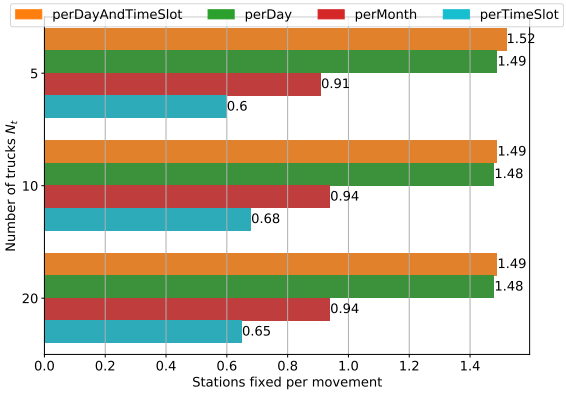


Figure 12. Number of fixed stations per bike movement with varying N_t.

As the value of N_t changes, the average value of fixed stations per movement remain almost stable. This means that the top N_t rules used to plan the N_t rebalancing operations assigned to the N_t trucks (one rebalancing operation per truck) are characterized by a similar rebalancing quality.

In this case, the trend of the total number of fixed critical stations is dependent on the data partitioning strategy. Those strategies that have worse performances in terms of total number of fixed stations when five trucks are considered are the ones that are more positively affected by the increase of the number of trucks and planned rebalancing operations. To understand this trend, we analysed the top N_t rules in the ranking for each partitioning strategy and we noticed that some strategies (e.g., the per timeslot partitioning one) have a higher number of complementary association rules among the top rules related to different set of nearby stations. This means that the selected top N_t rules are related to rebalancing operations that impact on different neighbourhoods, increasing the overall number of fixed critical stations. Conversely, the top N_t rules in the ranking when the per day of the week or the per day of the week and timeslot strategies are applied are significantly overlapped and many of them are related to subsets of the same neighbourhood. It implies that similar redundant rebalancing operations should be simultaneously assigned to more trucks, reducing the overall benefit of increasing the number of trucks. This result suggests that an analysis of the overlaps among rules could be beneficial to further improve the quality of our methodology.

Given the small difference in terms of absolute performance (from 225 to 228), we chose to keep $N_t = 5$ in the reference configuration to limit the number of trucks used and hence the costs.

5.5.5. Effect of Neighbourhood Radius d and Critical Threshold cr

The neighbourhood radius d and the critical threshold cr define the problem itself and its complexity. Changing either one or both of them results in a different number of critical stations. Consequently, to properly compare approaches with differing d and cr values we compared the percentage of fixed stations instead of the absolute number of fixed stations. While an increase in d can easily justify the lower performances achieved due to the fact that a greater radius may induce new critical situations, no qualitative explanation can be provided for cr. Considering the results shown in Figures 13 and 14, we chose $d = 500$m and $cr = 0.2$.

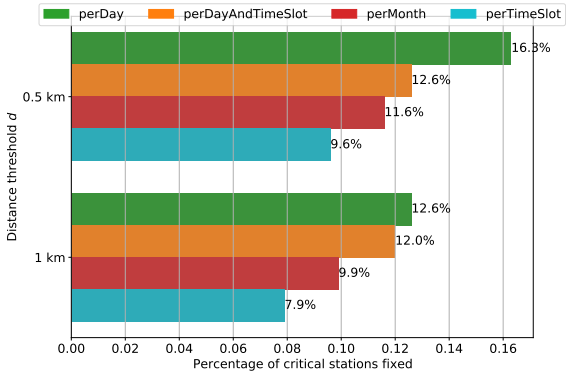


Figure 13. Percentage of fixed stations with 500 m and 1 km neighbourhood radius.

