A machine learning approach for optimisation in railway planning

Lu Dai

A machine learning approach for optimisation in railway planning

by

Lu Dai

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Monday March 12, 2018 at 10:00 AM.

Supervisor:	Dr. Sicco Verwer	
	Dr. Wan-Jui Lee	
Thesis committee:	Dr. Mathijs de Weerdt,	TU Delft
	Dr. Sicco Verwer,	TU Delft
	Dr. David Tax,	TU Delft
	Dr. Wan-Jui Lee,	Nedtrain

CONTENTS

Abstract

Planning and scheduling problem is a hard problem, especially in real life cases. The time and space complexity increase quickly along with the increase of problem size. In transportation systems, such problems exist a lot. The automation of transportation systems depends a lot on the improvements of developing planning and scheduling algorithm. Nowadays, machine learning, as modern technology, has been adopted in every field. The power of machine learning is its ability to obtain useful information from large datasets. Considering problem instance information and corresponding solution status as data and label respectively, there is the possibility that the solvable instances hold patterns in common. This is where machine learning comes into the stage.

This research is focusing on how to combine machine learning with traditional planning and scheduling algorithm based on the Dutch Railway System. Specifically, making use of a large amount of stored instance and solution details to improve the service site planning process is our purpose. Combing machine learning and traditional scheduling system is a new and hard topic. In this work, we implement a machine learning system adapting to the scheduling algorithm content. We define and collect dataset matching our research goals. A framework is designed according to the imbalanced characteristics of our datasets. To explore the nature of algorithms, we choose to calculate features from the problem instances and scheduling process directly. Besides, we perform a high volume of experiments to select the based machine learning techniques to adopt. What's more, we also design a test framework to evaluate our machine learning systems. Improvements are observed in our work. Additionally, we would like to explore the features and machine learning techniques to improve the performance in the future work.

Keywords Planning & scheduling algorithm, Machine learning, Local search algorithm, Imbalanced learning, Feature extraction

Preface

This thesis is the final work of my master study in Computer Science at TU Delft. The thesis project is performed at Nedtrain as a one-year internship. The process of production of my thesis work is quite interesting and enjoyable.

I would like to express my gratitude to the following people for their help in the production of my thesis project. My dear supervisors Sicco Verwer and Wan-Jui Lee, without their professional guidance and suggestions, I could not produce so much wonderful work in this project. Dr. Mathijs de Weerdt, who is so nice helping with my graduation process and helping me a lot with my thesis writing. Dr. Davide Tax, who is kind to be my committee member. My boss Bob Huisman, who provides me with my first internship opportunity in a foreign country. His attitude of research encourages me a lot. And my colleague Demian de Ruijter, one of the kindest people I have met here. Without his assistance and support, the performing of my thesis work would be much difficult.

I would also like to thank my dear friends, Yuxuan Kang, Shiwei Bao, the Sun Liang Dai family. Their support and their accompany makes my life much more fun. Especially, thank them for treating me with dinners a lot of times. Besides, I would like to give my thanks to my best friend, Chenfu Huang. With her encouragements and additional supervisions, I feel supported all the time.

Last but not least, I would like to express many thanks to my dear parents, who support me all the time and always give their best wishes to my life!

Lu Dai
Delft, March 2018

1
INTRODUCTION

1.1. BACKGROUND

This study is based on the service site scheduling problem in the Dutch railway network. The Dutch railway system is one of the busiest railway systems in Europe. Every day, around a million of passengers are transported. Thus, keeping trains in a good condition and clean state is the key to perform transportation tasks well and serve the customer comfortably. NS, the biggest railway operator in the Netherlands, is responsible for Dutch railway transportation. In the meanwhile, Nedtrain as a subdepartment of NS operates inspection, maintenance and cleaning missions etc. Usually, large opertions are performed at maintenance depots. Daily tasks are processed at service sites, which are connected to the local stations by tracks. This study focuses on the latter one. These service sites consist of a number of tracks and special facilities alongside some of the tracks, see Fig.1.1 as an example. As fewer trains are in demand during the off-peak time, the service sites usually function during these hours especially overnight. The main tasks include inspection, maintenance, cleaning, recombining, parking etc.

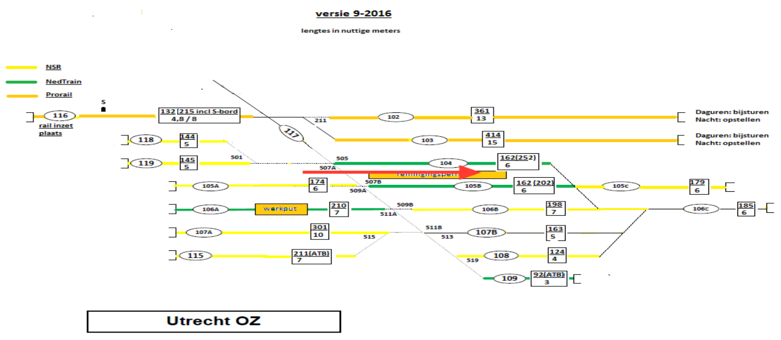


Figure 1.1: An example of service site

Considering the above, the incoming and outgoing traffic of service sites primarily take place during late evening and early morning respectively. A service site problem can be regarded as in a twenty-four hour period circumstance, during which a group of trains arrives during late evening with a set of service and parking tasks to be solved by a proper plan, in order to satisfy the next early morning's departure requirements with proper-served and recombined trains for scheduled timetable. To be more specific, the arrival and departure time are fixed, the plan needs to describe the order of tasks and on which tracks tasks are processed, every exact movement of trains, the composition of trains, the matching between arrival and departure trains, and the tracks trains parked on. Until now, the plans are manually created by Nedtrain and making a feasible plan can be hard and time-consuming.
For fulfilling the daily service process, methods for planning and scheduling the activities in service sites are necessary. The researchers at Nedtrain is putting effort into automating the plan generation process. The most recent progress is an initial software program, which has been under testing already for a while, see Fig.1.2 for the software structure. The algorithm component is the core code of finding plans for service site problems. In real life situations, plans are expected to be produced as soon as possible upon trains' arrivals in order to take action immediately. To balance the trade-off between performance and time cost, a local search algorithm with adjustable time limit settings is adopted. During the testing procedure, the time limit is set to two minutes, meaning the software will stop when time ran off even if the problem is still being solving. The roles of other components are quite intuitive as suggested by their names. Instance generator and instance checker generate instances based on the simulated data and check if the instances are reasonable respectively. The simulation data is generated according to the experienced planners at Nedtrain. The planners provide every detailed non-trivial data that can happen in the whole process such as task norm time, move-on-track time, recombined time, arrival and departure meantime and other details that might be needed. Then the algorithm component computes plans for the instances. After plans generated, the constraint checker adjudicates the plans to be feasible or not by checking if the resource and track constraints are satisfied. The Pareto front analyzer focuses on the solution status for each problem size, where size can be simply represented by the number of train units involved. A Pareto case is defined as one testing of instances holding the same number of trainunits. Each Pareto case explores the desired probability of getting feasible solutions for different problem size. The instance information and solution details are stored in the database.

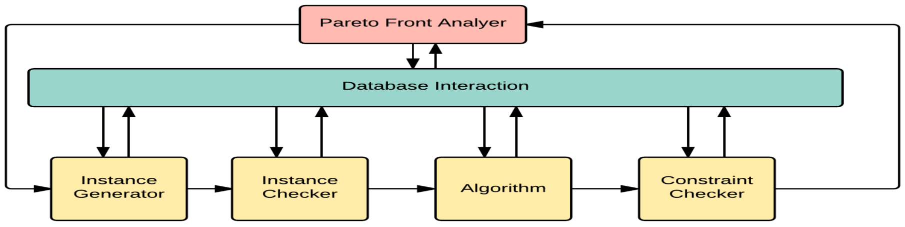


Figure 1.2: Software Framework

The purpose of automated plan generations contains both short-term needs and long-term visions. For practical needs, such an algorithm will support the decision making process during the planning procedure. Well-developed algorithms make plans much quicker than human. Even a generated infeasible solution with only a few human adjustments can become applicable. Time is the key in the service sites as complicated and flexible situations tend to happen frequently. Efficiency of these plans can affect the railway system performance. Because of the rapid increase of passengers, over time the demand for trains will increase. More trains bring much more stress to the transportation and service sites. As service sites are in the center of high-density urban areas, they have limited resources and capacity. With more trains on the railway, it will soon become important whether the facility and capacity of the service sites will be enough for handling daily services, especially considering that expanding service site can be hard and expensive. The capacity of a service site is the largest number of train units that can be feasibly planned. The algorithm along with the pareto front analyzer helps not only solving a lot of different problems but also defining the capacity of service sites.

1.2. PROBLEM STATEMENT

The most recent research focuses on the service site scheduling algorithm design and implementation. The research results with a developed local search approach and a runnable system for testing and analyzing. A large number of instances are tested and stored together with their solutions, it provides a basis for a new research area on this problem, the machine learning aspects. Nowadays, machine learning has been used in every field, due to its ability to obtain useful information from large sets of data. Considering instance information and corresponding solution status as data and label respectively, there is the possibility that the solvable instances hold patterns in common. This research is focusing on how to make use of the stored instance and solution details to improve the service site planning process. Specifically, knowing whether instances can be solved in advance is the main research purpose. The benefits lay in exploring patterns among the feasible and infeasible instances and avoiding wasting time on the infeasible ones. Furthermore, only the instances that are predicted as feasible will be computed, which largely decreases the testing time for each Pareto. This can fasten the process of exploring the maximum work package of the service sites.

1.3. RESEARCH QUESTIONS

There are many ways of applying machine learning techniques. To structure this research and choose the directions to proceed, several research questions are set up and will be answered based on the experiment results. To make use the advantage of local search, when gathering research data, every instance is run for six times. Each instance with six initializations and six final solutions are all stored.
After consideration, three research questions are defined.

- Can we predict the difficulty level of the instances?
  The difficulty level of an instance can be defined by the solution status of its six solutions' status. Predicting the difficulty level of instances can be used for adjusting the solving process. For instance, setting different time limit in the software according to the level of instances, or defining the number of initializations necessary to increase the chance of a good starting point.

- Can we identify the optimal areas of the initial solution space as starting points?
  The tiny time cost of initialization brings more benefits to predict if an initialization is a good enough start point to reach a feasible final solution. Researching the relationship between initialization and final solutions directly provide the ideas to continue or stop further search action. In addition, it guides the solving process by generating different initializations and continuing with the most promising one. This is the main research objective and is tested by live instances later on.

- Which are the dominant features in the above classification process for scheduling and planning problem?
  This research provides a new way of using machine learning techniques for scheduling and planning process. On the one hand, determining and then using a good feature set has a significant influence on the experimental results. On the other hand, it provides insights both on the relationship between problems and solution status and future research directions.

In summary, the research questions aim at exploring a way for combining machine learning techniques in scheduling and planning problems. In the meanwhile, analyzing important patterns and features existing in a traditional railway planning system and providing visions on how to enhance the automated process with modern techniques are the goals of this research.

1.4. RESEARCH SCOPE

This research will include three main areas. The first part is data engineering on the raw data including collection, cleaning, feature calculation and label creation. The second part lays on classification system training by applying common-used machine learning algorithms, imbalanced learning techniques etc. The third part contains the experiments methods discussion and results analysis to conclude the research questions. The most important part is the feature calculation section as it is both the most difficult problem and the most dominant factor in the experiments.

In this research, we analyze every aspect of the stored instance and initial solution information and calculated a comprehensive set of features out of that. As this is the initial step of a researching method on machine learning and optimization problem, only conventional machine learning techniques are discussed, where deep learning and other recently developed methods are not adopted in this study. However, considerable time is used for training different classifiers and adjusting the settings to get the best performance. What's more, the live test is carried out to evaluate the method, and it takes days to complete. Thorough result analysis is provided.

1.5. RESEARCH RELEVANCE

In most traditional industry, many processes involve a significant amount of human effort. However, some of the work is hard and time-consuming, which definitely exists a way for computer techniques to solve. The railway service site scheduling problem is such an example. This kind of optimization problems are always hard and can be continuously improved by new techniques. It usually stucks in the balance between time cost and performance. To make use of the capability of the computer and large amounts of stored information, machine learning seems a good choice to provide guidance for helping the decision problems. It is hard for human to analyze the large amount of historical problems and solutions while it is much faster for computers to compute and analyze patterns out of that. The service site problem is complicated and highly dynamic every day. An intelligent and continuously learning system will provide huge benefits for solving more problems and reaching the business goals better and quicker.
Although this research is based on the railway system, similar situations also exist in other transportation systems such as airway and waterway. Besides, scheduling and planning are also frequently needed in logistics industries. The approach discussed in this research can be extended to those areas as well. Primarily, local search is commonly used in a lot of optimization processes. This research also provides a way on how to build a machine learning framework based on a local search model, which benefits the similar optimization processes.

1.6. READING GUIDE

In the next chapter, a review of research related to machine learning in the railway system and optimization is provided. An introduction to the dataset and the calculation of feature set are discussed in detail in Chapter 3. In Chapter 4 we introduce our training system with machine learning algorithms, techniques adopted and classification results. Real life instances testing framework and the results are presented in Chapter 5. Finally, the conclusions of this research and topics for future studies are discussed in Chapter 6.

2 LITERATURE STUDY

This chapter provides a review of research on the railway system and how machine learning techniques are proposed and used in this area. We introduce a variety of research to depict how machine learning helps in traditional industries. Especially, we present articles aim at combining machine learning and scheduling problems to introduce the advantages, give insights of the methods implemented and indicate the knowledge gap in this area of study.
The chapter is organized into six sections as follows. The first section introduces machine learning and the supervised learning algorithms that are relevant to our study. The second section brings in concepts of how machine learning techniques are combined with railway systems. We introduce research for utilizing machine learning techniques to achieve different objectives in the railway systems. The following two sections discuss literature with the similar background to this study. The third section introduces the existing work of combining machine learning and scheduling problems. Especially, the next section focus on the modern learning methods on the local search algorithms, which is also the research content in our work. The fifth section explores feature extraction process that is beneficial for our case. This last section gives insights of the tackling the technic issues for our work, focusing on imbalanced learning.

2.1. MACHINE LEARNING

For data analytics, machine learning is a technique to create models to prediction[1]. The recent survey categories machine learning into three aspects, supervised learning, unsupervised learning and reinforcement learning. Supervised learning requires labeled data to train models and make predictions. Unsupervised learning finds patterns from unlabeled data. Enforcement learning allows learning from feedbacks received from interaction with external environments. We adopt supervised learning in our study. Supervised learning uses training set to fit the parameter of models and predicts labels for the test set, where the input dataset needs to be the same set of features. This process is called classification.

For our feature sets, we do not know which machine learning algorithms are the best fit. Thus, the commonly-used models are under consideration. A review of the common supervised learning algorithms is presented[2]. We follow the process of supervised machine learning as addressed by the authors. We start dealing with the data set by identifying required data, data pre-processing and then defining training set. The next step follows the repeat circle of algorithm selection, training, evaluation and parameter adjusting to the desired performance. Many classifiers are discussed in work for supporting algorithm selection procedure. The advantages and disadvantages of the models help us to make an initial selection to perform experiments. As presented by the authors, LDA and Naive Bayes provide probabilities an instance belongs to classes. SVM and neural networks hold a better performance on multidimensional and continuous feature sets. Naive Bayes classifier is fast as it requires little storage. KNN is sensitive to irrelevant data and intolerant of noise. Decision tree model has the strong interpretability to support further analysis on the dataset. The above-addressed aspects such as data quantity, accuracy, dimension, noise, further analysis are essential factors on our feature sets. Thus these common models are all considered to be tested and select the best satisfy our research needs.

Also, a variation of supervised learning is presented, which is multiple instance learning(MIL) [3]. In MIL, the labels are only assigned to bags of instances. The idea can be borrowed in our case that packing an optimization problem together with its different initializations to form training samples. However, the current MIL package limits in predicting 0 or 1 labels, which limits the possibility of defining proper labels and provides less value for performing analysis in problem level. The method is tested on the dataset where only two classes existed.

2.2. MACHINE LEARNING IN RAILWAY SYSTEMS

An overview of the railway optimization models for scheduling and routing problems is given by Cordeau [4]. The authors point out that the development of the optimization model is slow in railways compared to air transportation. The reason underlying is that railway scheduling and routing problems are large and difficult. However, in the last decades, the railway optimization problems have gained tremendous attention and with the development of sensor technologies, 5V data (volume, velocity, variety, value, and veracity) have been acquired in this domain[5]. This provides opportunities for introducing modern techniques such as machine learning to help solve problems more effectively.

The chances of utilizing AI techniques are increasing together with data. Utilizing machine learning techniques for decision support, maintenance, and transportation management is attracting many researchers. The use of machine learning on railway data in handling the traffic and explaining the usage of mobile phones, smart cards, and computers to predict the traffic and improve operations are presented[6]. Three main aspects of data are gathered in this research, including GSM data, vehicle data, passenger data. A three-step approach is designed as firstly analyzing vehicle performance, then the passenger impact, and later the travel time impact. Cost benefits are calculated based on the analysis for choosing the best strategy. Machine learning is adopted in forecasting operations and future ridership in the mentioned analysis. Improvements are achieved in the optimization of timetable scheduling, suggestions on additional travel time and corresponding business values. Notably, many pieces of research on machine learning for maintenance purpose in railways are performed.

General methods that can be applied to facilitate the decision of efficient railway track maintenance are proposed for railway track condition monitoring, based on the benchmark axle box acceleration(ABA) measurements in the Dutch tracks[7]. The ABA system provides measurements based on ultrasonic and eddy current to assess track conditions. A reduction method for big data processing is proposed to peform different frequency of monitoring based on the severe degrees of the failures. The general idea is to plan frequent monitorings for tracks that may cause problems in the near futrue. Dividing the entire datasets into smaller parts according to the track status can help implement the systems easier. Same idea is used for handling failures in big data. If a particular type of failure is predictable and the fast evolution for this kind of failures can be expected, then frequent monitorings are performed on it. If the failures cannot be predicted successfully, incorporating the data of the failures in an update of the detection algorithm is performed. In this way, the entire dataset can be divided into smaller ones regarding different failures. The systems can be easily implemented on the smaller datasets, which adapts to the processing limits. Manual intervention is also largely decreased when only data need to be incorporated. The research provides a way to play with the large quantity of data instead of specific machine learning models, especially a way of performing adaptive and self-learning mechanisms. The ideas help in monitoring tracks effectively and continuously. In our study, a significant amount of dataset also exists. Differentiating from the prediction purpose as in this research, it is not applicable for us to divide the data according to the prediction classes. However, as the prediction targets are feasibility and infeasibility of solutions, we can divide the dataset according to the difficulty level of the problem instances. In this way, we form the original problems as smaller and repeat subproblems. Also, we can expect better accuracy regarding each subproblem.

To diving into more specific methods and models in maintenance prediction, we discuss a study that adopts SVM techniques are adopted on large-scale data to produce valuable tools for operational sustainability[8]. Alarm prediction, employing distributed learning and hierarchical analytical approaches are the main aspects addressed in the research. The authors customize an SVM model by firstly defining a linear model aiming at minimizing the dot product of the hyperplane vector, with constraints of the dot product of feature vector and hyperplane multiply by the record label is above a parameter p. After rewriting the formula by Lagrange method, the authors project the feature set to a higher dimensional space by a kernel function, which characterizes a notion of pairwise similarity between instances, to achieve better accuracy. The experiments gain good results by largely reducing the False Positive Rate(FPR). Although the model is too specific for our research to study, the analytical approaches in this study provide values to perform. For instance, for alarm prediction, the authors calculated maximum, 95 percentile and mean of bearing temperatures in the historical reading time window, and the variation and trending of bearing temperatures as features. This kind of time series analysis provides an idea of calculating features from railway task status in our problem. Furthermore, studies in marketing decision for railway freight [9] and management of big data in this domain [10] have all been carried out. It is evident that every aspect of the railway system is starting to involve machine learning techniques and will need more and more attention to make a better combination. The researchers also addressed the challenges existed in common, not only regarding the continuously rising big data but also the temporal-spatial information need to take care of. It is essential to define representations of feature set and learning rules to success in useful predictions.

In conclusion, the above research gives us insights on various methods in combining machine learning in the railway sectors. Although the purposes vary a lot, the results show us the automating progress of this traditional industry and performance improvements of adopting this modern technology properly in the era of big data. The contents are much beneficial for our study, including the general way of managing data stream to help in developing artificial systems and the analytical approaches inspire us a lot in the feature extraction process.

