Expert Systems With Applications 281 (2025) 127383

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Comparison of two problem transformation-based methods in detecting the best performing branch-and-bound procedures for the RCPSP

Weikang Guo [a], Mario Vanhoucke [b,c,d,*], José Coelho [b,e]

[a] School of Management Science and Engineering, Southwestern University of Finance and Economics, Chengdu, China
[b] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
[c] Technology and Operations Management, Vlerick Business School, Reep 1, 9000 Gent, Belgium
[d] UCL School of Management, University College London, 1 Canada Square, London E14 5AA, United Kingdom
[e] INESC - Technology and Science, Porto (Portugal) and Universidade Aberta, Rua da Escola Politécnica, 147, 1269-001, Lisbon, Portugal

ARTICLE INFO

Keywords:
Project scheduling
Machine learning
Problem transformation
Classification
Performance detection

ABSTRACT

The branch-and-bound (B&B) procedure is one of the most frequently used methods for solving the resource-constrained project scheduling problem (RCPSP) to obtain optimal solutions and has a rich history in the academic literature. Over the past decades, various variants of this procedure have been proposed, each using slightly different configurations to search for the optimal solution. While most of the configurations perform relatively well for many problem instances, there is, however, no known universal best B&B configuration that works well for all problem instances.

In this work, we propose two problem transformation-based machine learning classification methods (binary relevance and classifier chains) to automatically detect the best-performing branch-and-bound configuration for the resource-constrained project scheduling problem. The proposed novel learning models aim to find the relationship between the project characteristics and the performance of a specific B&B configuration. With this obtained knowledge, the best-performing B&B configurations can be predicted, resulting in a better solution.

A comprehensive computational experiment is conducted to demonstrate the effectiveness of the proposed classification models and the performance improvements over three categories of methods from the literature, including the latest branch-and-bound configurations, the state-of-the-art classification models in project scheduling, and commonly used clustering algorithms in machine learning. The results show that the proposed classification models can enhance solution quality for the RCPSP without changing the core components of existing algorithms. More specifically, the classifier chains method, when combined with the Back-Propagation Neural Network algorithm, achieves the best performance, outperforming binary relevance, which demonstrates the impact of label correlation on the performance. The experiments also demonstrate the merits of the proposed model in improving the robustness of the solutions. Furthermore, these findings not only highlight the potential of the classification models in detecting best-performing B&B configurations, but also emphasize the need for future work and development to further improve the performance and applicability of these models.

1. Introduction

The *resource-constrained project scheduling problem* (RCPSP) is a widely discussed scheduling problem in project management that has attracted many researchers' interest during the past decades (Zheng H.-y et al., 2017). The basic resource-constrained project scheduling problem assumes that a single project contains a set of activities that have to be scheduled in order to finish the project (Hartmann & Briskorn, 2022). The activities are interrelated by finish-to-start precedence relations with a minimal time lag of zero and resource constraints, and an activity cannot be preempted once it has been started, i.e., once a activity started, it cannot be interrupted until it is completed. The goal of the RCPSP is to find a precedence and resource feasible schedule to minimize the total duration of the project, known as the project makespan (S., 2019).

Due to its NP-hardness, a wide variety of solution approaches have been proposed to solve the problem, including easy priority

* Corresponding author at: Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium.
E-mail addresses: guowk@swufe.edu.cn (W. Guo), mario.vanhoucke@ugent.be (M. Vanhoucke), jose.coelho@uab.pt (J. Coelho).