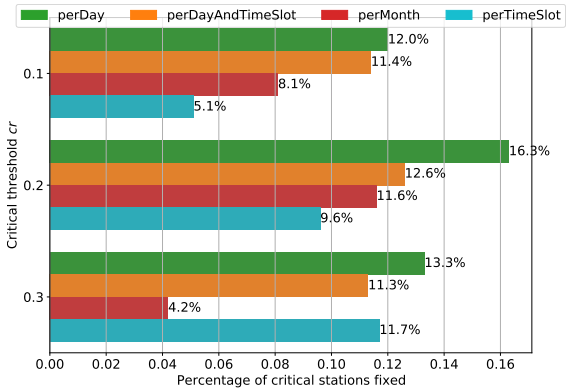


Figure 14. Percentage of fixed station varying *cr*.

6. Discussion and Conclusions

In this paper, a rebalancing framework for the dynamic bike sharing problem was presented, leveraging on data mining techniques applied on past data. The approach is configurable both in terms of problem definition, i.e., neighbourhood radius and critical threshold, and the data mining process itself, assuring diverse degrees of versatility. To solve the problem, we provided the definition of critical stations and identified frequent critical neighbourhoods by means of association rules. Depending on the city and the system to which the framework is applied on, we provide four different data partitioning approaches to extract meaningful patterns from past usage data, enabling further degree of flexibility and contextualization. The rebalancing process takes into account the past system's information and users' frequent patterns by leveraging on the extracted association rules and performs a configurable amount of rebalancing operations at different time of the day. The framework was validated on Barcelona bike sharing data. Furthermore, the outputs generated by the proposed framework and the obtained rebalancing policies are fully interpretable by domain experts and administrators of bike sharing systems. Each rebalancing operation scheduled by the framework is supported by an association rule with a given support and confidence, underlining the strength of the pattern. Consequently, interpretation does not require any knowledge in data mining algorithms. The main drawbacks of the proposed approach are the difficulty of (i) learning new patterns incrementally as the system receives new data, (ii) handling and rebalance critical situations caused by special events (e.g., football matches), (iii) the fact that a fixed rule ordering criterion pre-determined by confidence and support, computed at training phase, may cause the framework to constantly rebalance few, highly frequent critical neighbourhoods and (iv) limited predictive capabilities of critical stations caused by the absence of temporal sequences, especially in case of long-term predictions.

As future works, we plan to incorporate spatio-temporal sequence mining into our methodology by leveraging on sequence and graph mining algorithms, such as [52,53]. In

particular, by analysing frequent occurrences of critical stations throughout the city in the temporal dimension, we are interested in identifying temporal sequences of critical stations and plan the rebalancing policy according to medium and long-term predictions. Such approach would allow for timely identification of imbalanced stations and compensate the lack or excess of bicycles before the occurrence of critical situations.