2.3. MACHINE LEARNING IN SCHEDULING & PLANNING

Scheduling and planning problem is a common problem in real life situations, especially in Flexible Manufacturing systems, Transportation industries, logistics systems, etc. Such problems involve a lot of activities; an example can be seen in Fig.2.1

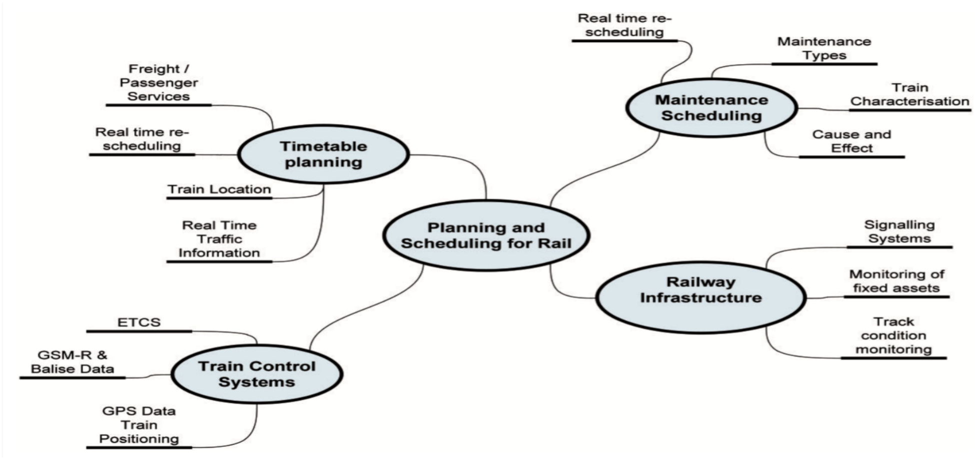


Figure 2.1: Planning & scheduling activities in railways, figure by Christopher et al. [11]

On the one hand, scheduling and planning problems can be quite complicated to solve, due to the randomness characteristic and numerous constraints. On the other hand, the automation of scheduling and planning is eagerly desired for solving problems quickly and in scale. These two points make combining machine learning and scheduling and planning quite a hard topic.

In research, these real-life cases are classified as hard problems. Different methods have been explored to solve scheduling and planning problem. According to [12], analytical, heuristic, simulation-based and artificial intelligence-based approaches are the most frequently studied approaches. We introduce the artificial intelligence approaches here as it is relevant to this study. The earliest research in artificial intelligence approaches in scheduling problems starts in the early 1980s [13]. Primarily, a review of combing machine learning and scheduling are presented[14], including the methods from the early research. The authors firstly address the importance of AI methods in helping solve scheduling problem, which is the solving processes need to have learning ability(be adaptive). This is often based on the expert systems. The expert systems are constructed by acquiring knowledge from human experts and then transfer this knowledge into knowledge representations such as rules, frames, etc. Based on these concepts, different learning methods to improve the expert system are discussed. The learning methods can be categorized as following,

- Rote learning[15].
  The trait of this method is that the problem is solved by dividing it into frames and selecting the best schedule for a new state from stored frames that are identical or similar. More specifically, this method represents knowledge with frames. Three dimensions are constructed, one includes the FMS description, the second is for part mix, while the last one contains the proposed schedule. Any state in the three dimensions that solves the problem is saved for future use as references. Thus, for any point in the three dimensions is previously stored, the expert system select it as the best schedule. If not, given the information on the first two dimensions, a simulation system selects the best schedule on several scheduling strategies to use at that point and also save for further searches. The approach avoids complicated calculations but is also limited by the initial chosen set and system states existed in the problem instance.

- Inductive learning[16–18].
  Similar to the previous method, this method also saves the system states for future searches. The difference is that a sequence of expert's decision is saved, meaning the relationship between anterior-posterior states is considered. The authors classify the system states into different classes, where each class contains a set of decision rules. An evaluation system is defined for examing the decision rules' performance, where thresholds are set up representing expert's performance. While determining the next action, all the decision rules in the class are evaluated. The best rule meeting the expert's level is selected to perform the corresponding action. If no rules perform well, the state is subdivided into smaller classes to repeat the above process until the schedule is decided.

- Case-based learning[19].
  Case-based learning approach also shares something in common with the above two methods by saving successful problem-solving cases in the past. The distinguishing point lays in the searching is based on the case context, e.g., searching for a previously solved case with similar task sets and priorities. The authors point out the indexing schemes are inadequate for building case base and subsequential retrieval due to a large number of constraints in scheduling problems.

- Genetic algorithms[20, 21].

> Genetic algorithms apply many search operators using simple heuristic strategies. A classifier system using genetic algorithms learns sequencing heuristics of jobs. Each prediction represents a heuristic. For instance, comparing the attribute values of jobs(e.g., processing time, due date, etc.) corresponds to different strategies. According to the comparison result, job positions are exchanged or remained until all job pairs are compared. This method ensures legal schedules and has the ability to find the optimal rules.

The experiments show that above methods seem to improve the performance of scheduling systems generally.
In such a system, not only learning approach matters but also the machine learning algorithms themselves make a difference. While little work discusses the machine learning algorithms, Priore et al[12] introduce commonly used machine learning algorithms in the above-mentioned approaches, they are the nearest neighbor, back propagation neural network and decision tree algorithms. The advantage of these algorithms is that their characteristics matching the decision making rules better. Although a learning system automates the scheduling process and largely deduces human efforts, embedding machine learning techniques do not always guarantee better performance than exact scheduling algorithms. The reason can be the training examples are subsets or partial states of the original problem. Plus the selected rules might be useful for the state but also hold the possibility of causing poor influence over a more extended period.
The literature discussed in this study address different ways of embedding machine learning in scheduling systems, they emphasize the learning ability in the scheduling processes. For example, learning from the past frames or cases uses for the current one, or learn to select the best rules to apply currently. However, the goal of our work is to predict whether an instance can be solved in advance, which is different from the purposes of the above studies. Our work needs more feature analysis on the problems themselves. The first three methods involve the comparisons among different frames and cases, where the comparison ways provide insights for feature calculation.

2.4. MACHINE LEARNING IN LOCAL SEARCH

As this study is based on a local search algorithm, we would like to address several pieces of work on machine learning in local search. Local search is a common method for solving optimization problems holding the advantage of balancing time and performance. The search space consists of many candidate solutions. The local search starts with an initial solution as a starting point and ends at a local optimal or ends when time run-off as an ending point. It creates a trajectory moving from solution to solution by adjusting local changes. Update formulas perform local changes. The search stops when local optimal is found or time runs off.
Automating optimization processes seems to become a favorite field where different machine learning approaches are getting involved. The relevant research emphasizes the core of improvement depends on whether learning system leads to a better trajectory. Two representative paper are discussed in the following. A recent work [22] points out that different update formulas are used in various optimization algorithms, where learning the update formula is the key to learn an optimization algorithm. It proposes to

model the update formula as a neural net with parameters. The appealing advantage is, this makes the algorithm expressive. And a whole set of parameter values provides the capacity of any update formula or their combinations to be selected. Also, it provides a broad search space for achieving the goals. In this way, the training system trains on approximately the whole solution space and yields a bigger probability of predicting a better move.

A different research [23] studies machine learning for global optimization, which provides a different view. Instead of learning for the update formulas, this research illustrates the dependency between the initial solutions(starting points) and outcomes (ending points). The authors first define label +1 and -1 to associate the starting points with outcomes beneath and beyond the threshold respectively. Those starting points are classified into two classes according to the labels assigned. The authors propose an SVM model to explore the pattern between two classes and find the separating hyperplane. A large number of historical data is used for training the model, and strong relationship is observed between the starting points and outcomes. This approach brings prediction value at the beginning of a search algorithm.

In conclusion, the first study focuses on utilizing machine learning to improve the optimize process itself to reach better performance. A number of the same kind studies adopts similar ideas. The improvements in the performance depending on control the trajectory of the moves. The second work aims at exploring the attractive areas among the entire search space to increase the chance of finding likely outcomes. The work proves the possibility of the dependency of initial solutions and outcomes. The idea is close to our research context. However, this research put more efforts in refining the SVM model to improve the performance while we put more efforts in extracting useful features to represent the initial solutions.

2.5. FEATURE EXTRACTION IN SCHEDULING PROBLEMS

In our study, we do not focus on learning of update or evaluation functions in the optimizer. Instead, we analyze the dependencies between instances and initial solutions against outcomes. We propose a way to extract features directly from the problem instances and initial solutions to represent as input. We believe this helps us understand what the critical factors underlying scheduling problems are. This section discusses the relevant literature to helping to extract features out of scheduling processes.

For service site scheduling problems, time and space are the crucial factors of a successful plan. Features regarding these elements need to be taken account. The service scheduling problems are too complicated to solve in a short time. To solve problems quickly and extract useful information out of it, we remove the constraints of resources, tracks' length and connectivity, etc., but only keep the information of tasks and available tracks for processing. The reduced problem is a relaxation of the original problem, which can be solved similar to the tardiness problem. Briefly, the multiple machine tardiness problem is considered as the problem of scheduling n jobs on m parallel machines[24]. The goal is to find an optimal schedule of jobs and minimize the total delay. Research emphasize the tradeoff between the time cost and near-optimal solutions[24–26]. How to formulate the problem and calculate for acceptable solution quickly are beneficial for computing our relaxed problems.

However, we observe the computed solutions and stored initial solutions are strongly related to time. We would like to computer features along with these timelines. Some studies provide knowledge on the state supervision in the context of the rail system monitoring. The authors model the scheduling states by curves[27–29]. The curves are acquired by continuously observing condition measurements along with time. These research adopt regression models on the curve data, which is not interested in this study. We put our references on how the curve data is handled. A general way is to separate curves into pieces, where its mean and variance characterizes each piece. A broad category of how to deal with time series data can be found in book[30]. The time window manner to scan the curves seems to be a good fit for extracting features from our dataset. These studies share characteristics in common with the service site scheduling in our case. They are all dynamic in time, state and space. Although curve data is not directly saved, the timestamps along with trains, tasks, and tracks are stored in the database. We can obtain the curve data related to trains, tasks and tracks state by writing a simple algorithm. Based on a similar idea, those data can be summarized in a way to represent their characteristics as part of the feature set. For example, calculating statistic values in each time window on the curves values can help us approximate the entire curve.

2.6. IMBALANCED LEARNING

The imbalanced learning problem is concerned with the performance of learning algorithms in the presence of underrepresented data and severe class distribution skews[31]. This is a practical issue exist in our study. From the dataset we collected, imbalances exist in most of the datasets. The complexity of the problems mainly causes this, e.g., bigger size problems are rarely easy to be solved resulting in significant amount in the infeasible class. The critical issue is to gain a delight performance on the imbalanced data sets. Following the research in this field, several techniques can be utilized in our study to tackle imbalances.

- Assessment Metrics[31].
  The commonly used assessment metrics are precision, recall, F1 score, roc curves, AUC, etc. As well known, accuracy is the most straightforward metric to examine the performance. However, this metric is highly sensitive to the dataset. In a dataset with high imbalance, classifying the entire dataset to the majority class will have good accuracy. Thus, it risks in causing misunderstandings about the predictions in highly imbalanced case. For imbalanced feature sets in our case, we can introduce precision, recall, F1 score, ROC curves and AUC. Precision assesses the exactness of the classification results, i.e., the proportion of correctly positive-labeled samples among all the positive-labeled samples. Recall measures the completeness, i.e., the proportion of correctly positive-labeled samples among the entire actual positive samples. The F1 score combines precision and Recall to tell the effectiveness of the classification. ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various discrimination thresholds. It helps us find the best model settings. AUC provides a direct value for the ROC

curve that can be used to compare easier.

- Resampling techniques[31, 32].
Random oversampling and undersampling are commonly used methods. The methods are quite simple by adding or removing a randomly selected set of samples from the minority class or the majority class. In this way, the number of samples in the two classes can be balanced. These two methods both have drawbacks. The oversampling method can lead to overfitting in the minority class while the undersampling can cause the potential loss in the majority class. Another effective undersampling method is Tomek link. AS described by its name, Tomek links represent the pairs of nearest neighbors from opposite classes. The algorithm removes the links until the nearest neighbor pairs are from the same classes. Quite intuitively, this method helps a lot in removing the overlaps among classes. ENN shares the same idea with Tomek link; the difference is that ENN removes the sample from opposite class to the majority of its N neighbors.

- Generating Synthetic Samples[33–35].
Instead of creating copies, the synthetic methods generate new data. Regarding this kind, SMOTE and ADSYN are worth discussing. SMOTE method generates synthetic data based on the similarity between samples in the minority class. This algorithm does not consider the neighboring samples, thus has the risk of increasing overlapping. ADASYN algorithm considers the above issue, it creates samples by creating new data according to their class distribution. It works by adaptively changing the weights of samples in the minority class to avoid overlapping. Besides, integrations of SMOTE and ENN or TOMEK link are common to use. The integrations help in decreasing the drawbacks from different methods and achieve better distribution.

- Cost matrices[31].
The research shows a strong relationship between cost matrices and imbalanced datasets. An alternative to the resampling techniques is the cost matrices, which targets the classification process and provides a way to set up the costs for misclassifications. The high weights on the minority class shifting the boundaries to the desired direction to achieve goals.

The techniques mentioned above are the frequently used methods, where their principles and drawbacks are discussed above. They cover broad categories regarding dataset, learning process, and evaluation metrics. The experiments from the research have proven the increase in the performance by a right imbalance learning technique. The characteristics of these methods bring us insights on how to choose methods when performing tests. As the distributions of our data vary from one to the other, we should set up different settings for each dataset. For highly imbalanced dataset, we do not try to large decrease or increase the majority or minority classes. Instead, we test with different costs to improve the performance. For overlapped dataset, we try the combination with SMOTE and ENN or Tomek Links to test the performance. In addition, we also control the sampling proportion in combination with cost-sensitive learning. ntrol the sampling proportion in combination with cost sensitive learning.

3
Data Set & Feature Engineering

In this chapter, we primarily present our work in extracting features in scheduling problems. Finding good representations for feature set is not easy in this case. We explore different aspects from the saved information to define proper representations. A large number of visualizations are drawn on the feature set to discuss the hidden patterns and values. In conclusion, we aim at presenting our approach in analyzing the structure and valuable information in the context of scheduling problems in railway sector.
This chapter is mainly organized as in four sections. The first section introduces the raw dataset collected at N.S, the cleaning process and the final format of the dataset. Importantly, it defines the entire feature set to be calculated. The second section introduces the classes defined for instance learning and initial solution learning. Particularly, two sets of features and labels are created for experimental needs. The third section provides additional information on computing features and the visualization of the feature set. Whether patterns exist is discussed based on the visualizations. The last section concludes our work in feature engineering, pointing out the further work on the feature engineering in scheduling problems.

3.1. Dataset Introduction& Feature Set Definition

3.1.1. Dataset Introduction

The raw dataset used in this research is provided by Nedtrain, generated by the simulation software program during the last few months. The dataset consists of four main aspects of data, depicted below. We reintroduce the problem and the sloving approach again to illustrate information recorded and relavant terms.

- Infrastrucuture information.
  This research is service site-specific. The distribution of tracks, the tracks' prefer-

ence on task, avaible resources are the important components for solving problems. A thourough information of the infrastructure are stored.

- Problem instances information.
  The scheduling problem involves a set of trains arriving at the service site in the evening, with various tasks to be solved. The necessary parkings and executions of tasks need to be arranged at the service site during the night to satisfy the next morning's departure. A set of trains departure according to the timetable in the next morning. Each train is composed of one or more specific train units with desired length and materials. The arrival trains and departure trains are composed of the same set of train units but may have different combinations. Each task is defined on the train unit level. The trains can be splited and combined at the service site either due to train units' maintenance or to satisfy later departure requirement. All of the above information is recorded in a time dimension. A movement refers to the move of trains on one track to another track. The location of train or train unit is the track they are on at a certain time. A track is said to be occupied when one or more trains are on the track. Two statuses of a train unit exist: whether the train unit is in task exclusion or not.

- The intial solutions information.
  The scheduling is solved by a local search algorithm, three factors support the analysis of initial solutions. First, every local search starts with an initial solution. Second, local search does not start from or end in the same position, meaning it generates different initial solutions and final solutions even in the same instance. Third, generating an initial solution takes little time. Thus, six initial solutions regarding to the same instance are recorded in the database.

- Final solutions inforamtion.
  The data structure of final solution information is identical to the structure of initial solution information.

The dataset is well-structured and stored in an SQL database, shown in Fig.3.1. The *TA* tables contain information about the infrastructure of the service site such as available tracks, facilities, etc. The *TI* tables store problem instances including the tasks to solve, the exact arrival and departure time of the trains, the compositions of trains, etc. The *TO* and *TOPGo* tables are have the same structure where *TO* tables store the initial solutions, and *TOPGo* tables store the final solutions. In the *TO* and *TOPGo* tables, every train composition, movement and track status are recorded in detail. Each combination, split, location, and status of train units can be tracked for a given time.

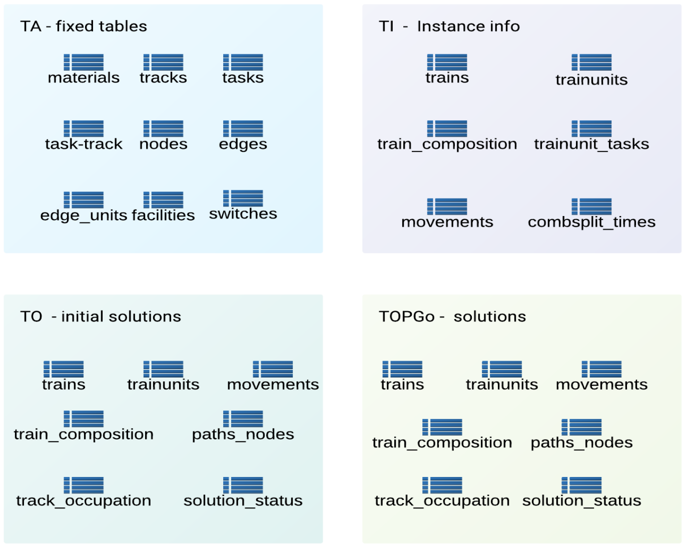


Figure 3.1: Database Tables

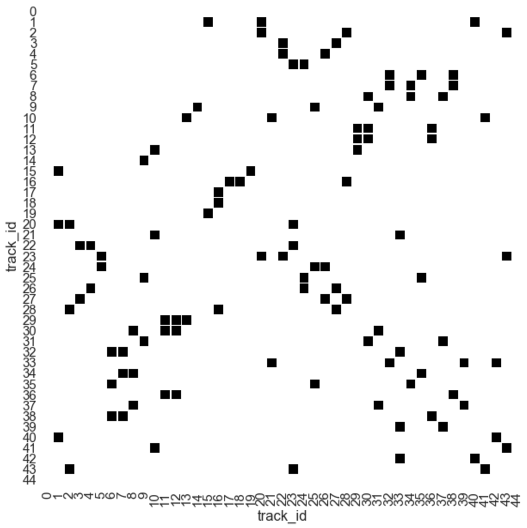
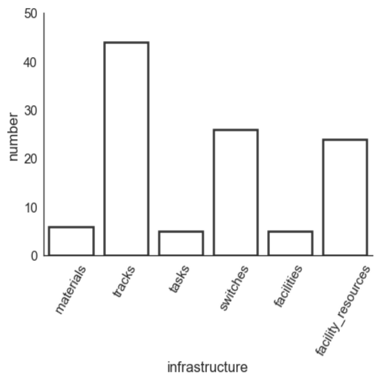
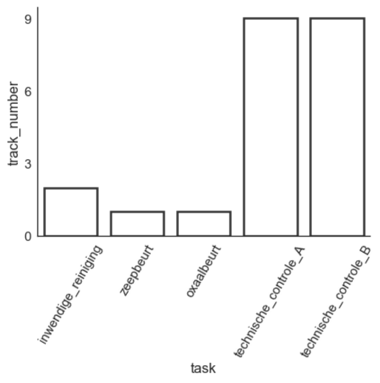


Figure 3.2: Track Connectivity

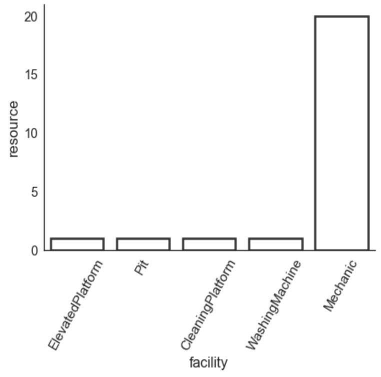
Briefly, Fig.3.2 describes connectivity situation of the forty-four tracks. On average, each track is directly connected with two or three other tracks. The track connectivity largely influences the complexity of arranging routes for shunting trains. Fig.3.3a depicts the infrastructure information in the chosen service site, including the availability of materials, tracks, tasks, switches, facilities and facility resources. A train usually consists of one or more train units, where six material types of train units are available in this case. The material type matters when performing specific tasks or doing recombinations to form new trains. The service site contains forty-four tracks and five different kinds of tasks. Priorities exist among tasks. Some tasks can only be executed on specific tracks. Fig.3.3b gives the number of available tracks corresponding to certain tasks. Together with number of tracks and resources for each shown in Fig.3.3c and Fig.3.3d, these factors determine the parallelism capability of task excutions.