https://doi.org/10.1016/j.eswa.2025.127383
Received 17 January 2023; Received in revised form 17 February 2025; Accepted 20 March 2025
Available online 9 April 2025

rules, more advance meta-heuristics, and exact algorithms. NP-hardness denotes a class of problems within computational complexity theory. A problem is classified as NP-hard if it is at least as difficult to solve as the most challenging problems in NP (nondeterministic polynomial time). Specifically, if an efficient solution exists for an NP-hard problem, then efficient solutions would also be possible for all problems in NP (Johnson, 1985; Tillmann, 2014). Most of the known branch-and-bound procedures in the literature work under only slightly different configurations, and they can therefore all solve the challenging project scheduling problem for relatively small project instances quite well. However, it was recently shown in work by Coelho and Vanhoucke (2018) that there are still differences between the different configurations. In particular, they showed in their so-called composite branch-and-bound procedure (CBP) that some configurations work better than others for some instances, and almost no configuration was unanimously considered best. The authors concluded that there is still some room for improvement by aligning the configurations better with the characteristics of the project to be planned.

In the literature, various approaches have been proposed to solve the RCPSP to achieve improvements, Messelis and De Causmaecker (2014) introduce empirical hardness models to select the algorithm for solving the *multi-mode resource-constrained project scheduling problem* (MRCPSP), and the models are used to find the mapping between problem instance features and the performance of an algorithm. The observations demonstrate that problem features affect the choice of meta-heuristics used to solve the MRCPSP. Coelho and Vanhoucke (2018) develop a branch-and-bound procedure to take into account all best performing components from literature to improve the current solutions. Guo et al. (2021) propose an automatic classification model to learn the relation between project indicators and the performance of priority rules, and the results show that the mapping can be found by the machine learning classification models. Messelis and De Causmaecker (2014) introduce empirical hardness models to select the algorithm for solving the *multi-mode resource-constrained project scheduling problem* (MRCPSP), and the models are used to find the mapping between problem instance features and the performance of an algorithm. The observations demonstrate that problem features affect the choice of meta-heuristics used to solve the MRCPSP. Bahroun et al. (2023) summarize the application of Artificial Intelligence (AI) in project scheduling and mention that most of the studies adopted Artificial Neural Networks, Bayesian Network and Reinforcement Learning techniques to tackle Project Scheduling Problems under a stochastic environment. Strahl and Gounaris (2023) propose a novel priority rule compatible with both serial and parallel schedule generation schemes that integrates shared due date information into the heuristic scheduling decisions. Golab et al. (2023) introduce the serial schedule generation scheme to schedule the project activities using an evolved convolutional neural network, which work as a tool to select an appropriate priority rule to filter out a candidate activity for the resource-constrained project scheduling problem. To better understand the existing literature related to our current work on machine learning, project scheduling, and the combination of machine learning and project scheduling, a more specific literature review, the motivation for conducting this work and our contributions are given below.

1.1. Motivation for conducting the current work

In machine learning, problem transformation is the most intuitive and straight-forward used approach to decompose the original multi-label learning task into several binary classification tasks, including binary relevance (BR) (Zhang et al., 2018) and classifier chains (CC) (Read et al., 2011). The main difference between them is that the latter takes into label correlations. In project scheduling, the branch-and-bound (B&B) procedure is one of the most widely used techniques to get optimal solutions for the resource-constrained project scheduling problem. As mentioned early, a branch-and-bound procedure is developed by Coelho and Vanhoucke (2018) to take into account all best performing components from literature to improve the current solutions. Liu et al. (2023) propose a branch-and-bound algorithm to solve the unit-capacity resource constrained project scheduling problem with transfer times. Li et al. (2024) present a branch-and-bound algorithm for the proactive scheduling problem, which incorporates a degree of anticipated variability in practical projects. Vanhoucke and Coelho (2024a) conduct a study on the usefulness of an exact method to make meta-heuristics more powerful. Vanhoucke and Coelho (2024b) present an instance transformation procedure to modify known instances of the resource-constrained project scheduling problem to make them easier to solve by heuristic and/or exact solution algorithms.