Author Contributions: Conceptualization, M.C. and P.G.; methodology, M.C. and P.G.; software, M.C.; investigation, M.C., P.G., and L.C.; writing—original draft preparation, L.C. and P.G.; writing—review and editing, P.G. and L.C.; visualization, M.C. and L.C.; supervision, P.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The research leading to these results is supported by the SmartData@PoliTO center for data analysis and Big Data technologies.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Martens, K. The bicycle as a feedering mode: experiences from three European countries. *Transp. Res. Part D Transp. Environ.* **2004**, *9*, 281–294.
2. Dell, M.; Iori, M.; Novellani, S.; Stützle, T. A destroy and repair algorithm for the bike sharing rebalancing problem. *Comput. Oper. Res.* **2016**, *71*, 149–162.
3. DeMaio, P. Bike-sharing: History, impacts, models of provision, and future. *J. Public Transp.* **2009**, *12*, 3.
4. Eren, E.; Uz, V.E. A review on bike-sharing: The factors affecting bike-sharing demand. *Sustain. Cities Soc.* **2020**, *54*, 101882.
5. Zhang, Y.; Mi, Z. Environmental benefits of bike sharing: A big data-based analysis. *Appl. Energy* **2018**, *220*, 296–301.
6. Qiu, L.Y.; He, L.Y. Bike sharing and the economy, the environment, and health-related externalities. *Sustainability* **2018**, *10*, 1145.
7. Otero, I.; Nieuwenhuijsen, M.; Rojas-Rueda, D. Health impacts of bike sharing systems in Europe. *Environ. Int.* **2018**, *115*, 387–394.
8. Sun, F.; Chen, P.; Jiao, J. Promoting public bike-sharing: A lesson from the unsuccessful Pronto system. *Transp. Res. Part D Transp. Environ.* **2018**, *63*, 533–547.
9. Han, J.; Pei, J.; Yin, Y. Mining frequent patterns without candidate generation. *ACM Sigmod Rec.* **2000**, *29*, 1–12.
10. Kaltenbrunner, A.; Meza, R.; Grivolla, J.; Codina, J.; Banchs, R. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive Mob. Comput.* **2010**, *6*, 455–466.
11. Dell'Amico, M.; Hadjicostantinou, E.; Iori, M.; Novellani, S. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega* **2014**, *45*, 7–19.
12. Karp, R.M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*; Springer: Boston, MA, USA, 1972; pp. 85–103.
13. Hoffman, K.L.; Padberg, M.; Rinaldi, G., Traveling Salesman Problem. In *Encyclopedia of Operations Research and Management Science*; Springer US: Boston, MA, USA , 2013; pp. 1573–1578, doi:10.1007/978-1-4419-1153-7_1068.
14. Flood, M.M. The traveling-salesman problem. *Oper. Res.* **1956**, *4*, 61–75.
15. Toth, P.; Vigo, D. *The Vehicle Routing Problem*; SIAM: Philadelphia, PA, USA, 2002.
16. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91.
17. Savelsbergh, M.W.; Sol, M. The general pickup and delivery problem. *Transp. Sci.* **1995**, *29*, 17–29.
18. Berbeglia, G.; Cordeau, J.F.; Gribkovskaia, I.; Laporte, G. Static pickup and delivery problems: a classification scheme and survey. *Top* **2007**, *15*, 1–31.
19. Padberg, M.; Rinaldi, G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **1991**, *33*, 60–100.
20. Clarke, G.; Wright, J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **1964**, *12*, 568–581.
21. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472.
22. Ren, Y.; Meng, L.; Zhao, F.; Zhang, C.; Guo, H.; Tian, Y.; Tong, W.; Sutherland, J.W. An improved general variable neighborhood search for a static bike-sharing rebalancing problem considering the depot inventory. *Expert Syst. Appl.* **2020**, *160*, 113752.
23. Dell'Amico, M.; Iori, M.; Novellani, S.; Subramanian, A. The bike sharing rebalancing problem with stochastic demands. *Transp. Res. Part B Methodol.* **2018**, *118*, 362–380.