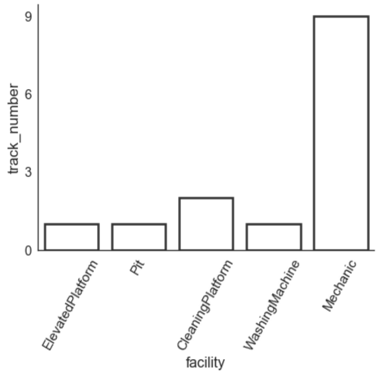
(a) Infrastrucuture In the Service Site



(b) Tasks



(c) Facilities



(d) Facility vs. Tracks

Figure 3.3: Task & Facility Availability

In real life, the timetable and routes of daily-functioned trains in the everyday transportation system are settled in advance and remain relatively fixed. To make sure the transportation systems function well and satisfy passengers daily, regular train services (e.g., cleanings), inspection and maintenance are essential to keep. On top of this, other unpredictable situations such as weather resulting in delays, out-of-blue break-ups of trains, all together make service site scheduling problems different every day and quite complicated. Intuitively, instances with more train units often indicate more compli-

cated situations, for which it is harder to find the solutions. Therefore, we refer the size of problem instances by the total number of train units in them. For a thorough analysis, instances in different sizes are generated and analyzed.
Given the nature of local search algorithm, the starting point and the ending point of the local seach's solution space are seldom to be the same even for the same problem instance. If one single solution is generated for each instance, the analysis can be tricky. For instance, a single solution can not provide label for an instance's solving feasibility. It is possible that an instance is solved nine times out of ten runs, but it is stored along with an infeasible solution that happens to occur when collecting data. This brings inaccurate labels and may cause poor performance in the learning system.

To decrease the bad influence in machine learning process, we try to collect more data on solutions. The data generation process is set up follwing. When the instance generator generates one instance, the instance gets six runs by the scheduling algorithm. Each of the six runs generates one initial solution and one final solution. The instance, the six initial solutions and the six final outcomes are all stored in the database. Thus the number of initializations and final solutions are both six times of the number of instances. The exact number of instances and solutions are shown in Table.3.1.1 and Fig.3.4 in the category of problem sizes. The number of instances for each category is not the same. This is because we consider a thousand instances as enough samples for each problem size, but the exact number is not controlled when generating instances.

Trainunits	Instances	Solutions
16	7077	42462
17	4789	28734
18	3261	19566
19	1467	8802
20	3012	18072
21	1658	9948
22	9544	57264
23	3067	18402
24	3422	20532
25	2077	12462

Table 3.1: Instance & Solutions

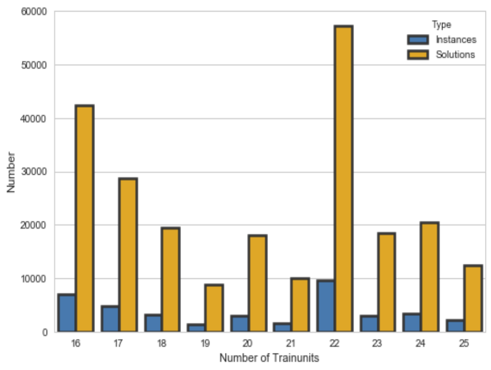


Figure 3.4: Instance & Solutions Number vs. Instance Size
A visualization of the table on the left.

3.1.2. FEATURE SET DEFINITION

Since detailed data is necessary for storing exhaustive information for scheduling and planning problems and their solutions, the raw dataset from Nedtrain is formatted regular, relational and thorough. Those kinds of data have less representation power for instance or solution. A large number of calculations need to be performed on the raw dataset to extract features. The purpose of feature engineering in this study is to define more distinguishing information to identify objects from different classes and format this data as machine learning algorithm input.
After manual analysis of the structure of the original dataset, feature calculation can be carried out in the following aspects. The features describing the instances include train

analysis, tasks analysis and the feature set for the initial solution are extracted from the initial solution graph including task analysis, time analysis, and track analysis, etc.

Instance Feature Set The relevant information for problem instance mainly focus on the trains arrival & departure, tasks. We define a set of features out of these two aspects of data to be calculated.

Train Model

- Arrival Trains: the total number of trains arrive at the service site in one instance.
- Departure Trains: the total number of trains departure from the service site in one instance.
- Arrival Time: average arrival time of the arrival trains in an instance.
- Departure Time: average departure time of departure trains in an instancce.
- Arrival Time Interval: the average intervals among all anteroposteriorly arrival trains.
- Arrival Time Interval Std: the variance of the arrival time intervals among all anteroposteriorly arrival trains.
- Departure Time Interval: the average intervals among all anteroposteriorly departure trains.
- Departure Time Interval Std: the variance of the departure time intervals among all anteroposteriorly departure trains.

Task Model

- Task Number: the total number of tasks on all train units in an instance.
- Task Arrival Density: average of tasks between two successively-arrived trains per minute.
- Task Departure Density: average of tasks between two successively-departured trains per minute.
- Task Density: average of non-executed tasks in the service site per minute
- Task Peak: max number of non-excluded tasks during the shunting process
- Task Process Period: the total time that are under task execution.
- Task Waiting Time: the total time tasks wait to be processed.
- Track Spare Time: the total time track is not occupied.

- Task parallelism: the average of tasks that are processed simultaneously.
- Task Process Time Efficiency: the average of task wait time to the task process period.

Initial Solution Feature Set From the data, initial solution seems to contain more information than the problem instance. More aspects can be analyzed in this situation, for instance, the tasks, the tracks and the graph model.

Statistics

- Shunt Time: the active time of the service site from the first train arrives until the last train departures.
- Movements: the total movements of trains in the shunting process.
- Paths: the toal number of paths trans take.
- Shunt Moves: the number of movement with a specific shunting purpose such as to execute tasks.
- Recombined trains: the number of new trains generated by split or combination in the shunting process.

Task Process Efficiency

- Task Process Time Total: the total time used for processing tasks.
- Task Wait Time Total: the total wait time of tasks to be executed in one instance.
- Task Process Time Overlap: the total time overlap of tasks that are simultaneously under processing.
- Task Process Interval: the total time of the intervals between dealing with two successively tasks.
- Parallelism of Tasks: the total pairs of tasks that share any part of processing period.

Track Utilization

- Track Occupation Time Total: the total time that tracks are occupied with trains.
- Occupated Track Number: the number of tracks that have been occupied in one instance.
- Track Occupation Time Average: the average occupation time of each track.
- On Track Time per Train unit: the average time that each train unit spends on tracks.

Graph Model

- Crossings: the number of crossings of trains' routes happen in one initial solution.
- Arrival Delay: the number of arrival delays in one initial solution.
- Arrival Delay Sum: the sum of time of arrival delays.
- Combines on Departure: the number of combinations of trains on the departure track.
- Track Length Violation: the number of tracks that the length is not enough for trains that are on them.
- Track Length Violation Legnth: the length that the tracks need more to contain the trains properly.
- Track Length Violaition Duration: the time period of tracks are occupied with longer trains.
- Overal Cost: the over coast is a weighted linear combination of the above mentioned features.

Both of the feature sets require large computations to get the feature values out of thousands of instances and initializations. More aspects can be analyzed in the initial solutions than in instances. This is because the solutions contain the thourough plans, where more information exist. We define our feature with repspect to general statistics, time and space. We believe the feature sets along with these factors have certain representation ability and classification power. We provide a section on visulizing the features later in this chapter to discuss patterns and necessarity of the features.

3.2. SOLUTION STATUS & LABEL SETS

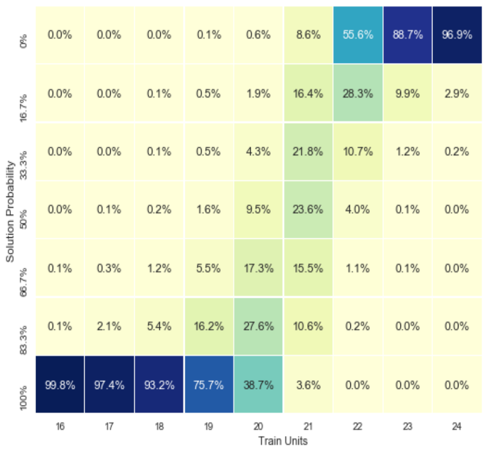
Knowing number of instances and solutions, the table below gives more detailed information on solution status and feasible solution existence in the different problem sizes. In the table, the ***solution with status*** field shows the exact number of infeasible (0) and feasible (1) solutions among all final solutions for each size of problems. The ***instances with feasible solution probability*** field shows the number of instance with corresponding solution probability. The feasible solution probability stands for the probability of getting a feasible solution for an instance, calculated as $\frac{number\ of\ feasible\ solution}{6}$. As each instance can hold zero to six feasible solution(s), seven classes are generated correspondingly.

To compare the situation in each dataset. Fig.3.5 emphasizes the percentage of instances and solutions from Table.3.2, categorizd by different problem sizes. It holds that smaller instances are yielding higher feasible solution percentage, which addresses the hugely imbalanced datasets for feasible and infeasible classes, especially for small and big size instances. Two phenomenen pop up in the figure below, which proves that the service site scheduling algorithm generates different solutions for the same instance. More importantly, the solvability level varies from instance size - for small and big sizes

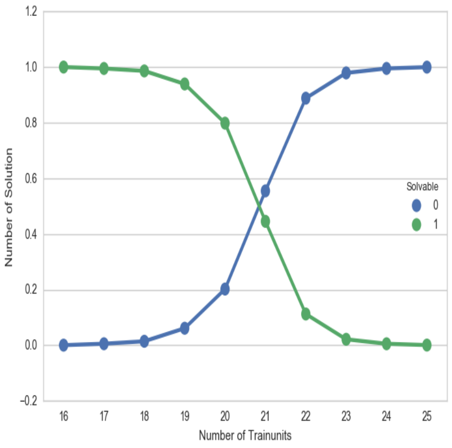
Table 3.2: Dataset

Size	Solutions with Status		Instances with Feasible Solution Probability						
	0	1	0%	16.7%	33.3%	50%	66.7%	83.3%	100%
16	25	42437	1	0	0	1	5	6	7064
17	163	28571	1	1	0	7	16	99	4665
18	284	19282	0	2	2	5	38	175	3039
19	540	8262	1	7	8	23	80	238	1110
20	3652	14420	19	57	131	285	521	832	1167
21	5488	4460	141	270	360	389	255	175	59
22	50825	6439	5305	2703	1025	382	106	22	1
23	18004	398	2719	305	38	3	2	0	0
24	20418	3316	99	6	1	0	0	0	0
25	12451	11	2067	9	1	0	0	0	0

(with sixteen to eighteen or twenty-three to twenty-five train units) the vast majority of instances hold similar solution probabilities while for middle size instances (with nineteen to twenty-two train units), solution status varies a lot. Due to time and storage limit, we can't calculate more solutions for each instance.



(a) Instance Percentage in Problem Size vs. Solution Probability



(b) Problem Size vs. Feasible/Infeasible Solutions

Figure 3.5: Dataset Overview

From the above information, two sets of labels are defined. At instance level, the whole dataset can be mapped to seven classes corresponding to the seven feasible solution probabilities. At solution level, two classes exist as solution status zero and one. These labels are used for the training system later-on. In the following section, feature sets are introduced and visualized at both instance level and solution level to check if these feature representations hold any apparent patterns among different classes.

3.3. FEATURE CALCULATION & VISUALIZATION

In the field of public transportation, feature engineering desires more domain knowledge for creating a helpful feature set to support machine learning systems to work. In this case, two purposes of machine learning exist.

- Predict the solvability of an instance in advance.
  There are some possible reasons underlying instance solvability. The first is the complexity of the instance situation.Some cases can be solved easily while other instances themselves can be too hard to be solved. The second is the local search differences in the final solutions' status, indicating that the initial solution can also decide the instances solvability. That is to say, instances with more excellent initial solutions tend to be solved better.

- Predict the feasibility of an initialization in advance.
  For thousands of initialization and final solution pairs, there is possibility that some parts on the entire search space have bigger potential to lead to a good outcome. This means starting point of search can be essential. To figure out if there is strong dependency on the initializations and final outcomes, we train model to predict the promising initial solutions.

As defined previously, the following content introduces necessary feature computation process and feature visualizations on corresponding labels. Time series and approximation techniques are involved in the calculation process.

3.3.1. INSTANCE FEATURES

Train Model Train unit is the elemental composition of the railway system, and the train is the basic functional unit for the passengers. Trains arrived at and departed from the service site are not always the same both in number and composition. From **ti_trains**,**ti_train_composition**,**ti_movements** tables, train number, compositions and arrival and departure time can be calculated.

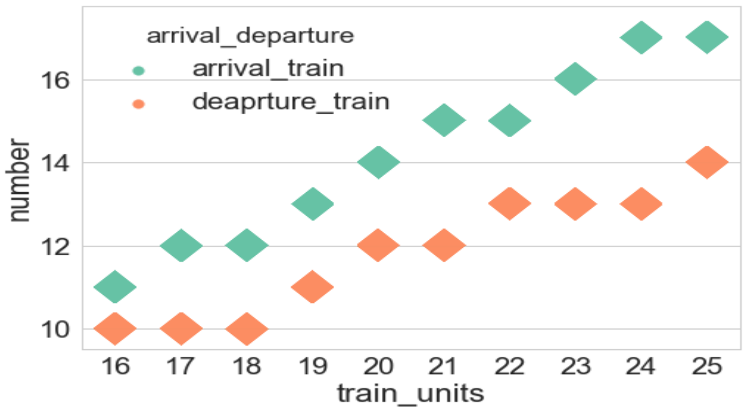


Figure 3.6: Arrival & Departure Train Num vs. Problem Size

Fig.3.6 shows the number of arrival trains and departure trains at the current service site. The number seem to be fixed for each problem size, and larger instances always have more trains participated in. Recombinations are detected in the number difference of arrival trains and departure trains. We calculate features about the trains' arrival & departure time, the average interval among trains' arrivals and departures in every instance, etc. The features are elaborated in more figures below.
Fig.3.7 describes the hitogram of average arrival and departure time of all trains in every instance categorizing by solution probability. Obviously, instances with solution probability 100% and 0% take a significant proportion and fit well for Gaussian distributions with similar mean value and slightly different std value. This feature does not represent much information as it proves that the average arrival and departure time is generated based on the settled mean values.

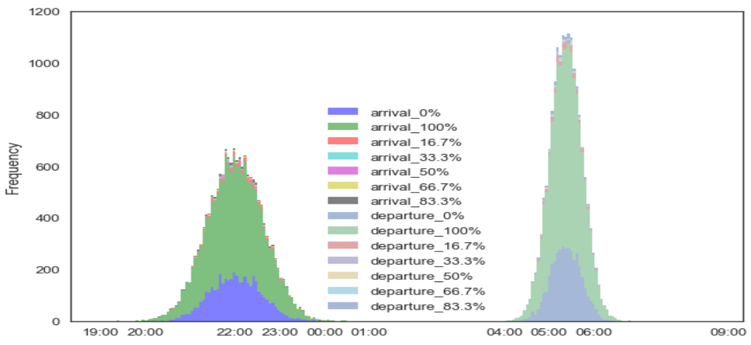
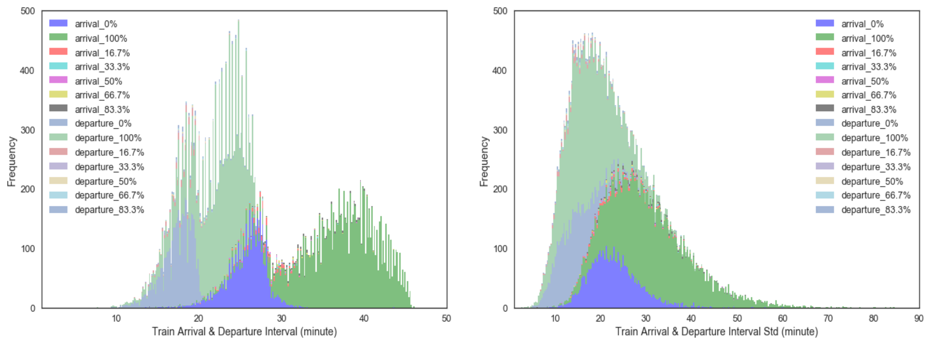


Figure 3.7: Arrival & Departure Time With Solution Probability

The time intervals of arrivals and departures of trains in one instance are more distinguishing. Fig.3.8a and Fig.3.8b depict the average and std value of time intervals among trains in each instance respectively. Intuitively, shorter arrival and departure intervals in an instance usually result in a more complicated situation for scheduling. The figures prove the trend that instances with higher solution probability holds longer arrival and departure interval and bigger std values, and for instances with shorter intervals, it is harder to find solutions. However, different degree of overlap exists among classes.



(a) Arrival & Departure Time Interval Avg (b) Arrival & Departure Time Interval Std

Figure 3.8: Arrival & Departure Time Interval Avg & Std With Solution Probability

Tasks Model For most instances, trains arrive with multiple tasks. Tasks are defined on train unit level. Tasks is an essential factor that dominates the complexity of situations. Two groups of task-related features are extracted. The first group is to represent task situations during the whole shunt process including task number, task arrival/departure frequency, the density of unfinished tasks and the maximum number of incomplete tasks in the entire process. The corresponding features are calculated as follows:

- Task Number:

$$task\_num = \sum_{i=0}^{m} tasks(train\_unit_i)$$

- Task Arrival Density : same calculation applies for task departure density

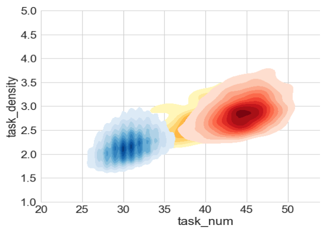
$$task\_arrivals = \frac{\sum_{i=0}^{n} \frac{tasks(train_{i+1}) + tasks(train_i)}{arrival\_time(train_{i+1}) - arrival\_time(train_i)}}{n-1}$$

- Task Density :

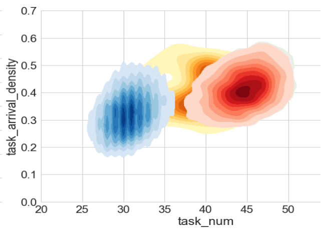
$$task\_density = \frac{\sum_{i=0}^{t} weight_i \times tasks(time_i)}{\sum_{i=0}^{t} weight_i}$$

- Task Peak :

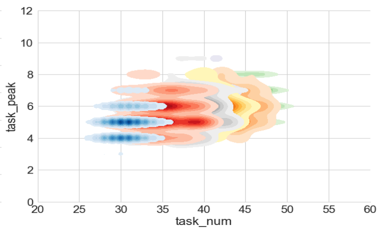
$$task\_peak = \max(tasks(time_i))$$



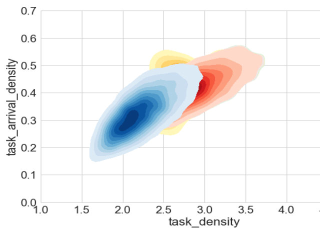
(a) Number of Tasks vs. Density



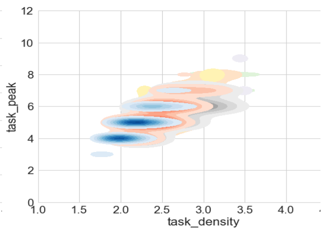
(b) Number of Tasks vs. Arrivals



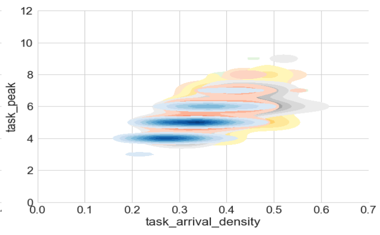
(c) Number of Tasks vs. Peak



(d) Task Density vs. Arrivals



(e) Task Density vs. Peak



(f) Task Arrivals vs. Peak

Figure 3.9: Task features Joint distribution with feasible solution probability classes
Color blue, red, grey, yellow, purple, orange, green stands for class 100% to 0% respectively.