Moreover, machine learning has achieved good results on scheduling problems in recent years. In addition to the work of Messelis and De Causmaecker (2014) and Guo et al. (2021) mentioned above. Adamu et al. (2018) examine an intelligent machine learning approach to determine good initial solutions for meta-heuristic procedures to solve multi-mode resource-constrained scheduling project. Sallam et al. (2021) propose an reinforcement learning to select the best-performing multi-operator differential evolution (MODE) and discrete cuckoo search (DCS) algorithms in single algorithmic framework for solving the stochastic-RCPSPs. Guo et al. (2023) introduce a label ranking model to rank branch-and-bound procedures for the *resource-constrained project scheduling problem*, and the model aims at mapping project indicators to a full ranking list of different procedures to provide detailed information on performance ranking of procedures for a specific project. Golab et al. (2023) develop a convolutional neural network approach to solve the standard single mode RCPSP. Cai et al. (2024) introduce a framework based on reinforcement learning (RL) and graph neural network (GNN) to solve the RCPSP. Tian et al. (2024) propose a genetic programming-based hyper-heuristic (GPHH) to design heuristic rules automatically for the multi-mode resource-constrained project scheduling problem. An overview of the relevant literature is given in Table 1.

Despite these efforts on using the machine learning methods to solve the project scheduling problem, little attention has been paid to finding the relation between different characteristics of instances and the performance of branch-and-bound procedures through machine learning classification models further to improve the quality of solutions for the RCPSP. In order to fill this research gap, we propose two problem transformation-based machine learning classification models to learn the relation between project indicators and the performance of various configurations of branch-and-bound procedures to detect the best-performing configuration for the resource-constrained project scheduling problem. Once the model is built, it can be used to predict the best B&B configurations for any new, unseen problem instances.

In this paper, we will therefore use two types of machine learning classification methods to determine the best-performing configuration for a specific project. More specifically, these models will propose a certain configuration based on project characteristics, and it will be shown in an extensive computational experiment that the automatic selection of the correct configuration works better than using each method from the literature separately. We will let the method choose between 24 different configurations using a so-called multi-label classification task that is discussed in detail in Section 2. This method will try to link the relationship between the configurations and the project characteristics in two different ways, and will use six different and well-known machine learning algorithms to train this relationship on project instances known in the literature and commonly used.

1.2. Contributions of this work

The contributions of the current work are five-fold:

Table 1
An overview of the relevant literature on machine learning and project scheduling.

Category	Literature	Label correlations		RCPSP	Variants of RCPSP
		Without	With		
Machine Learning (ML)	Zhang et al. (2018)	✓	–	–	–
	Read et al. (2011)	–	✓	–	–
Project Scheduling	Coelho and Vanhoucke (2018)	–	–	✓	–
	Liu et al. (2023)	–	–	–	✓
	Li et al. (2024)	–	–	–	✓
	Vanhoucke and Coelho (2024a)	–	–	✓	–
	Vanhoucke and Coelho (2024b)	–	–	✓	–
ML+Project scheduling	Messelis and De Causmaecker (2014)	–	–	–	✓
	Guo et al. (2021)	✓	–	✓	–
	Sallam et al. (2021)	–	–	–	✓
	Guo et al. (2023)	–	–	✓	–
	Golab et al. (2023)	–	–	✓	–
	Cai et al. (2024)	–	–	–	✓
	Tian et al. (2024)	–	–	–	✓
	The current work	✓	✓	✓	–

- A novel machine learning classification framework is proposed to find the relation between project characteristics and the performance of various branch-and-bound configurations to identify the best-performing B&B configuration for a given project.
- Two problem transformation approaches, namely binary relevance and classifier chains, are used to decomposed the multi-learning task into multiple learning tasks.
- To the best of our knowledge, this is the first work to explore the relationship between project characteristics and the performance of various branch-and-bound configurations for the RCPSP. We believe that our findings will be of significant interest to the project scheduling community.
- The current work provides experimental evidence demonstrating that significant improvements in identifying the best-performing solution for the RCPSP can be achieved without changing the core components of existing algorithms.
- This work contributes to machine learning research by experimentally demonstrating, on a large set of project instances, that considering label correlations in the classifier chains method can have an impact on the model performance.