24. Gendreau, M.; Jabali, O.; Rei, W. 50th anniversary invited article—Future research directions in stochastic vehicle routing. *Transp. Sci.* **2016**, *50*, 1163–1173.
25. Chemla, D.; Meunier, F.; Calvo, R.W. Bike sharing systems: Solving the static rebalancing problem. *Discret. Optim.* **2013**, *10*, 120–146.
26. Erdoğan, G.; Battarra, M.; Calvo, R.W. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *Eur. J. Oper. Res.* **2015**, *245*, 667–679.
27. Cruz, F.; Subramanian, A.; Bruck, B.P.; Iori, M. A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Comput. Oper. Res.* **2017**, *79*, 19–33.
28. Benchimol, M.; Benchimol, P.; Chappert, B.; De La Taille, A.; Laroche, F.; Meunier, F.; Robinet, L. Balancing the stations of a self service "bike hire" system. *RAIRO Oper. Res.* **2011**, *45*, 37–61.
29. Chalasani, P.; Motwani, R. Approximating capacitated routing and delivery problems. *SIAM J. Comput.* **1999**, *28*, 2133–2149.
30. Schuijbroek, J.; Hampshire, R.C.; Van Hoeve, W.J. Inventory rebalancing and vehicle routing in bike sharing systems. *Eur. J. Oper. Res.* **2017**, *257*, 992–1004.
31. Contardo, C.; Morency, C.; Rousseau, L.M. *Balancing a Dynamic Public Bike-Sharing System*; Cirrelt: Montreal, QC , Canada, 2012; Volume 4.
32. Dantzig, G.B.; Wolfe, P. Decomposition principle for linear programs. *Oper. Res.* **1960**, *8*, 101–111.
33. Benders, J. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **1962**, *4*, 238–252.
34. Chemla, D.; Meunier, F.; Pradeau, T.; Calvo, R.W.; Yahiaoui, H. *Self-Service Bike Sharing Systems: Simulation, Repositioning, Pricing* ; 2013.
35. Caggiani, L.; Ottomanelli, M. A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. *Procedia Soc. Behav. Sci.* **2013**, *87*, 203–210.
36. He, M.; Ma, X.; Jin, Y. Station Importance Evaluation in Dynamic Bike-Sharing Rebalancing Optimization Using an Entropy-Based TOPSIS Approach. *IEEE Access* **2021**, *9*, 38119–38131.
37. Hu, R.; Zhang, Z.; Ma, X.; Jin, Y. Dynamic Rebalancing Optimization for Bike-Sharing System Using Priority-Based MOEA/D Algorithm. *IEEE Access* **2021**, *9*, 27067–27084.
38. Chiariotti, F.; Pielli, C.; Zanella, A.; Zorzi, M. A dynamic approach to rebalancing bike-sharing systems. *Sensors* **2018**, *18*, 512.
39. Karlin, S.; McGregor, J. The classification of birth and death processes. *Trans. Am. Math. Soc.* **1957**, *86*, 366–400.
40. Fischer, W.; Meier-Hellstern, K. The Markov-modulated Poisson process (MMPP) cookbook. *Perform. Eval.* **1993**, *18*, 149–171.
41. El Sibai, R.; Challita, K.; Bou Abdo, J.; Demerjian, J. A New User-Based Incentive Strategy for Improving Bike Sharing Systems' Performance. *Sustainability* **2021**, *13*, 2780.
42. Chiariotti, F.; Pielli, C.; Zanella, A.; Zorzi, M. A bike-sharing optimization framework combining dynamic rebalancing and user incentives. *ACM Trans. Auton. Adapt. Syst. TAAS* **2020**, *14*, 1–30.
43. Hulot, P.; Aloise, D.; Jena, S.D. Towards station-level demand prediction for effective rebalancing in bike-sharing systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 378–386.
44. Zaki, M.J. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.* **2000**, *12*, 372–390.
45. Agrawal, R.; Srikant, R. Fast algorithms for mining association rules. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, 12–15 September 1994; Citeseer: State College, PA, USA, 1994; Volume 1215, pp. 487–499.
46. Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, 25–28 May 1993; pp. 207–216.
47. Houtsma, M.; Swami, A. Set-oriented mining for association rules in relational databases. In Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 6–10 March 1995; IEEE: Piscataway, NJ, USA, 1995; pp. 25–33.
48. The pandas development team. pandas-dev/pandas: Pandas. *Zenodo* **2020**, doi:10.5281/zenodo.3509134.
49. Wes McKinney. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; van der Walt, S., Millman, J., Eds.; pp. 56–61, doi:10.25080/Majora-92bf1922-00a.
50. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 95.
51. Available online: https://smartdata.polito.it/computing-facilities/ (accessed on 02/04/2021).
52. Han, J.; Pei, J.; Mortazavi-Asl, B.; Pinto, H.; Chen, Q.; Dayal, U.; Hsu, M. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001; Citeseer: State College, PA, USA, 2001; pp. 215–224.
53. Zaki, M.J. Efficiently mining frequent embedded unordered trees. *Fundam. Informaticae* **2005**, *66*, 33–52.