Fig.3.9 draws joint distributions among these features in the seven classes. From the subgraphs, it can be easily observed that task peak has no great importance in classifying feature set. The reason can be that task peak number only represents the maximum number tasks during shunting process and remains in several fixed values. The same values can be easily reached in instances from different classes. In the remaining joint distributions, class 100% is naturally separated from the other classes while the other classes have substantial overlap. One possible reason can be that the number of samples for class 16.7% to 83.3% are too little to be separated. Another possible reason is that no matter which class is, the instances in the largely-overlapped area has at least one infeasible final solution, resulting in the common values in the features. This analysis inspires the idea to merge classes that are less in quantity and similar in patterns. Furthur experiments are provided on this.

The second group of task-related features aims at estimating the optimal task process situation, including total process period, task waiting time, track spare time, task parallelism and task process time efficiency. For extracting that information in a short time, we calculate a relaxation of the original service site problem. We remove other constraints but only consider the tasks and the availability of tracks. In this way, we treat the relaxed problem as similar to the multiple machine tardiness problem. Define: $C(i,t,j)$ as completion time of task j of train unit i on track t, $A(i,t,j)$ as available time of task j of train unit i arrives on track t, $Track(j)$ as tracks for processing task j, $Time_t$ as current time of track t, $S(i,t,j)$ as the start time of task j for train unit i, which is the max of previous task $j-1$ completion time on track j and the arrival time of task j on track t, aka $\max(C(i,t,j-1), A(i,t,j))$. The completion time equals start time plus task duration. For arranging a task to a proper track, it could $Minimize \max Time_{t=0}^{num\_of\_tracks}$.

So the schedule can be acquired by inserting task to available tracks and getting its previous task's finishing time. Then selecting out the best track with min $S(i,t,j), t \in Track(j)$, inserting the task into the track and sorting the task by $A(i,t,j)$. After getting the schedule on each track, the features as mentioned above can easily be calculated in the time series manner.
Fig.3.10 illustrates the task process features' behaviours among different instance class. Obvious increasing and decreasing trends can be detected along the classes, proving a good criterion for separating different classes.

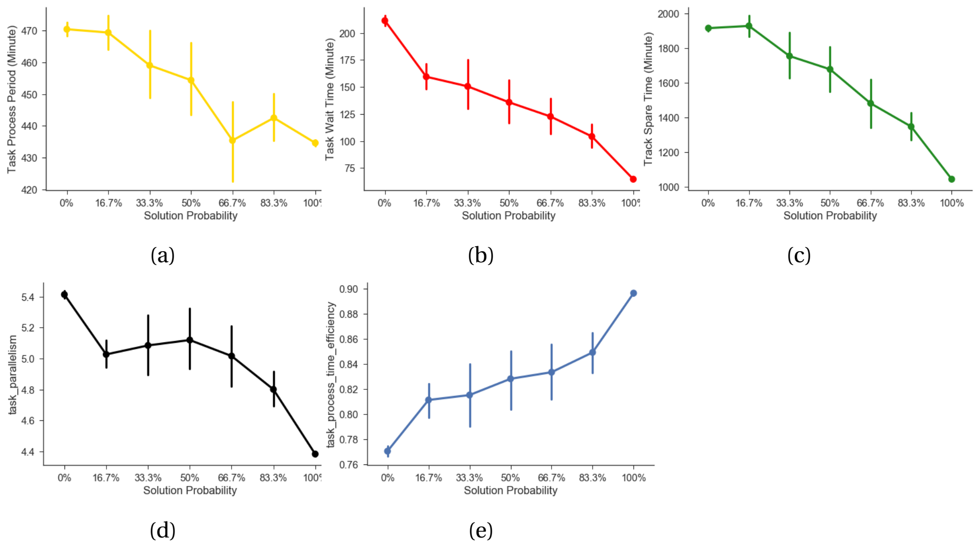


Figure 3.10: Task Process Features

3.3.2. INITIAL SOLUTION FEATURES

As discussed beforehead, good starting points increase the chance for solving problems. Extracting features from initial solutions are of great use. Two reasons can support initial solution analysis. Firstly, the instance information provided is limited, and the calculation of functional instance features are complicated and time-consuming. Besides, the variation in the final solutions may cause undesired learning results if the learning system only trains on instance features. Secondly, one production of initial solutions take little time; this provides feasibility for extracting initial solutions features. The below paragraphs introduce the calculated initial solution features in detail. These features are mapped on the two classes(feasible and infeasible), demonstrating their correlation status. The objective of analyzing initial solutions is extracting distinguishing features for identifying whether it is an optimistic start position to continue searching. For thousands of instances in the database, none of the initial solutions is feasible. However, this is a typical situation, and there must be promising areas in the solution space pointing at the right directions to solve the instances.

In general, we first calculate the active time of the service site for each problem instance, see Fig.3.11. The feasible class holds slightly smaller shunt time than the infeasible class. Briefly, this shows that better initializations carry shorter shunting process, which is potentially caused by more train units in infeasible class on average.

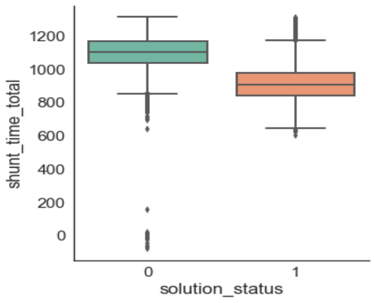


Figure 3.11: Service Site Active Time

Path & Movement Analysis Then we analyze the trajectory of trains in the initializations, which can be a big reason for costing time and causing complex situations. The trajectories largely depend on track structures in the service site. Ideally, a fully-connected and resource-rich track structure could help scheduling much more comfortable. While in more difficult track distributions, trains have to take much more complicated routes. The following figure.3.12 presents statistics of movements of trains, nodes that trains passed, paths and shunt movements in the initial solutions. Significant distinctions between two classes show up in their self-correlations while considerable overlaps exist on the joint distribution among different features. This may indicate that it is easy to classify between very good and dangerous initializations, but hard to distinguish the rest.

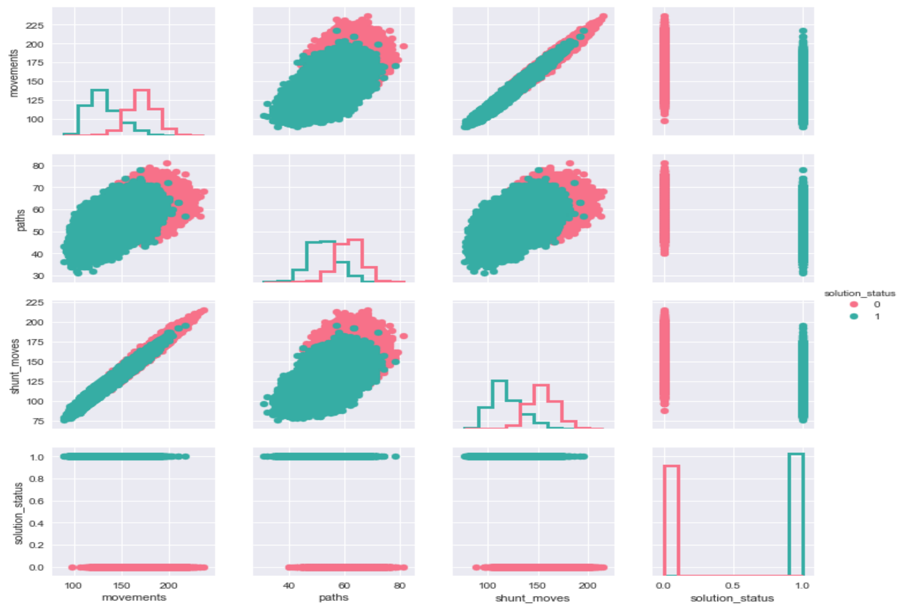
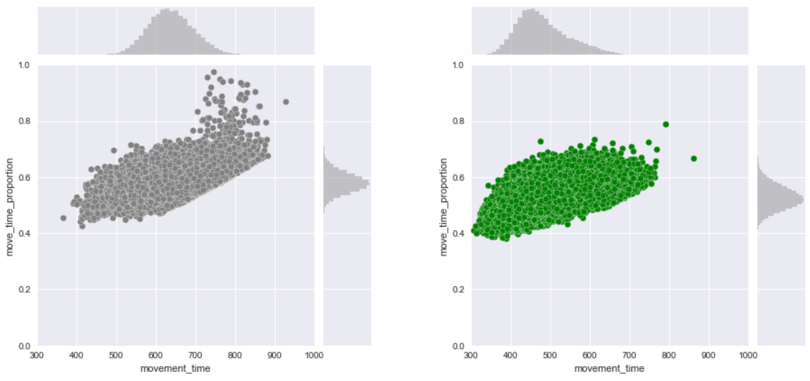


Figure 3.12: Path & Movement Statistics & Correlation



(a) (b)

Figure 3.13: Movement Time & Move Time Proportion Joint Grid
(a) stands for the infeasible class;(b) for the feasible class; Move time proportion is calculated as movement time divied by service site active time

Besides, how much time the movement consumes is taken into consideration. The figure below contains the information of movement time and proportion of movement time to service site's active time. These features give some insights on time used in the shunting process. For feasible cases, the range of movement time is from 300 minutes to 750 minutes, concentrates on around 450 minutes and the portion of movement time shows a normal distribution with a mean value of 0.55. While, for infeasible cases, movement time varies from 350 minutes to 900 minutes approximately, with a mean value of 650 minutes. The mean value of portion is also normally distributed, from 0.5 to 0.7, a mean value of 0.6 can be obtained from the graph. In a comparison of these two conditions, the overlap of movement time proportion is bigger. This information indicates that the movement time and the total shunt time have similar behaviors.

Combined & Splitted Trains From the solution information stored in the database, we also notice that many splits and combination of new trains happen during the shunting process. Those are the shunt trains. This action is mainly caused by the limitation of track length and requirements of departure trains. The following figure counts the shunt trains in different classes, where fewer shunt trains exist in the feasible class.

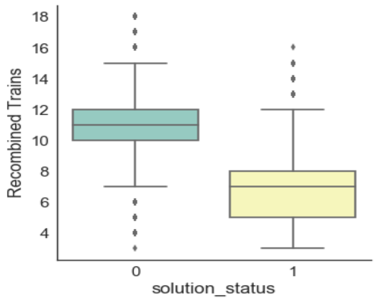


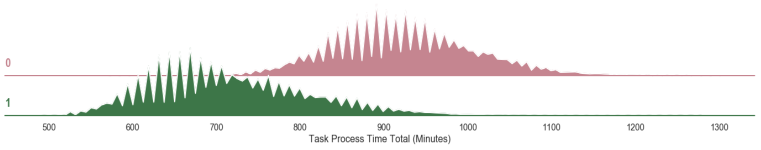
Figure 3.14: Recombined Trains in Shunt Process

Task Process Efficiency The next analysis focuses on the task analysis, as the information concerning the exclusions of tasks is available in initializations. Task process efficiency analyzes time usage while performing the tasks. Different aspects of task process are calculated, including the time for processing tasks, the time that tasks wait to process, the parallelized process time among various tasks, etc. Part of the features has been visualized in Fig.3.15. For bringing convenience to observe the two classes, the infeasible class is drawn on the top, and the feasible class is drawn at the bottom. An explanation of depicted features is as follows,

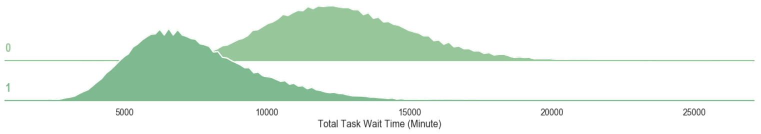
- Subfigure 3.15a presents the total task process time, where clear boundary exists in two classes. Feasible cases have lower mean values compared with infeasible cases.
- Subfigure 3.15b is the sum of wait time for tasks from their arrivals at service site until their exclusion starting moment. A similar trend is observed as the previous one.
- Subfigure 3.15c demonstrates task process time overlap, meaning the common period between every two tasks that are currently under process, providing a measure of good use of time.
- Subfigure 3.15d gives information about time waste during the task process period.

- Subfigure 3.15e describes the number of tasks can be processed in parallel, where parallelism is identified once two tasks the time overlap without considering the overlap period length.

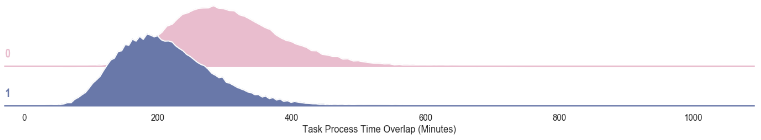
The features show similar distributions with different size of overlapped areas. As the following figures suggest, some features get close mean value the mean values of task process, which might not be of use of classifications especially for middle size instances.



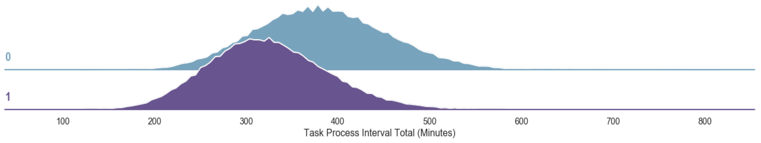
(a) Task Process Time Total



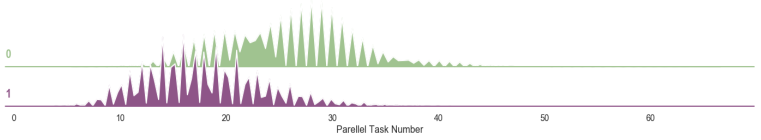
(b) Task Wait Time Total



(c) Task Process Time Overlap



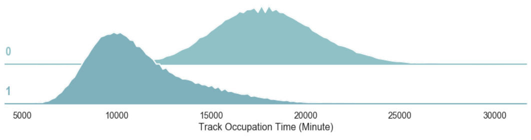
(d) Task Process Interval



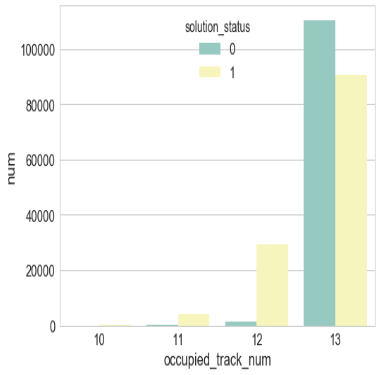
(e) Parallel Task on Process

Figure 3.15: Task Process Efficiency

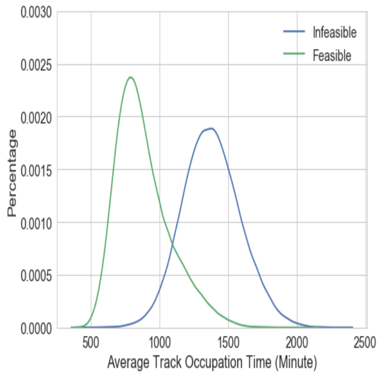
Track Utilization Except for the efficiency of task exclusion procedures, the effectiveness of track usage is also carried out. These features address the time and the number of occupied tracks. For total occupation time of tracks, there is a clear boundary appearing in Fig.3.16a, which may be a helpful potential criterion. The number of tracks occupied in Fig.3.16b holds no important differences. The remaining two features in Fig.3.16c and Fig.3.16d represent the occupied time per track and per train unit respectively. Again, similar trends show up between the two classes.



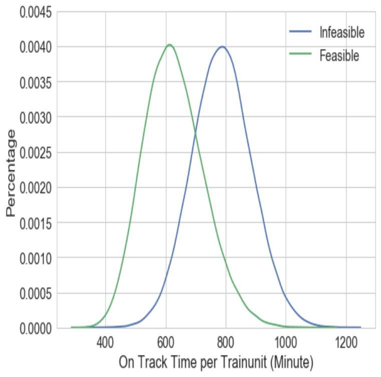
(a) Track Occupation Time Total



(b) Occupated Track Number



(c) Track occupy time average



(d) On Track Time per Trainunit

Figure 3.16: Track Occupation

Graph Model In addition, the local search algorithm models the scheduling problem with graph and provides an estimation of the initial solutions. This estimation information can be directly extracted from the algorithm process. Below, we provide a visualization of the features we store directly from the solving process. Observing all the subfigures in Fig.3.17, the mean values differentiate between feasible and infeasible classes. Some features show a relatively big difference, which holds the ability to classify classes to some extent.

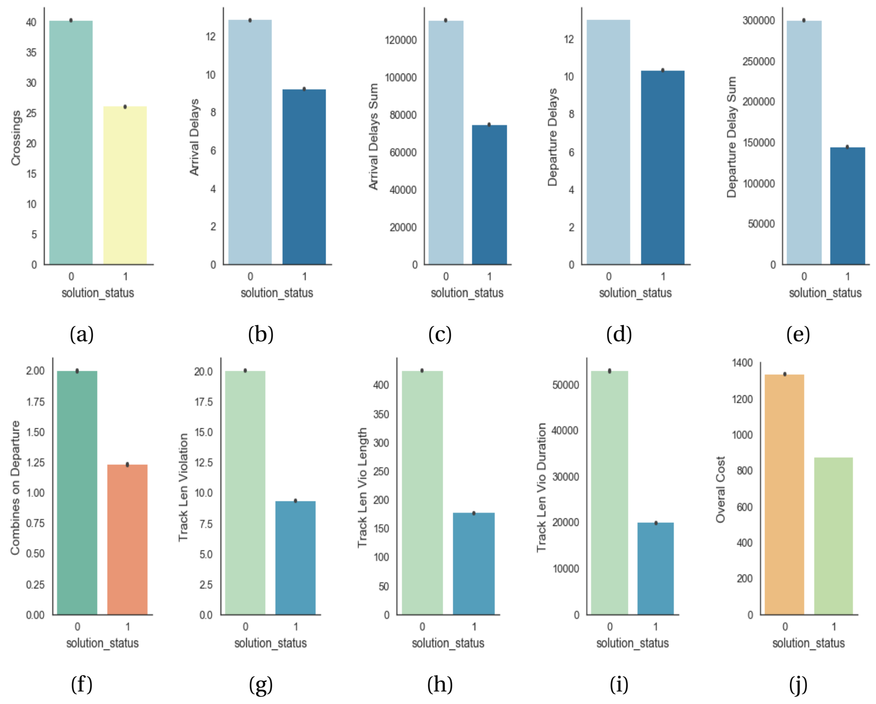


Figure 3.17: Graph Features vs. Solution Status

3.4. DISCUSSION

In conclusion, the feature sets are broad, which contain the most aspects exist in the information stored in the database. The instance features are less than the initial solution features, as the instance information is less thorough than the initial solution instances. Besides, the calculation of instance feature involves of manually defined approximation algorithms. In practice, we want to keep the time for computing features as less as possible. The trade-off between feature set and time cost is also an important factor to be considered.

Some of the extra computed features are not included, such as the analysis on one specific task which appears most frequent but with fewer resources than other tasks. However, its feature values do not show proper patterns. Also, the differences in train compositions, train unit type, and materials have been studied. The results do not indicate potential values on this kind of information. Also, we compute a lot of features in a timestamp manner, such as the features regarding tasks. We use time window and mean value to approximate the timestamp curves to summarize as a single feature value. This might decrease the potential value hidden in the feature curves. A more in-depth dig of values of features should be performed. For example, further adjustings with time window sizes and better summarize methods could be adopted to get the feature values holding the most apparent patterns.

From the visualization of features on the class labels, lots of features show certain patterns that can be separated among classes, especially the features regarding tasks. However, the visualization is based on the entire feature set, which is not for each problem size. Separate visualizations for each problem size can eliminate the obvious patterns we observe from above. This can somehow explain why the classification performance in the later chapter do not match with expections as the clear pattern we see.

4 TRAINING SYSTEM

4.1. FRAMEWORK

In the previous chapter, we introduce two sets of features and classes. The learning processes are carried out on the two aspects, earning on the instances and learning on the initializations. The first part of work is to predict solution probability out of the instance features. Due to the imbalance in classes, classes are appropriately merged when training. The second part is to train a system on the initialization features. We use the system to directly predict whether an initialization is good enough to search on. In this training process, different preprocessing techniques are involved. The results are estimated with multiple metrics.

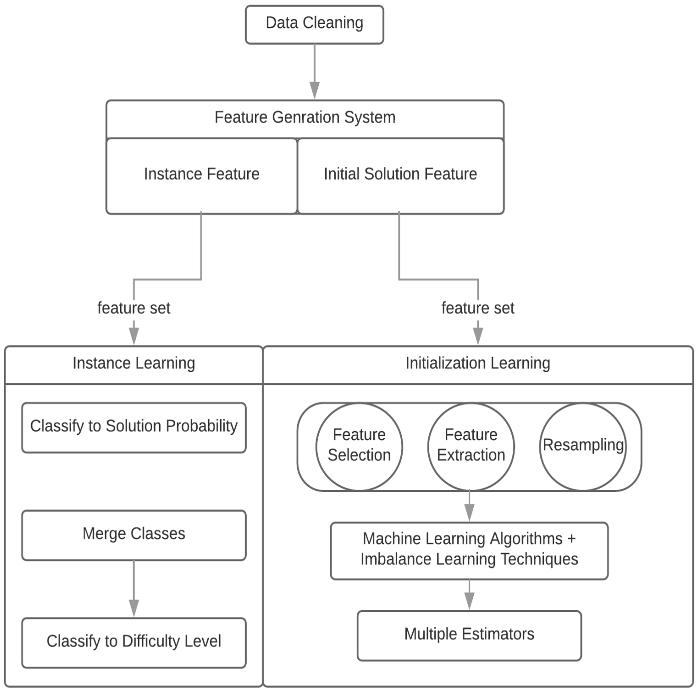


Figure 4.1: Machine Learning System

The contents mentioned above of machine learning system are depicted in Fig.4.1. Following the diagram, the whole process starts with data cleaning to remove untidy dataset, for example, drop instances with generation error or with less than six solutions, etc. Then generating features in the feature generation systems. After that, the training system takes over. An additional test system is designed for analyzing and testing the training system in the next chapter.

4.2. IMBALANCED LEARNING

Traditional machine learning algorithms aim at improving the overal accuracy. However, these algorithms often generates misleading results on imbalanced datasets. Imbalanced learning techniques need to be adopted in the research. As addressed in the last chapter, different levels of imbalance exist in the datasets. For instances-level training, instances with size 16 to 19 tend to overfit with the class holding 100% solution probability. While instances with size 23 to 25 overfit the class holding 0% solution probability. For datasets of instances with size 20 to 22, the situation is relatively much better. The same situation holds for initialization learning between feasible and infeasible classes. For initial solutions generated from instances with size 21, the dataset is almost balanced. Other than that dataset, the rest sets either overfit in the feasible class or the infeasible class. For tackling this issue to achieve a good machine learning system for real use, several techniques for imbalanced learn will be used later in the experiments. As collecting more data is not helpful with the situation, we perform the experiments on the dataset we have. The python package of imbalance learning is used. The used strategies are as following,

- Initial training on a broad range of algorithms.
  At this moment, we do not know the structure of the patterns in the feature set clearly. There are a large number of machine learning algorithms available. In order to not stuck in our favorite algorithm, a broad check on the common machine learning algorithms help us to find the best types of algorithms on this problem.

- Multiple estimate metrics.
  Various evaluation metrics are used such as accuracy, class accuracy, confusion matrix, roc curves, AUC, etc. For imbalanced feature set, accuracy is often not enough but even cause misunderstandings about the prediction results. In the experiments, we induce various metrics to keep track of the actual performance. Class accuracy gives us information about the correct prediction probability on each class, especially for the minority classes. Confusion matrix contains not only correct classifications but also incorrect classifications, and which classes the incorrect classifications are assigned. ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various discrimination thresholds. It helps us find the best model settings. AUC provides a direct value for the ROC curve that can be used to compare easier.

- Weight & cost adjustments.
  Setting weights for each class or costs for misclassifications can help adjusting the classification surface. This technique often provides us with better accuracy in the desired way.

- Resampling Methods.
  Another nice choice is to increase the minority class or decrease the majority class to find a good balance feature set. The common approaches for resampling are oversampling, undersampling and over-under sampling. For oversampling, we adopt random oversampling, SMOTE, etc. Naive random oversampling generates new samples by randomly creating copies of samples in the under-presented classes. It keeps the original information but increases the risk of overfitting. SMOTE is also a common oversampling method. It works by creating synthetic samples from the minor class instead of creating copies. However, when generating new samples for one class, SMOTE does not consider the instances from other class in the overlapped area. This method avoids overfitting issue but has a chance to increase the overlap among classes. For undersampling, there are methods such as random undersampling, such as cluster centroid undersampling, tomeklinks. As the opposite to random oversampling, random undersampling holds the same technique but for decreasing the over-presented classes. The disadvantage is obvious that it highly risks in losing valuable information. Cluster centroid undersampling reduces the number of samples by K-means centroid. Then each class

is synthesized with the centroids of K-means. This method fastens the training speed but also risks in losing information. TOMEK links remove unwanted overlap between classes by pairing nearest neighbors and removing majority class links until all pairs are of the same class. The techniques of combining undersampling and oversampling (such as SMOTETOMEK, SMOTEENN) are also adopted in the experiments. The principle is to use both oversampling (SMOTE), and undersampling (TOMEK Links, KNN) mentioned above. The advantage is that it balances the dataset by SMOTE, and decrease the overlap by the undersampling methods at the same time.

- Classes merge.
  In the situation of multiple classes, some classes may have too little data and yield closer feature space with each other. Merging those classes into a new category brings benefits for both fixing the imbalance level and restructure clearer patterns. Oppositely, we could also split the majority classes into smaller classes.

There are many potential factors can affect the learning system in this problem, not only the problem is hard but also the feature set is not perfect, the algorithms and techniques have their limitations, plus imbalanced learning is hard to manage. We do not yet know what the best way and algorithm fit for this problem is. Thus, the above-addressed techniques are broadly tested in the experiments, and then we select out the best setting and strategies based on the results.

4.3. INSTANCE LEARNING

4.3.1. SOLUTION PROBABILITY CLASSIFICATION

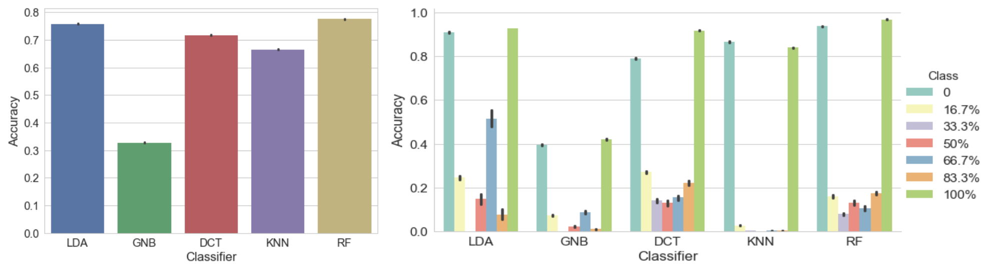
The goal of instance learning is to predict an instance's solution probability based on the instance features. Briefly, six solutions are generated resulting in seven labels from solution probability 0%, 16.7% to 100%. Table.4.1 counts the instances number in each class of all problem sizes. It can be easily seen that most instance are labeled with 0% and 100%, and much less data are in the other five classes. This will cause a foreseeable situation in classification that the accuracy will be high for the two rich classes and low for the other classes.

Table 4.1: Instance Number Statistics

Instances with Feasible Solution Probability						
0%	16.7%	33.3%	50%	66.7%	83.3%	100%
13570	3453	1571	1096	1023	1547	17105

Number of instances for each class among all datasets

This is verified by a general experiment on the whole dataset with five frequently used classifiers in their default settings, where the results are shown in Fig.4.2. For keeping the feature set more reasonable, features considering train units, arrival train and departures are not included as they will highly affect the classification results in the majority classes. Still, what fit assumptions are that the majority classes have very high accuracy while the minority classes are highly misclassified among each other. Detailed misclassifications information is shown in Table.4.2, represented by the confusion matrix from decision tree classifier. The green numbers stand for right classification of each class while the red ones represent the largest number in the misclassifications among other classes. As the instances in the first four classes hold solution probability smaller than 50%, while the instances in the rest classes have higher chance to be solved. There may exist some common patterns among the two categories respectively. Plus, the first and the last classes are the vast majority classes. Thus, it can be the reason why the first four classes tend to be classified into the first class and why the last three classes tend to be classified to the last class. However, the high misclassifications in the results do not generate high practical value. We need to depend on the imbalanced learning techniques to improve the performance.



(a) Accuracy of Classifiers (b) Class Accuracy of Classifiers

Figure 4.2: A General Test on Instance Learning

Table 4.2: Confusion Matrix - DT

		Predicted Class						
		0%	16.7%	33.3%	50%	66.7%	83.3%	100%
	0%	3308	628	239	125	81	55	42
	16.7%	547	268	117	75	50	35	48
Actual Class	33.3%	188	122	68	51	24	27	38
	50%	101	57	44	38	25	40	57
	66.7%	56	33	26	37	44	48	94
	83.3%	41	47	20	39	41	75	247
	100%	40	38	33	66	112	248	5108

4.3.2. INSTANCE DIFFICULTY CLASSIFICATION

We concluded from the analysis above, that is not an ideal way to classify into seven classes. In real life, solution probability does not limit only to these seven layers. If each instance is tested more than six times, the randomness can be better eliminated, and the solution probabilities will become more accurate. This will result in more classes. Thus, setting ranges on the solution probability to classify them into several categories will be a better and reasonable choice for both experiments and practical use.

Knowing the feature set situation, three classes are defined. Specifically, class 0% to 20% are merged as hard level instance, 20% to 80% are joined as the middle-level instance and 80% to 100% as the easy level instance. The new classes can be regarded as descriptions of the difficulty levels of instances. As class 0% and class 100% hold the vast majority of instances. Classifying into two categories are not helpful for identifying the middle-level instances. These two categories will hold very good predictions. While merging into more than three classes have no advantages than merging into three classes. On the one hand, the instances in the middle-level difficulty are already few. On the other hand, there are no existing merging rules to make a proper division.
After re-defining classes, the statistics data can be found in the following table. A highly imbalanced dataset still exists for every problem size. The first version of classification results are then shown in Fig.4.3. By adjusting higher weights to the middle class, Fig.4.4a shows a much better performance in each class. While taking a look at Fig.4.4b, small parts of both easy and hard classes are classified to middle classes. As the middle class is more flexible especially around its lower and upper bounds, the good point is that easy and hard classes do not hold high misclassifications among each other.

Size	Instances with Difficulty Level		
	Easy	Middle	Hard
All	18652	3690	17023
16	7070	6	1
17	4764	23	2
18	3214	45	2
19	1348	111	8
20	1999	937	76
21	234	1004	411
22	23	1513	8008
23	0	43	3024
24	0	7	3415
25	0	1	2076

Table 4.3: Instance Statistics

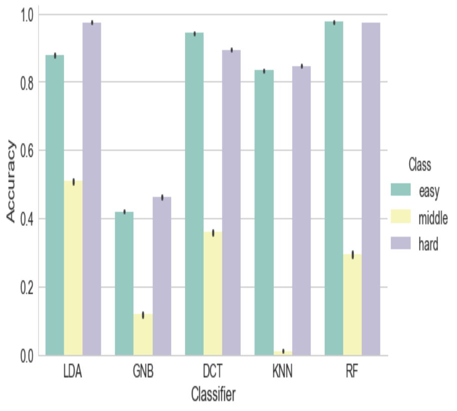
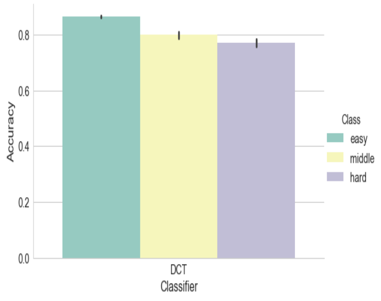
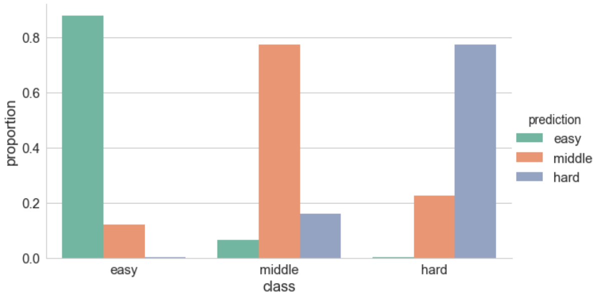


Figure 4.3: Class Accuracy of Classifiers



(a) Classifier Accuracy



(b) Classifications in Each Class

Figure 4.4: Decision Tree on Instances

As Table.4.3 suggests, training on the whole dataset can be affected by features that correlate heavily with train units. For example, the majority class in problem size 21 is the middle-level class. Classifiers can classify most of the instances with 21 train units into the middle class. In this way, the classifiers have no big prediction ability in separating instances in the three difficulty levels but separating instances in different problem size. Although specific related features are already removed, as the number of each class varies a lot in different problem sizes, this may have a substantial influence on the classifiers. For getting more accurate results on the performance on each dataset, classifications will be carried out in subsets in the following experiments.

Trainunit Size 21 Taking instances with size 21 as a concrete example, Table.4.4 gives the average class accuracy among the five classifiers. Due to the fact that middle class is the majority class with much more data than other classes, all classifiers hold poor accuracy for the other two classes. Decision tree classifier is slightly better. By multiple experiments with adjusting settings and using different resampling techniques, LDA classifier is selected to represent the performance for problem size 21. As we can observe, improvements of accuracy on the minority classes sacrifice the accuracy of the majority class. One of the reason is that both easy and hard classes have a large overlap with the middle class while they are distinguishable from each other easily. Thus the improvements in these two classes will definitely decrease the accuracy of the middle class. Table.4.5 gives the confusion matrix for this classifier. Most misclassifications occurs on easy class with middle class and hard class with middle class.
In Fig.4.6, we visualize the importance of features provided by random forest classifier. We can conclude that task-related features have dominant importance in this classification. However, we observe that high std values exist in some of the task-related values. This indicates that these task-related features are good individual criterions. We can say that different importance orders of these features can be reached following the different first criterion that choosed.
In conclusion, although the performance is improved by trying various machine learning techniques, the prediction power is not high enough. During the trade-off of the performance of the three classes, there are two situations. The middle class can hold very high accuracy, while the other two classes hold very low accuracy. When the other two classes are improved to above 50% accuracy, the middle class decreases to under 50% accuracy. Neither of the results is ideal for practical use.

Table 4.4: Test results on size 20 & 21

		Class Accuracy - Easy | Middle | Hard	
	Size	20	21
Classifier	LDA	0.927 | 0.224 | 0.147	0.014 | 0.912 | 0.222
	GNB	0.885 | 0.102 | 0.004	0.019 | 0.888 | 0.109
	KNN	0.939 | 0.125 | 0.0	0.074 | 0.843 | 0.169
	DT	0.677 | 0.370 | 0.062	0.216 | 0.584 | 0.368
	RF	0.880 | 0.236 | 0.013	0.085 | 0.836 | 0.232

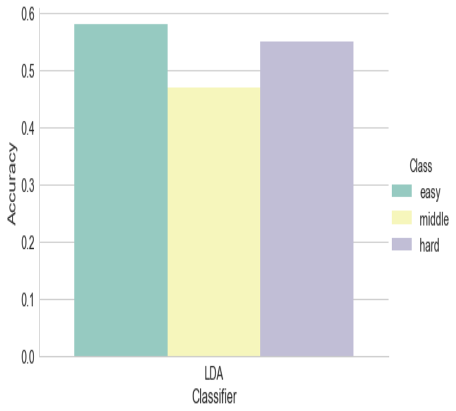


Figure 4.5: LDA Class Accuracy

Table 4.5: LDA-Confusion Matrix

		Predicted Class		
		Easy	Middle	Hard
Actual Class	Easy	135	80	19
	Middle	279	472	253
	Hard	50	139	222

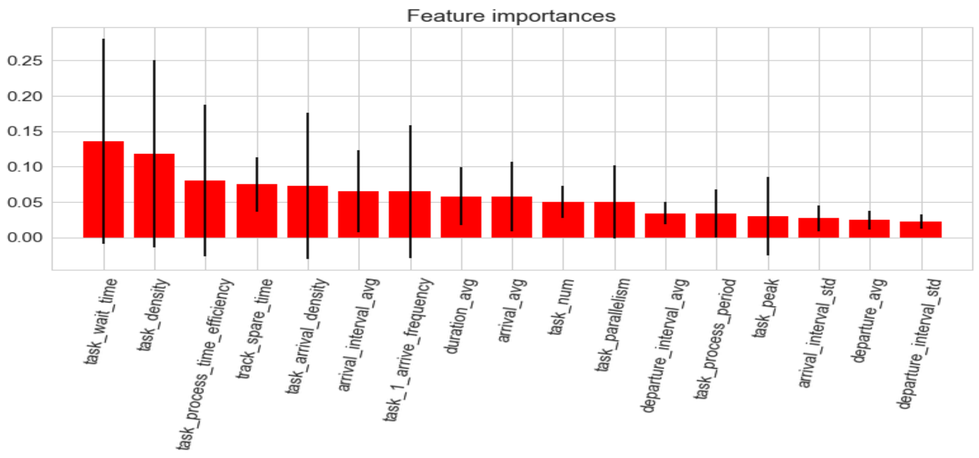


Figure 4.6: RF-Feature Importance

Other Trainunit Size For the problem size 21, the performance is slightly over random selection. While for other problem sizes, we do not train each feature set separately. Observing from Table.4.3, instance with train units 16 to 19 have the same imbalance trends. Notably, the samples in the middle class and the hard class are remarkably few. Thus, classifier training is performed on these datasets together. Considering the sample number shown in the below table, the performance seems to have good classification ability. The same way is applied to feature sets of problem size 22 to 25. Again, as the smallest class have too few data, we consider the performance is good enough to separate the middle class and the hard class.

Table 4.6: Classification results in Easy | Middle | Hard classes

		Class Accuracy	Sample Number	Classifier
	20	0.53 | 0.55 | 0.53	1999 | 937| 76	LDA (Priors=[0.31,0.30,0.39])
Size	16-19	0.76 | 0.56 | 0.47	16396 | 185| 13	GNB (Priors=[0.15,0.45,0.4])
	22-25	0.17 | 0.64 | 0.73	23 | 1564 | 16523	LDA (Priors = [0.25,0.35,0.4])

4.3.3. RESULTS ANALYSIS

For instance learning, a large number of experiments have been performed in order to figure out the best way to do classification and the best classifiers. As the results suggest, for different problem size, the perfomance varies a lot. Notebly, the results for more balanced datasets (problem size 20 and 21), the accuracy is between good and bad. For the others, the performance is acceptable for practical use. From the view of the three classes, the accuracy seems optimmistic for the majority classes While the accuracy for the extreme-minority classes are generally low. However, it is often hard to improve the accuracy of the extreme-minority class in machine learning. In addition, different imbalanced learning techniques can not balance the perfromance over the classes perfectly. This is caused by the existance of overlap especially among the middle class with other two classes.

In conclusion, from the current feature set, predicting the difficulty of instances is not accurate. Further improvements need to be done to generate classifiers for practical use. There are two possible ways to improve the accuracy. One is to calculate features which share less value in common. The other is to try different ways of merging the classes and finding the best combination.

4.4. INITIAL SOLUTION LEARNING

Initial solution analysis aims at predicting in advance whether an initialization is likely to lead to a feasible solution. As mentioned previously, different levels of imbalance exists in every dataset. The training process is in high volume, where both popular classifiers, different parameter settings and imbalance learning techniques are involved. During the experiments, different settings and methods are crucial for getting a good performance. The training procedure follows a set of steps. Firstly, preprocessing the feature set. The methods include scaling, feature selection, feature extraction and resampling. The method with the best performance is chosen. Then a set of machine learning algorithms will be performed on the processed feature set, followed by different evaluation metrics. The following paragraphs will demonstrate machine learning algorithms' performance on these datasets. To outline the final choice of machine learning algorithms, only the critical classifiers and methods are discussed accordingly.

4.4.1. RESULTS

Problem Size 16 - 18 For datasets with instance size 16,17 and 18, the infeasible class is the minority, and the feasible class is the majority class. The quantity of data for each class is hugely different. Fig.4.1 shares a spot check on simple classifiers for the original feature set without any preprocessing for instance size 16. The classifiers use default settings without parameter adjusting. As expected, the infeasible class is not correctly predicted, except for the KNC classifier. Similar results apply for instances in size 17 and 18. By adjusting class priors of the training data, good performance can be achieved. For all three datasets, LDA shows fast calculation and good results compared to other classifiers. Fig.4.8 gives precise accuracy, AUC, class accuracy along with different class priors. For size 16, see Fig.4.8a and Fig.4.8b, the ideal classifier setting for infeasible and feasible classes is (0.6,0.4). With these class priors, each class has over 85% accuracy. Observing Fig.4.8c and Fig.4.8d for size 17 problems, the best choice of class priors is (0.55,0.45) for infeasible and feasible classes. The average accuracy for the two class is between 70% and 75% respectively. Form Fig.4.8e and Fig.4.8f, class priors (0.55,0.45) can be selected. The average class accuracy is above 75% for both two classes.

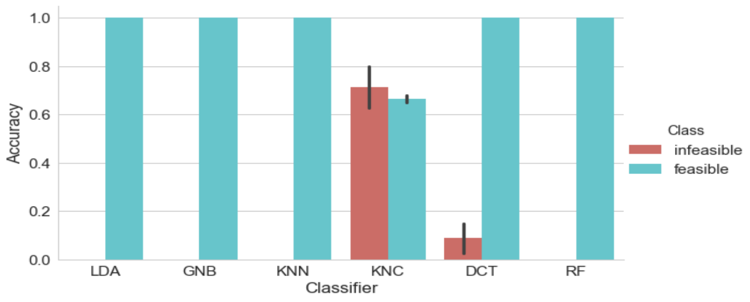
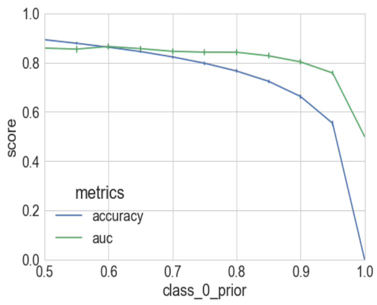
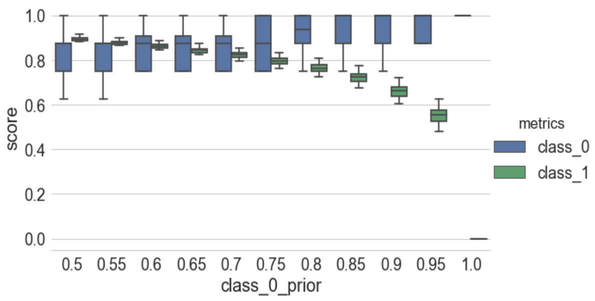


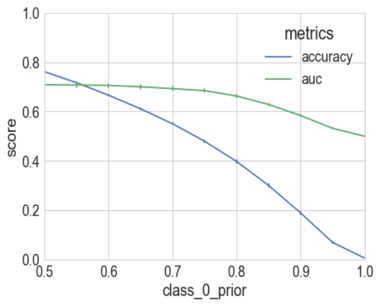
Figure 4.7: (Size 16) Average Class Accuracy with Classifiers in default setting



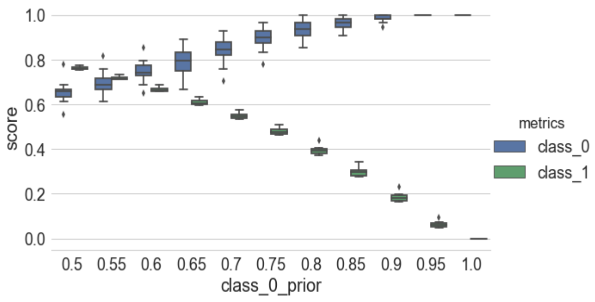
(a) (Size 16) LDA - Accuracy & AUC



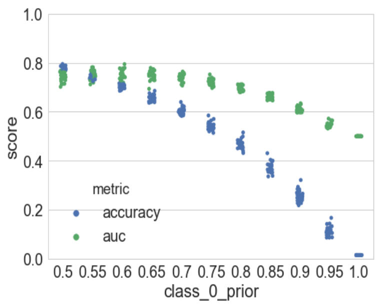
(b) (Size 16) LDA - Class Accuracy



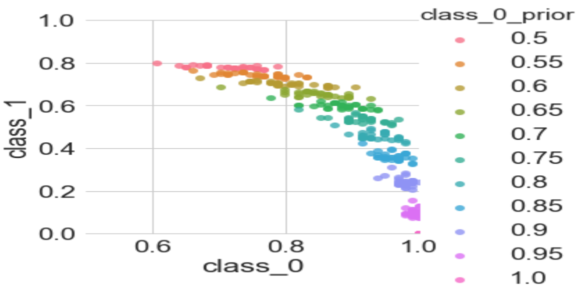
(c) (Size 17) LDA - Accuracy & AUC



(d) (Size 17) LDA - Class Accuracy



(e) (Size 18) LDA - Accuracy & AUC



(f) (Size 18) LDA - Class Accuracy

Figure 4.8: LDA Performance
Among all the figures, the x-axis represents different infeasible class priors, while the feasible class priors equal $1 - class\_0\_prior$ correspondingly.
20 repeat tests are performed on each setting.
The subfigures plot the evaluation results along with different settings.

Problem Size 19 For problem size 19, the imbalance proportion is slightly better than the above datasets. The more balanced feature set hold larger overlap areas, which brings more difficulty to the classification. Thus, several more classifiers are brought to the stage to find out a good learning result, including svc, mlp, ensembled classifiers and combined classifiers. By roughly adjusting the class weights according to the imbalance level in the dataset, Fig.4.9 depicts a sanity check on the performance generally. Four classifiers seem promising and are selected out to test further. The further results are stored in Table.4.7. Again, LDA is the final choice with the highlighted results and parameter settings. The best results and settings are marked green in the table.

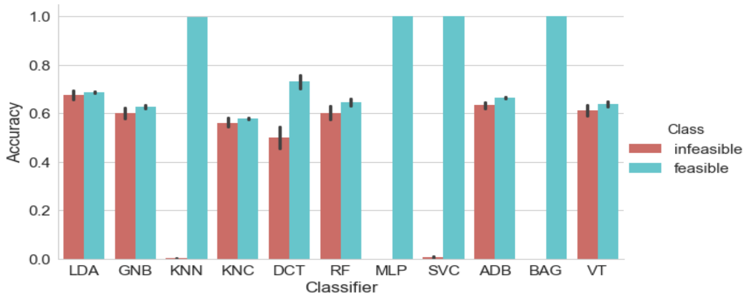
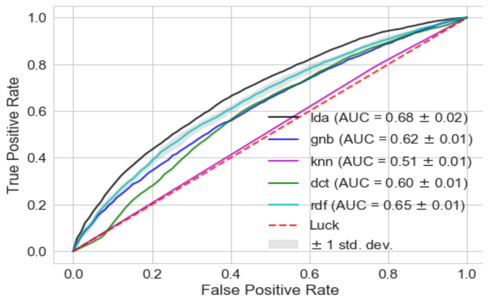


Figure 4.9: Average class accuracy for multilple classifiers

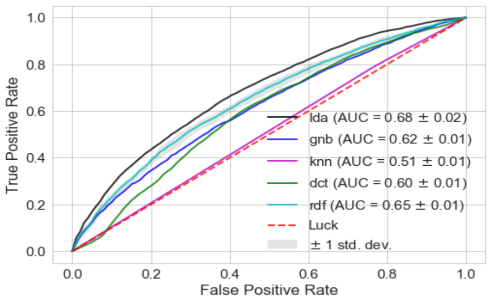
Table 4.7: Best performance for problems of size 19

	Estimation Metrics				
	Accuracy	Auc	Class_0_Accu	Class_1_Accu	Setting
LDA	0.70338	0.70545	0.70850	0.70239	Priors=[0.52,0.48]
RF	0.66556	0.65266	0.63795	0.66736	Class_weight=20:1 Max_Depth=5
ADB	0.68900	0.68381	0.67790	0.68973	Weights=7:1 n_Estimator=100
VoteClf	0.63695	0.61962	0.59988	0.63937	vote weight = 1:1:1

Size 20 For size 20, each class holds over thousand solutions while the number of feasible solutions is four times the number of infeasible solutions. Fig.4.10a depicts the roc curves of several classifiers, again the LDA classifier outperforms others, with class accuracy beyond 60%. Further adjustments are made in the different class prior settings and resampling techniques. Among the experiments, the best performance can be reached as shown in Fig.4.10b, resulting in around 70% class accuracy.



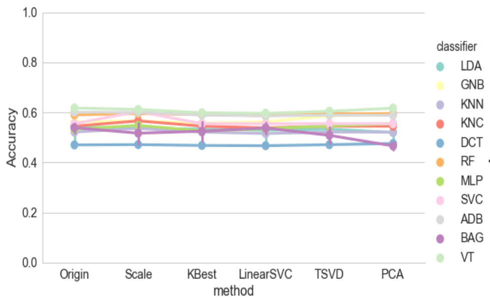
(a) LDA - AUC



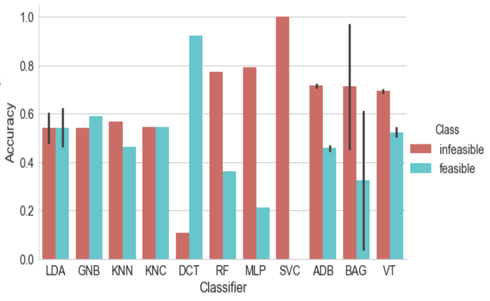
(b) LDA - AUC

Figure 4.10: Weighted LDA Performance

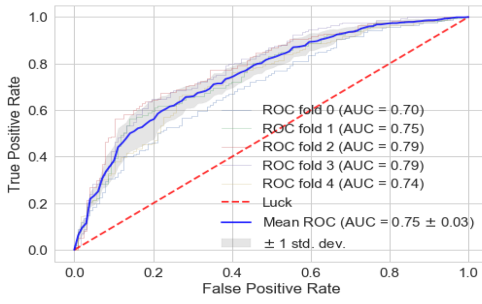
Size 21 The most balanced dataset is from instance size 21, the quantity difference between the two classes is just around a thousand. Thus, this is the most interesting dataset and it is also the most difficult dataset to achieve nice performance. To achieve good results, we also apply various preprocessing techniques to the feature set for training the classifiers. The following four figures demonstrate the results from the different steps. By applying distinct preprocessing techniques, the overall accuracy and class accuracies are calculated on a range of classifiers. From the initial tests, the original feature set and LDA classifiers are selected to process further. Fig.4.11d points out that SMOTENN method help improving performance. As in this dataset, the two classes can overlap more than in other feature sets. By firstly oversampling and then undersampling, SMOTEENN method help to reduce both the overlap and information loss. This is the main reason why this method saves the performance for LDA classifier.



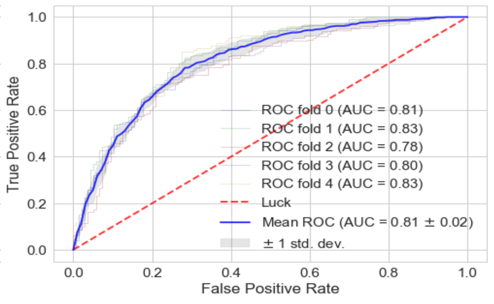
(a) Classifiers - Accuracy



(b) Classifiers - Class Accuracy



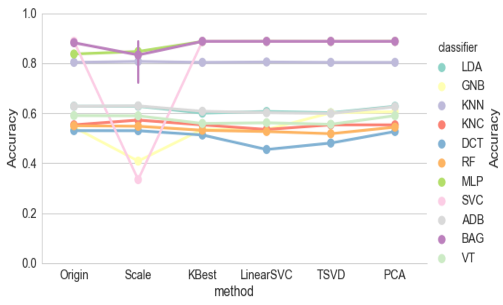
(c) LDA - ROC Curve



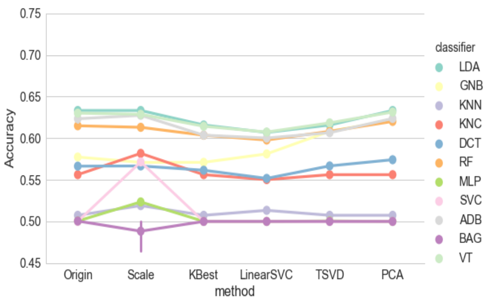
(d) LDA -ROC Curve in Smoteenn

Figure 4.11: Weighted LDA Performance

Size 22 From this size on, problems get much harder to solve. The dataset overfits in the infeasible class a lot more than the feasible class. The experiment process is similar to the above dataset. Again, SMOTEENN and LDA prove to be the best to choose. The rough results are in Fig.4.12 and the roc curve depicts the final choice in Fig.4.13.



(a) LDA - Class Accuracy



(b) LDA - Class Accuracy

Figure 4.12: Weighted LDA Performance

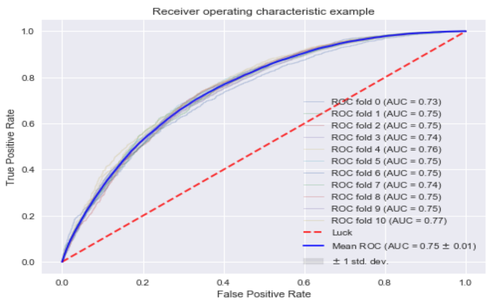
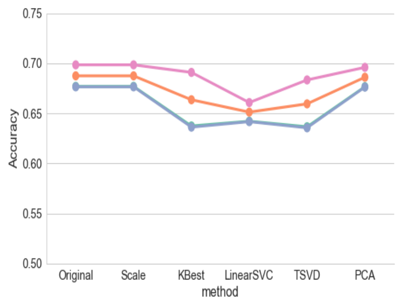


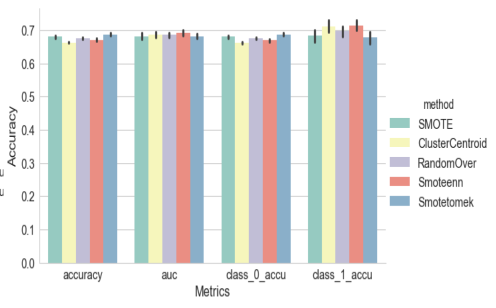
Figure 4.13: LDA -ROC Curve in SMOTEENN

Size 23 - 25 For these three datasets, the feasible class becomes the minority class with much less data than the majority class. The initial tests assigned with different costs to the two classes are performed. This works fine for the previous datasets but not for this group. This could indicate the fact that for smaller problems, the feasible class and infeasible class hold different distributions among features. While for bigger problems, features for the feasible and infeasible class have no big differences.

For the dataset of size 23, several classifiers show potentials in the initial tests where LDA is selected as it performs good and fast. Scaling, feature selection, and feature extraction are applied to the feature set, the performance is shown in Fig.4.14a. The original feature set turns out to be the best of them. Based on the original feature set, resampling methods are carried out. As the big difference of data in the two classes, sampling the minority class as the same quantity of the majority class is quite a significant change in the original data. In the tests, a reasonable ratio is considered when oversampling the minority class or undersampling the majority class. Cluster centroid method undersamples the infeasible class to 5000. Randomoversampler resamples the feasible class to 3000. SMOTE generates synthetic data for feasible class around 3000. A slightly higher weight for the feasible class is adopted according to the new imbalance ratio. SMOTEENN and SMOTETOMEK resample the feature set to a balanced level. From Fig.4.14b, SMOTEENN outperforms other classifier and is selected for practical testing.



(a) LDA - Accuracy



(b) LDA - Class Accuracy

Figure 4.14: Weighted LDA Performance

For dataset in size 24, the same testing process is used as above, and the results are demonstrated in the figure below. Again, the further tests are performed with LDA classifier. In this case, cluster centroid undersampler technique is a good fit for practical use.

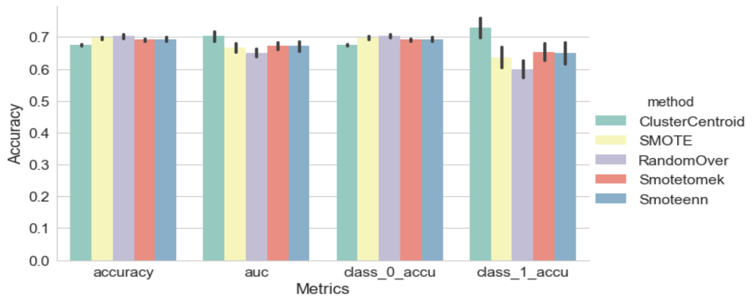
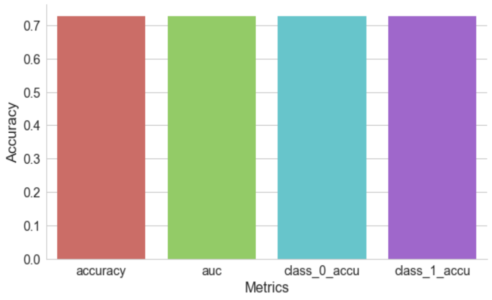
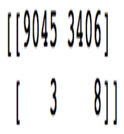


Figure 4.15: (Size 24) LDA Performance with resampling methods

For problem size 25, extremely rare feasible class exists in this dataset. Cross-validation is used for testing every feasible labeled initial solution. The following figure depicts the best results among the classifiers, LDA with class priors 0.25 for infeasible class and 0.75 for feasible class. A confusion matrix in Fig.5.3h shows detailed cross validation results.



(a) Size 25) LDA Performance


```
[[9045 3406]
 [   3    8]]
```

(b) (Size 25) LDA Confusion Matrix

Figure 4.16: LDA Performance

4.4.2. RESULT ANALYSIS

To overview the performance, we records the best classifiers, their parameter settings and class accuracies in the following table.

Table 4.8: Machine Learning Results

		Classifier	Class Accuracy (class 0 | 1)	Setting	Sample Number (class 0 | 1)
Size	16	LDA	88% | 85.7%	class priors = [0.60, 0.40]	25 | 42437
	17	LDA	73% | 70.4%	class priors = [0.55, 0.45]	163 | 28571
	18	LDA	74.7% | 74.0%	class priors = [0.55, 0.45]	284 | 19282
	19	LDA	70.8% | 70.2%	class priors = [0.52, 0.48]	540 | 8262
	20	LDA	66.3% | 67.8%	SMOTE class priors = [0.51, 0.49]	3652 | 14420
	21	LDA	65.2% | 66.7%	SMOTEENN class priors = [0.50, 0.50]	5488 | 4460
	22	LDA	65.9% | 66.4%	SMOTEENN class priors = [0.50, 0.50]	50825 | 6439
	23	LDA	67.6% | 69.5%	SMOTEENN class priors = [0.50, 0.50]	18004 | 398
	24	LDA	66.8% | 70.3%	ClusterCentroids class priors = [0.49, 0.51]	20418 | 3316
	25	LDA	72.6% | 72.7%	class priors = [0.25, 0.75]	12451 | 11

Among all the results, LDA is always the best choice, both for the performance and training speed. The experiment process provides us with some insights on the results. For the smaller size of problems (16 to 19), the performance is very good. The reason underlying can be that the pattern between the infeasible samples and feasible samples in those datasets are apparent. The features of the two classes yield bigger differences. There is no need for adopting other machine learning techniques but still generates good performance.

While for the middle problem size, the accuracies are a bit lower. In the opposite situation of the smaller problems, the solution status for the middle problems is more random, meaning most instances have both feasible and infeasible solutions. Thus, the datasets are more balanced. As the feature sets are manually defined, it is possible that we miss analyzing some more aspects in initializations. This situation can decrease the difference of features, especially for infeasible solutions and feasible solutions from the same instances. Bigger overlaps show up in these datasets, which decrease the performance.

For bigger problem size (23-25), good performance is achieved by adopting imbalanced techniques. Instances with solution probability 16.7% produce the most of the feasible solutions. This means that the feasible solutions are seldom from the same instances. Plus, these problems are more complicated than the above ones. The features of feasible samples could be in the infeasible feature areas. Making use of resampling method helps in these situations.

Regarding the resampling methods, we can conclude the following. For relatively balanced datasets, combing over and under sampling together is the best way to get better boundaries. In the results about, SMOTENN is a good choice for more balanced and overlapped datasets. For extremely-imbalanced datasets (e.g. size 16,17,18 25 in this case), there is no need for adopting resampling. Defining a proper quantity for resampling in this situation is hard. Either information loss or overfitting is a risk in this case. For the datasets in between, the quantity for each class is not extremely few. The risks for causing information loss or overfitting is much smaller than the previous kind of datasets. Adopting undersampling or oversampling to deal with majority or the minority class seems to work better.

In conclusion, we are satisfied with the performance. The classifiers are considered useful for practical use. Although the feature set is not perfect yet, the analysis proves that this feature set holds certain distinguishability between good and poor initialization. More importantly, the optimistic area among initial solutions space is predictable. We see potential in this analysis for real-life use.

5 TESTING

This chapter introduces the last part of work for this research - the testing part. The final testing is an essential component of demonstrating the machine learning work. The goal is to check whether the proposed approach in our research brings any effect to the traditional scheduling algorithms and draw conclusions to our research questions. From the experiment results shown in the last chapter, the results of initial learning part are much better compared to the results of instance learning. Due to time and storage limit, the real-life testing is focusing on the initial solution learning this time. The chapter is structured as follows. First, we introduce the framework design of our testing method. Second, we write about the classifiers we choose and the testing results. Then we show our analysis and discussion based on the testing results. In the end, we conclude the corresponding research question.

5.1. TESTING METHOD

In machine learning work, the final test is a significant demonstration of system performance. At Nedtrain, a software program is developed for researching and testing scheduling algorithms, which is introduced at the beginning of this thesis. We embed our machine learning system in the software program to check the performance.
Observing the machine learning results from the last chapter, the performance of classifying instances are not good enough for practical use. Thus, we focus on the test of the initial solution features. The testing system of this research interacts with the algorithm module and the database, detailed in Fig.5.1. The process is that when the algorithm accepts an instance, it generated several initial solutions. Then inputting the initial solutions' information into the feature generator computes initial solution features. Afterward, the learning system can immediately output the results of the feature sets using the pre-trained classifiers. Once an initial solution is classified as feasible, it continues the calculation to generate a final solution. If not, no further calculation is processed. The number of initial solutions is adjustable. The ideal number of initial solutions for testing is unknown. In this test, we use six initial solutions for all problem sizes as our testing setting.

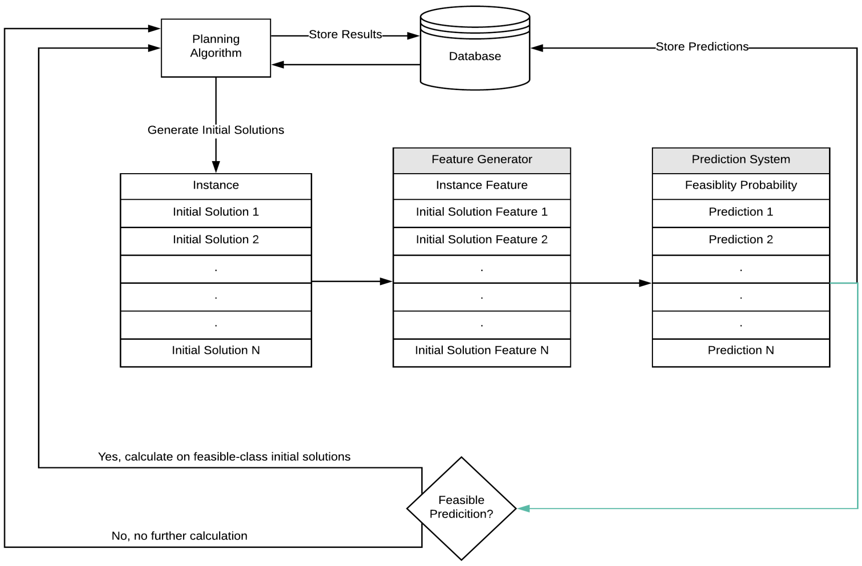


Figure 5.1: Testing System

5.2. TESTING SET

From the previous section, LDA is the classifier that yields best performance. However, in practical use, the LDA package is incompatible with the testing environment in the company. Gaussian classifiers rank the second among all the classifiers, which yield the advantage of fast training and classification speed. Thus, we implement the Gaussian classifiers in this test case.
This test contains a number of instances with six initial solutions for each. The exact tested instances and initial solution numbers are shown in Table.5.1. At least three hundred instances are tested for each problem size. As the amount is manually controlled, the number varies a lot, and the data for problem size twenty-four are not gathered properly. Fig.5.2 gives an overview of the classification results among the initial solutions. The proportion of positive and negative classifications satisfies the expected trends in different instance sizes. A more specific histogram of the positive class probability is shown in Fig.5.3. The trends for smaller sizes and bigger sizes are more stable, while the middle size problems hold more averagely distributed positive probability. The proportion of further calculated initial solution is decreasing along with the increasing of problem size.

Table 5.1: Test Num per Problem Size

	16	17	18	19	20	21	22	23	24	25
instance	2170	416	416	1243	2557	341	1248	1786	3	2145
initial solution	13020	2496	2496	7458	15342	2046	7488	10716	18	12870

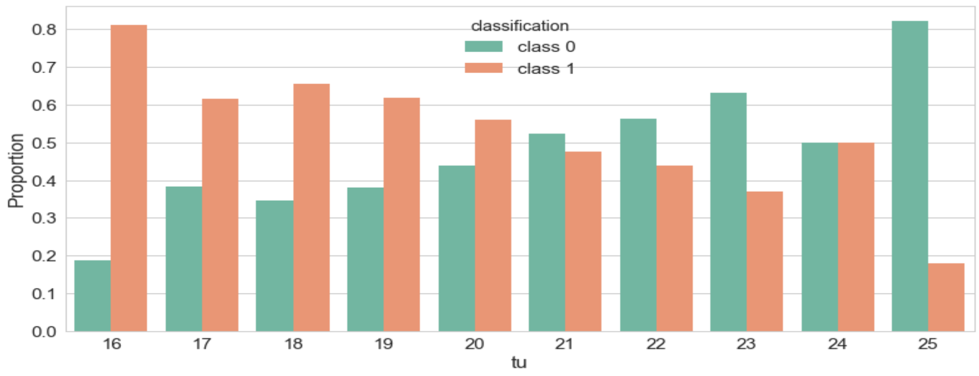
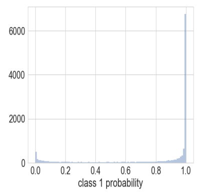
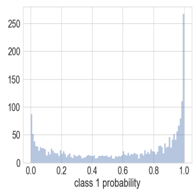
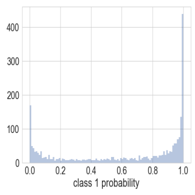
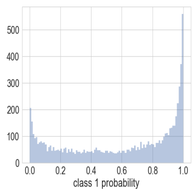


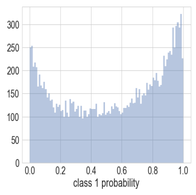
Figure 5.2: Test 1

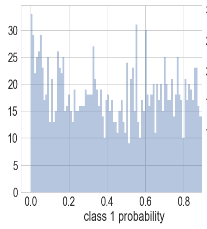

(a) TU 16

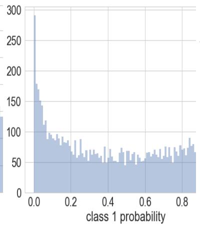

(b) TU 17

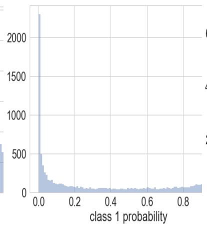

(c) TU 18


(d) TU 19


(e) TU 20


(f) TU 21


(g) TU 22


(h) TU 23

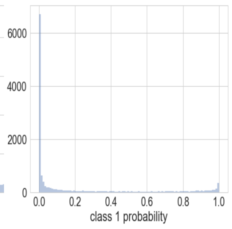
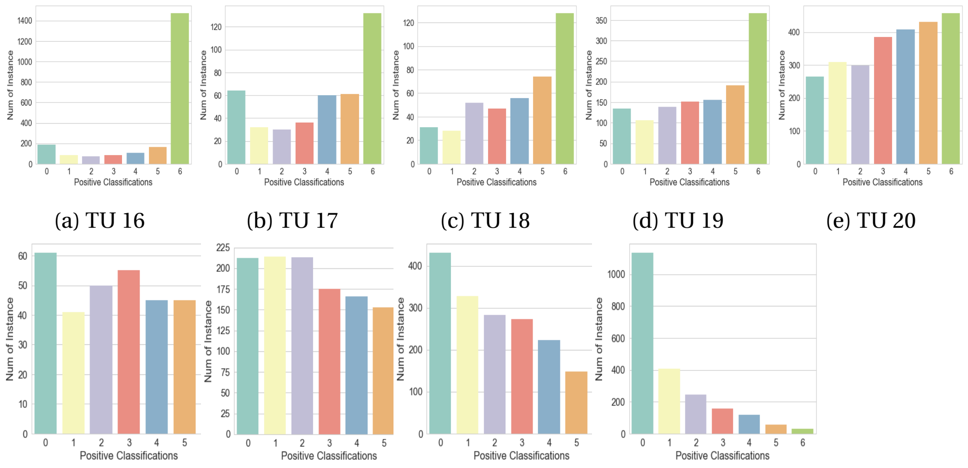

(i) TU 25

Figure 5.3: Class 1 Probability Histogram

Fig.5.4 shows how many instances hold positive classified initial solutions. The x-axis represents the number of initial solutions classified as positive in one instance while the y-axis tells about the instance distributions along this criteria. As further calculations only continue on positive-classified initial solutions. The instances with no positive initial solutions stop there. For bigger size problem, there are always more instances stop calculation at the stage right after generating bad initial solutions. As the classifiers are not perfect, there are always misclassifications. Especially for the very imbalanced dataset, the experiments from the last chapter show that the classifiers tend to classify a bit more initial solutions as negative for very small size problems (e.g.17,18) and a bit more initial solutions as positive for very large size (e.g.24,25). Thus, for these very imbalanced datasets, some instances miss calculation for small sized problems, and some redundant instances are calculated for big size problems. However, if the classifier is perfect, for instances with none potential initial solutions, this method can save a lot of time by avoiding calculating on those instances.



(a) TU 16 (b) TU 17 (c) TU 18 (d) TU 19 (e) TU 20

(f) TU 21 (g) TU 22 (h) TU 23 (i) TU 25

Figure 5.4: Instance Num vs. Positive Initial Solution Number
X-axis represents the number of initial solutions classified as positive in an instance. Y-axis represents number of instance along this criteria

5.3. INITIAL SOLUTION LEVEL ANALYSIS

Since only high positive probability (above 0.5) initial solutions are calculated to further, it helps to save a lot of time, especially for harder instances. The following table shows the proportion between the num of feasible final solutions and the number of all completely calculated initial solutions. For comparison, the table consists of two groups of data. The first group of data is from the dataset we collected for analysis use. The second group of data is newly generated data for testing when embedded with machine learning system. From the results, different level of improvements can be seen in different instance sizes. For size 16 and 17, the instances are easy to solve. There is seldom space remaining for improvements. We observe the increased proportion is from size 18 and the increment is getting bigger along with problem size. Two advantages show up from the results. One is that calculations on the selected initial solutions avoid spending time on the unpromising calculations. The other is that these positive classified initial solutions improve the solution probability.

Table 5.2: Testing Results - Initial Solution Level

Size	Training Data			Testing Data			Proportion
	Calculated	Feasible	Ratio	Calculated	Feasible	Ratio	Growth
16	42462	42437	99.9%	10566	10558	99.9%	0%
17	28734	28571	99.4%	1534	1525	99.4%	0%
18	19566	19282	98.5%	1635	1620	99.0%	0.5%
19	8802	8262	93.8%	4612	4457	96.6%	2.9%
20	18072	14420	79.7%	8605	7250	84.2%	5.6%
21	9948	4460	44.8%	975	492	50.4%	11.3%
22	57264	6439	11.2%	3267	455	14.0%	12.5%
23	18402	398	2.1%	3947	138	3.5%	66.7%
25	12462	11	0.08%	2294	5	0.2%	150%

5.4. INSTANCE LEVEL ANALYSIS

Excepting for the improvements on the initial solution level, we interest more in whether this approach brings benefits in solving more instances.

5.4.1. COMPARING WITH TRAINING DATASET

When collecting data, each instance is calculated six times. Intuitively, more instances can be solved with multiple tries. In real life, it is impossible for providing enough time and storage to solve each instance multiple times. To decrease the influence of six calculations to make the comparison more reasonable. We take the difficulty level of instances into consideration. We adopt a weighted count on the instances with feasible solutions. We use $\sum_1^{num(instance)} Feasibility_i * Solution\_probability_i$ to calculate the number of instances that have been solved. Table.5.3 shows the situation of solved instances in the two groups of data. The decreasing trends in size 16 and 17 are trivial. We can easily observe the ratio growth since size 18. It seems that these positive classified initial solutions increase the chance for finding solutions for the instances that can be solved.

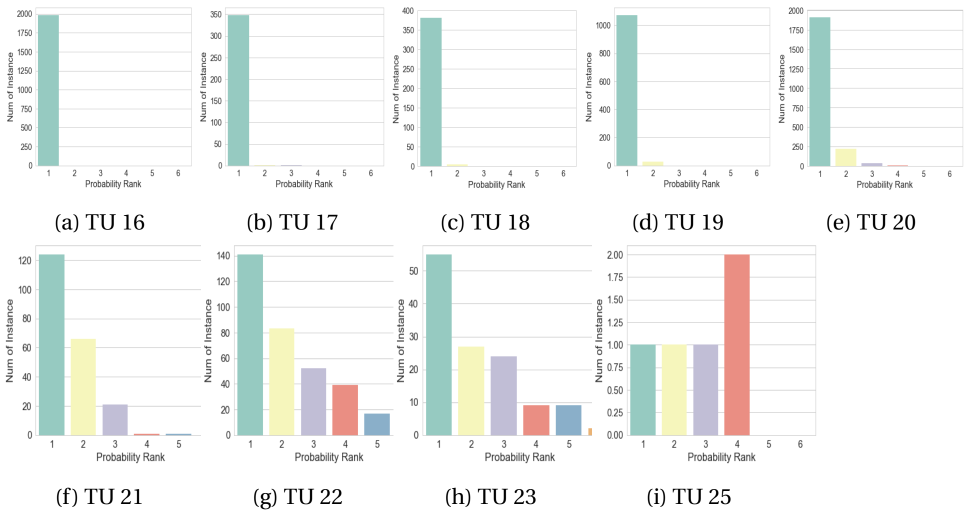
Table 5.3: My caption

Size	Training Data			Testing Data			Proportion
	Calculated	Weighted Feasible	Ratio	Calculated	Weighted Feasible	Ratio	Growth
16	7077	7072.83	99.94%	1984	1982.33	99.92%	-0.01%
17	4789	4761.83	99.43%	335	331.60	98.99%	-0.04%
18	3261	3213.67	98.55%	368	364.45	99.04%	0.4%
19	1467	1377.00	93.87%	1108	1062.37	95.88%	2.1%
20	3012	2403.33	79.79%	2292	1871.30	81.64%	2.3%
21	1649	734.33	44.53%	242	111.42	46.04%	3.4%
22	9544	1073.17	11.24%	1031	131.30	12.74%	13.3%
23	3067	66.33	2.16%	1353	43.12	3.19%	47.7%
25	2077	1.83	0.08%	1010	1.28	0.13%	62.5 %

5.4.2. Comparing With Random Selection Situation

The tricky point of the above comparison lays on the two groups of instances are not the same. A good comparison often tries to keep as most part as the data the same. However, in this software program, it is easy to test on the same instance sets, but not possible to keep the same initial solutions. Thus, to demonstrate the performance more reasonable, we perform an analysis on how these promising initial solutions work out feasible final solutions and contribute to the instances.

In the following figure, we extract the instances with at least one feasible final solution. The positive class probability ranks the initial solutions. The first feasible solution for each instance counts for the corresponding rank category. Fig.5.5 demonstrates the results in each problem sizes. This result aims at showing that how the positive classified, initial solutions contribute to the corresponding instances. The ideal situation will be that the initial solution with highest positive probability in one instance lead to a feasible solution. From the figure, we observe it in most problem sizes except for problem size 25. The trends also indicate that 80% of instances get solved by the top two positive initial solutions. We consider this as a good sign for digging the value of predicting the feasibility of initial solutions.



(a) TU 16 (b) TU 17 (c) TU 18 (d) TU 19 (e) TU 20

(f) TU 21 (g) TU 22 (h) TU 23 (i) TU 25

Figure 5.5: Solved Instance Num vs. Initial Solution Rank
The initial solution are ranked by predicred probability as x-axis. The figures show statistics on which initial solution instances are solved.

The figure above shows us the trends visually. For further analysis, we list the specific numbers in the table below. If we rank the initial solutions by their predicted positive class probability, we consider the initial solutions ranked 1^{st} as the first stage. The initial solutions ranked 2^{nd} as second stage, etc. In this way, the below table tells us about on which stage the instances can be solved.

Table 5.4: Number of Solved Instance vs. Initial Solution Ranking

Initial Solution	Size								
Ranking	16	17	18	19	20	21	22	23	25
1	1983	348	381	1072	1911	124	141	55	1
2	0	1	4	26	220	66	83	27	1
3	0	1	0	0	33	21	52	24	1
4	0	0	0	0	4	1	39	9	2
5	1	0	0	0	1	1	17	9	0
6	0	0	0	0	0	0	4	2	0
sum	1984	350	385	1098	2169	213	336	126	5

We define on which stage the instances are solved as a good metric to compare the performance between before embedding our system and after embedding our system. To calculate the same metric for the system, we perform the following steps. Firstly, we calculate the solution probability for each instance and summarize its statistics along with problem sizes. However, the experimental results from the last chapter show the predictions on instances' solution probability are not ideal. We did not perform any testing using instance learning system. Thus, we choose to calculate the solution probability using the testing results from this test set. The statistics are shown in the following table.

Table 5.5: Number of Solved Instance vs. Solution Probability

Solution	Size								
Probability	16	17	18	19	20	21	22	23	25
1/6	85	32	31	118	386	65	238	115	5
2/6	73	32	51	141	355	74	79	10	0
3/6	83	34	49	150	423	39	17	1	0
4/6	107	62	56	168	402	17	2	0	0
5/5	166	61	72	191	341	14	0	0	0
6/6	1470	129	126	330	262	4	0	0	0
sum	1984	350	385	1098	2169	213	336	126	5

In Table.5.4, we rank the initial solution probability to see that on which stage, the instances are solved. This provides us the idea that whether the new system holds better ability to solve instances more and quicker. Before embedding our testing framework, the traditional algorithm generates one initial solution for each instance and tries to find a solution within the time limit. This equals the situation that the software program randomly selects an initial solution among all the generated initial solutions and continues processing of the selected one. We wonder about the instance solving the situation on each stage in this random selection manner. Below we perform an analysis to compare the two different systems. We define the same metric for the random selection situation to make the comparison reasonable. We calculate the average number of instances can be solved in a random selection manner in each stage based on the data in Table.5.5. The formula is defined as follows. $Avg\_N(Stage_i)$ represents the estimated number of

instances solved at i^{th} initial solution. $N(ins_i)$ represents the number of instances with i feasible solutions.
The average number of instances can be solved at 1^{st} stage:

$$Avg\_N(Stage_1) = N(ins_1)*\frac{1}{6}+N(ins_2)*\frac{2}{6}+N(ins_3)*\frac{3}{6}+N(ins_4)*\frac{4}{6}+N(ins_5)*\frac{5}{6}+N(ins_6)$$

The average number of instances can be solved at 2^{nd} stage:

$$Avg\_N(Stage_2) = N(ins_1)*\frac{5}{6}*\frac{1}{5}+N(ins_2)*\frac{4}{6}*\frac{2}{5}+N(ins_3)*\frac{3}{6}*\frac{3}{5}+N(ins_4)*\frac{2}{6}*\frac{4}{5}+N(ins_5)*\frac{1}{6}$$

The average number of instances can be solved at 3^{rd} stage:

$$Avg\_N(Stage_3) = N(ins_1)*\frac{5}{6}*\frac{4}{5}*\frac{1}{4}+N(ins_2)*\frac{4}{6}*\frac{3}{5}*\frac{2}{4}+N(ins_3)*\frac{3}{6}*\frac{2}{5}*\frac{3}{4}+N(ins_4)*\frac{2}{6}*\frac{1}{5}$$

The average number of instances can be solved at 4^{th} stage:

$$Avg\_N(Stage_4) = N(ins_1)*\frac{5}{6}*\frac{4}{5}*\frac{3}{4}*\frac{1}{3}+N(ins_2)*\frac{4}{6}*\frac{3}{5}*\frac{2}{4}*\frac{2}{3}+N(ins_3)*\frac{3}{6}*\frac{2}{5}*\frac{1}{4}$$

The average number of instances can be solved at 5^{th} stage:

$$Avg\_N(Stage_5) = N(ins_1)*\frac{5}{6}*\frac{4}{5}*\frac{3}{4}*\frac{2}{3}*\frac{1}{2}+N(ins_2)*\frac{4}{6}*\frac{3}{5}*\frac{2}{4}*\frac{1}{3}$$

The average number of instances can be solved at 6^{th} stage:

$$Avg\_N(Stage_6) = N(ins_1)*\frac{5}{6}*\frac{4}{5}*\frac{3}{4}*\frac{2}{3}*\frac{1}{2}$$

Following this formula, we make the same metric for the two situations need to compare. The results for the random selection situation is shown in the table below.

Table 5.6: Average Number of Solved Instance vs. Stage

Stage	Size								
	16	17	18	19	20	21	22	23	25
1	1760	254	270	743	1208	82	76	23	0.83
2	115	51	60	179	450	49	66	22	0.83
3	48	21	27	81	226	32	58	21	0.83
4	28	11	14	46	133	23	51	21	0.83
5	19	8	9	29	88	16	45	20	0.83
6	14	5	5	20	64	11	40	19	0.83
sum	1984	350	385	1098	2169	213	336	126	5

Comparing Table.5.6 with Table.5.4, we observe that our approach always yields more instances solved on the top-ranked initial solutions than the traditional system. Along with the increasing of problem size, the differences get even more significant. For bigger sizes (except for size 25), the number of instances solved at the first stage is reaching one time more than the ones from the traditional system. This proves the efficiency of our approach in finding feasible solutions.

5.5. DISCUSSION

5.5.1. CONCLUSIONS

The above analysis demonstrates the improvements brought by embedding the machine learning system in the traditional scheduling algorithm in our case. We analyze from both initial solution level and instance level. We perform comparisons between the two different datasets and also the within the same dataset. We observe different levels of improvements from the different comparisons.

The analysis results show our approach helps in increasing the probability of finding feasible solutions and avoiding valueless calculation. The absolute values observed in Table.5.4 and Table.5.6 demonstrates more instances are solved at early stages with our approach. For illustrating the results, we compare the ratio of the quantity of instances solved in our system to the ones in the traditional system. We get the following results by division of each corresponding field value of Table.5.4 and Table.5.6. The ratio demonstrates the improvement degree in each problem size. It proves that for harder instances, our system holds more apparent improvements. Notably, we hold the most significant improvement in the first stage. Additionally, much fewer instances are solved at later stages in our system.

Table 5.7

Stage	Size								
	16	17	18	19	20	21	22	23	25
1	1.13	1.37	1.41	1.44	1.58	1.51	1.86	2.39	1.2
2	0	0.02	0.07	0.15	0.49	1.35	1.26	1.22	1.2
3	0	0.05	0	0	0.15	0.66	0.89	1.14	1.2
4	0	0	0	0	0.03	0.04	0.76	0.42	2.4
5	0.05	0	0	0	0.01	0.06	0.38	0.45	0
6	0	0	0	0	0	0	0.1	0.1	0

We conclude the research question, where predicting the good initial solutions is feasible and useful in tackling the scheduling problem in railway systems.

5.5.2. LIMITATION

Despite the improvements we observed, this testing has its limitations. Firstly, the data for problem size 24 is missing. Thus, we miss this analysis on this subset. Secondly, due to the time limit, more tests on adjusting the number of initial solutions are not performed. We expect to see the more significant ratio between the number of the solved instances and the number of all instances by generating more initial solution numbers. Thirdly, not every instance is calculated in this test. From the testing perspective, we would like to perform at least one complete calculation on each instance to see if it brings a bit more solved instances. For practical use, the trained classifiers are not perfect. It is better to perform at least one try on the most promising initial solutions for each instance. Fourthly, this testing is not big enough for problem size. More data could make the results more stable.

6

Discussion & Future Work

In this chapter, we conclude our study on embedding machine learning system in traditional planning & scheduling algorithm. The chapter consists of recall of our main work, discussion on our methods, outcomes & limitations, conclusions to our research questions and suggestion on future work.

6.1. Reasearch Work Recap

This research studies the usability of embedding machine learning into traditional planning & scheduling algorithm. The research is based on the Dutch Railways system. As this is a new research topic, we have no previous work as a reference. Each part of work is a fresh exploration in this field.
We design a framework to tackle this problem. In our experiment results, we see positive results and effects of predicting feasibility on the initializations. Our work consists of several main parts.

6.1.1. Dataset collection

The datasets are collected from the existing software program at Nedtrain. The collecting process is time-consuming. One reason is that collecting enough data cost a lot of time itself. More important is that we continue adjusting the system to collect several different datasets. Once a new dataset is collected, we carry out the statistical analysis and initial experiments to understand the structure and check its feasibility to perform research. This collecting process lasts for three months until we finally decide on the dataset introduced in the second chapter. We choose the method of performing multiple runs on each instance and storing each instance with its multiple initial solutions and final solutions. In this case, six runs are carried out. Spending time on exploring different datasets helps us a lot in defining and adjusting our research questions in this new area.

6.1.2. Feature Calculation

The next step of our work is the feature calculation part. As machine learning is the main approach, we would like to combine with the scheduling algorithm, feature set counts as an important component in this work. The original dataset is large and records every detail for instance, initial solution and final solution. However, it is not expressly and distinguishable to be the input of a machine learning system. We manually define a broad category of features and perform large amounts of calculations on that. Based on the dataset we adopted, two feature sets are calculated from instance information and initial solution information respectively. These feature including statistics information, original problem's relaxation's approximate results, time window calculation outcomes, etc. More specifically, the feature set covers the extractions of information from tasks, tracks, trains, time, etc. For example, we calculated on the efficiency of performing tasks in one instance including the waiting time, the process time, the tasks that are process simultaneously. Each feature contributes to digging out nature and working ability of the shunting processes. Around 40 features are saved for machine learning use.

6.1.3. Classifier Training

The machine learning system also comprises a big part of our work. Starting with no idea which machine learning methods are good fits for the research, we have tried most of the commonly used machine learning algorithms and techniques. Imbalanced learning and multiple instance learning are also involved in the process. During the initial experiment processes, we gradually draw a framework for performing machine learning in our work. Two aspects are tested here, learning on instances and learning on initial solutions. Realizing the imbalance nature in our dataset, we separate the whole dataset by the problem sizes. In this way, the learning process is more flexible and convincing where we adopt different methods to tackle each dataset. By continuously calculating new features and experimenting various machine learning techniques, we improve the accuracy of 10% to 15% in the initial solution learning part since the beginning. The experiment content is huge, which includes a tremendous amount of repeatable work of training classifiers by frequently adjusting parameter and adopting various imbalanced learning techniques. The outcome is not only the good results but also the combination of techniques we used for different subsets. This also provides us more insights into the machine learning techniques themselves.

6.1.4. Testing & Evaluation

The last part of work is the testing part. We run two weeks of testing on the new software program with embedded machine learning system to collect our test set. For illustrating the outcome of our approach, we analyze the test results on both initial solution level and instance level. We first compare the performance with the training set that we collected at the beginning. The test dataset shows the ability to improve the solution probability regarding the entire dataset. Although the training set is large and can represent certain trends, the instances and initial solutions are different from the testing set. Thus, we also carry out analysis on the test set itself. To illustrate the performance of old and new systems, we consider a random selection as equal to the behavior of the early program. We define the stage that an instance is solved as the metric for comparison. The better results are observed from our new system. It helps to demonstrate the ability of our approach for practical use.

6.2. Discussion

The system we designed is plantable and can be expanded to other service sites. The work we have done is initial and service site-specific. Regarding the content recalled in the last section, the data collection and feature calculation parts are based on the same service site. However, the system can be planted and expanded. The differences between the service sites lay in the infrastructure such as track distribution, resource amount and distribution, etc. This infrastructure information is stored in the database. The software program at Nedtrain fetch the infrastructure information from the database and use it to generate data and run the solving process. From data collection, updating only the infrastructure information in the database can easily gather data from different service sites. For feature calculation, some of the features regarding the tasks are calculated with the help of infrastructure information. However, it does not depend much on the complicated distribution. In the original problem's relaxation, we remove some of the constraints from the track distribution. Instead, we keep the resource distribution to perform the calculation. Thus, a pre-analysis on the availability of resources and the tracks they distributed on will support our calculation process to work in another service site.

Our framework is scalable concerning adding more problem sizes and more features. In the above work, the sizes of problem instances are manually selected to fit our experiments purpose. In the framework, we distribute each problem size as a separate module of the entire system. It is easy to add new modules and dataset from different problem sizes without disturbing the current system. The features are regularized using the basic vector format as the input of our machine learning system. New features can be directly expanded in the feature vectors to perform learning process. We believe our system is also flexible with additional re-combination of separate modules.

6.3. Conslusion

To draw conlcusions, we first recall our research question.

- Can we predict the difficulty level of the instances?
- Can we identify the optimal areas of the initial solution space as starting points? In other words, can we predict the potential of an initial solution for leading to a feasibible final solution?
- Which are the dominant features in the above classification process for scheduling and planning problem?

6.3.1. Research Question 1

The first research question is as following.

- Can we predict the difficulty level of the instances?

In the experiments of instance learning, the results show that the expected accuracy is not yet reached in every class. This means we can not predict the difficulty level of the instances. As the experimental results suggest, some of the problem sizes hold better accuracy than others. Also, for extreme minority classes, in this case, high accuracy is hard to achieve. The results are affected by both instance features and recombined training sets.

6.3.2. RESEARCH QUESTION 2

The second reasearch question is as following. This is also the main research question in this study.

- Can we identify the optimal areas of the initial solution space as starting points? In other words, can we predict the potential of an initial solution for leading to a feasibible final solution?

From both initial solution experiments, the results demonstrate the top classifiers have the ability in finding good initial solutions. From the testing and evaluation stage, we have shown the positive effects that these good initial solutions brought to the traditional algorithm. The evaluation results show the potential of good initial solutions of saving time and increasing probability of finding a feasible solution.

6.3.3. RESEARCH QUESTION 3

The third research question we discuss here is as following.

- Which are the dominant features in the above classification process for scheduling and planning problem?

In our work, we choose to directly extract features from the stored instance information and initial solution information. During the feature visualization and classifier training processes, we identify task-related features are the most dominant features for scheduling algorithms in this study. For instance features, we calculate the task-related features from both the instance information and the relaxation problem. Valuable features are the waiting time for tasks, the arrival density of tasks, the efficiency of time in solving tasks, etc. For initial solutions, the task features that relate to time are the most valuable features. We assume that the solving the tasks correctly and in time is a crucial part of scheduling problems.

6.3.4. PRACTICAL USE

The research value is not the only goal of this work. We also want to generate values for practical use. From the above conclusion, predicting on instances is not yet ready to be adopted in real-life cases. Furthur research and experiments are desired at this moment. Predicting on initial solutions shows its potential for practical use. Although the accuracy can still be improved, it achieves a useful level for real-life cases. Our first expectation of adopting this modern technique is to fasten the calculation speed and finding more feasible solutions. The evaluation process indicates the ability of the initial solution predicting system. Still, a better framework to make the best use of the predictions are need to be explored.

6.4. Future Work

6.4.1. Exploring More About Features

From the machine learning results, the misclassifications happen a lot in the overlapped areas. Similar feature values often cause the overlapped areas. From the visualization of our feature sets, there exist different sizes of such regions. More distinct feature values are always welcoming in machine learning work.

Our feature set is broad but lack of depth. We suggest on several improvements on the feature set. Adjusting calculation setting such as weights and window sizes may acquire better features. We use time window for calculating some of the features. However, the window size and settings are not deeply researched. Providing in-depth research on the time window setting may generate features with wanted values. The new features can also be acquired from better representations of feature sequences. For instance, some of the features are concluded from feature sequences. Mostly, we use average and standard deviation values to represent the whole feature sequence. This may eliminate the characteristics existed in the feature sequence. We suggest exploring other metrics to summarize those feature sequences. A third way can be combining existed features as new features. Combined features may have more expressive power than single ones.

Besides, calculating features costs time. We would like to address that the tradeoff between feature performance and extraction complexity should be studied concerning the software's performance. As mentioned, practical use is essential for this study. Within the time constraints of the algorithm, how to divide the time for the software components can yield the best overall software performance is valuable to research. We believe this also varies a lot with different difficulty levels of instances. A flexible model with different settings according to the instance difficulty can be considered.

6.4.2. Exploring Testing Methods

Although the testing results we observe is nice, more tests have not been carried out due to the expensive cost of time. In our testing, the number of initial solutions is the same for each size. Future tests can be done to trade off the with the amount of initial solution regarding the difficulty level of instances. Finding the balance point of performance and time cost of classifying more initial solutions is also an important work to be done. Additionally, the classifiers hold certain ratios of misclassifications. Our testing results indicate that for large size instances, the instances are not always solved with the best initial solutions predicted by the classifiers. More attention should be the focus on the harder instances. Future work is necessary for discovering the best way to make the biggest advantage of predictions for real-life use.

6.4.3. Exploring Deep Learning Method

Analyzing features directly from scheduling problem is not easy work. Poor analysis affects machine learning results immediately. Graphs often model those scheduling problems in research. To decrease manual analysis of features, future work can be considered in developing a graph model for representing the state of the situation. Then making use of deep learning method to learn may be a nice try. Also, it can provide comparisons with traditional machine learning algorithm.

6.4.4. Exploring Dispatching Rules

Our approach aims at solving more instances by filtering the initial solution space. However, the initial solutions are generated randomly. For difficult instances, it may require much more initial solutions to achieve the goal. Besides, our approach is focused on the local search manner. It is hard to be planted to other methods tackling scheduling problems. Existing research work has demonstrated the ability of training with the dispatching rules in scheduling problems. In this scheduling algorithms, a number of dispatching intuitive exist. Training the dispatching rules also adapts to this background. We suggest on future work on predicting the optimal dispatching decisions in the optimization process. The time complexity is always an important factor in solving real-life problems. Predicting the decisions may guide the optimization process and decrease the computation cost.

6.4.5. Exploring Improvements for Practical Use

For practical use, spending time on generating learning data is time-consuming. We suggest the next step is to develop a self-increasing learning system. For example, when expanding new service sites using the system, there is no data existed for training classifiers. It is not ideal and necessary to wait for generating enough data firstly, especially for each service site. We expect a practical system is having the ability to deal with the cold start, learn continuously and improve continuously.

REFERENCES

[1] S. Athmaja, M. Hanumanthappa, and V. Kavitha, "A survey of machine learning algorithms for big data analytics," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–4, March 2017.

[2] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," 2007.

[3] J. Yang, "Review of multi-instance learning and its applications," *Technical report, School of Computer Science Carnegie Mellon University*, 2005.

[4] J.-F. Cordeau, P. Toth, and D. Vigo, "A survey of optimization models for train routing and scheduling," *Transportation science*, vol. 32, no. 4, pp. 380–404, 1998.

[5] A. Thaduri, D. Galar, and U. Kumar, "Railway assets: a potential domain for big data analytics," *Procedia Computer Science*, vol. 53, pp. 457–467, 2015.

[6] N. Van Oort, "Big data opportunities in public transport: Enhancing public transport by itcs," *IT-TRANS, 18-20 February 2014, Karlsruhe, Germany*, 2014.

[7] A. Núñez, J. Hendriks, Z. Li, B. De Schutter, and R. Dollevoet, "Facilitating maintenance decisions on the dutch railways using big data: The aba case study," in *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 48–53, IEEE, 2014.

[8] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17–26, 2014.

[9] X. Zhang and D. Gong, "Application of big data technology in marketing decisions for railway freight," in *ICLEM 2014: System Planning, Supply Chain Management, and Safety*, pp. 1136–1141, 2014.

[10] Y. Q. Shao, R. K. Liu, F. T. Wang, and M. D. Chen, "Research on big data management for high-speed railway equipment," in *Applied Mechanics and Materials*, vol. 462, pp. 405–409, Trans Tech Publ, 2014.

[11] C. Turner, A. Tiwari, A. Starr, and K. Blacktop, "A review of key planning and scheduling in the rail industry in europe and uk," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 230, no. 3, pp. 984–998, 2016.

[12] P. Priore, D. de la Fuente, J. Puente, and J. Parreño, "A comparison of machine-learning algorithms for dynamic scheduling of flexible manufacturing systems," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 3, pp. 247–255, 2006.

[13] S. C. Graves, "A review of production scheduling," *Operations research*, vol. 29, no. 4, pp. 646–675, 1981.

[14] H. Aytug, S. Bhattacharyya, G. J. Koehler, and J. L. Snowdon, “A review of machine learning in scheduling,” *IEEE Transactions on Engineering Management*, vol. 41, pp. 165–171, May 1994.

[15] B. A. W. J. A. Blessing, “Infmss an intelligent fms scheduling system,” *World Productivity Forum and 1987 Ann. Int. Industrial Engineering Conf. Proc.*, pp. 82–88, 1987.

[16] M. J. Shaw, “Dynamic scheduling in cellular manufacturing systems: a framework for networked decision making,” *Journal of Manufacturing Systems*, vol. 7, no. 2, pp. 83–94, 1988.

[17] M. Shaw, U. Menon, and S. Park, “Machine learning methods for computer-aided process planning,” *Expert Systems and Manufacturing Designs, edited by A. Kusiak, Society of Manufacturing Engineers Press, Dearborn, MI*, 1988.

[18] R. Wysk, D. Wu, and R. Yang, “A multi-pass expert control system (mpecs) for flexible manufacturing systems,” *NBS Special Publication*, vol. 724, pp. 251–278, 1986.

[19] S. Piramuthu, N. Raman, M. J. Shaw, and S. C. Park, “Integration of simulation modeling and inductive learning in an adaptive decision support system,” *Decision Support Systems*, vol. 9, no. 1, pp. 127–142, 1993.

[20] K. Kempf, “Artificially intelligent tools for manufacturing process planners,” in *Intelligent Manufacturing, Proc. 1st Int. Conf. Expert Systems and the Leading Edge in Production Planning and Control*, pp. 131–163, 1988.

[21] P. Koton, “Smartplan: A case-based resource allocation and scheduling system,” in *Proceedings of the Case-Based Reasoning Workshop*, pp. 285–289, 1989.

[22] K. Li and J. Malik, “Learning to optimize,” *arXiv preprint arXiv:1606.01885*, 2016.

[23] A. Cassioli, D. Di Lorenzo, M. Locatelli, F. Schoen, and M. Sciandrone, “Machine learning for global optimization,” *Computational Optimization and Applications*, vol. 51, no. 1, pp. 279–303, 2012.

[24] D. Biskup and T. E. Cheng, “Multiple-machine scheduling with earliness, tardiness and completion time penalties,” *Computers & Operations Research*, vol. 26, no. 1, pp. 45 – 57, 1999.

[25] E. L. Lawler, “A “pseudopolynomial” algorithm for sequencing jobs to minimize total tardiness,” *Annals of discrete Mathematics*, vol. 1, pp. 331–342, 1977.

[26] C. Koulamas, “The single-machine total tardiness scheduling problem: review and extensions,” *European Journal of Operational Research*, vol. 202, no. 1, pp. 1–7, 2010.

[27] F. Chamroukhi, A. Samé, G. Govaert, and P. Aknin, “Time series modeling by a regression approach based on a latent process,” *Neural Networks*, vol. 22, no. 5, pp. 593–602, 2009.

[28] F. Chamroukhi, A. Samé, G. Govaert, and P. Aknin, “A hidden process regression model for functional data description. application to curve discrimination,” *Neurocomputing*, vol. 73, no. 7, pp. 1210–1221, 2010.

[29] A. Samé and H. El-Assaad, “A state-space approach to modeling functional time series application to rail supervision,” in *2014 22nd European Signal Processing Conference (EUSIPCO)*, pp. 1402–1406, Sept 2014.

[30] R. H. Shumway and D. S. Stoffer, “Time series analysis and its applications,” *Studies In Informatics And Control*, vol. 9, no. 4, pp. 375–376, 2000.

[31] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[32] A. Estabrooks, T. Jo, and N. Japkowicz, “A multiple resampling method for learning from imbalanced data sets,” *Computational intelligence*, vol. 20, no. 1, pp. 18–36, 2004.

[33] H. Han, W.-Y. Wang, and B.-H. Mao, “Borderline-smote: a new over-sampling method in imbalanced data sets learning,” *Advances in intelligent computing*, pp. 878–887, 2005.

[34] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 1322–1328, IEEE, 2008.

[35] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, “Smoteboost: Improving prediction of the minority class in boosting,” *Knowledge Discovery in Databases: PKDD 2003*, pp. 107–119, 2003.