The remainder of this paper is organized as follows: Section 2 presents the problem definition with various project indicators but also gives a summary of two problem transformation methods, known as binary relevance and classifier chains and the different branch-and-bound configurations. Section 3 reviews the six machine learning algorithms used in the implementation of the binary relevance and classifier chains classification models. Moreover, this section also explains the four phases of the proposed machine learning framework. Section 4 presents various benchmark datasets used in the experiments, the designs and results of five different experiments aimed at comparing the performance of the classification models with various state-of-the-art methods from the literature. Finally, Section 5 provides the general conclusions and directions for future work.

2. Problem statement and machine learning classification

In this section, we first give a basic introduction to the resource-constrained project scheduling problem (Section 2.1). Furthermore, we explain the general concept of machine learning classification tasks (Section 2.2) and define the features of classification models (Section 2.2.1), the labels of the classification models (Section 2.2.2), as well as the classifier chains method used in the research framework (Section 2.2.3). Then, we provide the foundation of the current work (Section 2.3). Finally, we detail the extensions of the previous work in the current work (Section 2.4).

2.1. Resource-constrained project scheduling problem

The *resource-constrained project scheduling problem* can be stated as follows: A single project contains a number of activities N and the activities have to be scheduled without pre-emption on a set R of renewable resources to complete the project. Each renewable resource $k \in R$ has a finite capacity a_k per period. Dummy start activity 0 and dummy end activity $n + 1$ indicate the start and completion of the project, where the duration and renewable resource demand of the dummy activities are equal to zero. Each non-dummy activity $i \in N$ has a fixed activity duration d_i and requires a constant number of units $r_{i,k}$ of resource type $k \in R$. The activities are subject to precedence and resource constraints. A project is presented by a directed, acyclic, and topologically ordered activity-on-the-node (AoN) network $G = (N, A)$ where the set of nodes N indicates activities and the set of arcs A denotes finish-start precedence relationship between activities with a time lag of zero. A schedule S is defined as a start-time list of all the activities and is called feasible if it satisfies all precedence and renewable resource constraints. The objective of the RCPSP is to find a feasible schedule for all activities such that the total schedule makespan of the project is minimized. The RCPSP can be denoted as $m, 1|T|cpm|C_{max}$ using the classification scheme of Herroelen et al. (1999) or as $PS|prec|C_{max}$ following the classification scheme of Brucker et al. (1999).

Due to its strongly NP-hard status (Blazewicz et al., 1983), the research on the RCPSP has been extensively investigated in the past decades, resulting in a wide variety of solution procedures such as exact algorithms (mainly including branch-and-bound algorithms and mixed-integer programming formulations), heuristic algorithms using priority rules or meta-heuristic algorithms such as genetic algorithms, scatter search methods, and much more. A summary of all the methods available in the literature would lead us too far and is widely available in other papers. For a recent summary and performance status of meta-heuristic procedures, the reader is referred to Pellerin et al. (2020). A good overview of priority rules to solve the problem is given in Kolisch and Hartmann (2006). A summary of the basic features and possible extensions of the RCPSP is given in Hartmann and Briskorn (2010) and updated a decade later in Hartmann and Briskorn (2022). A complete overview of the existing branch-and-bound procedures to solve the RCPSP is given in Coelho and Vanhoucke (2018).

The goal of this work is to detect the best-performing branch-and-bound procedures available in the literature based on known values of various project indicators through machine learning classification models, which is also the subject of Section 2.2.

2.2. Machine learning classification

Classification is a supervised learning task in which the computer learns from the labeled training data and uses this learned information to classify a new object (or sample). The classification model attempts to construct a classifier by using known observed data and is used to predict the category of an unknown object. Given a data set, attributes that are used to describe each sample are referred to as "features" or "variables", and the category identifiers are referred to as "labels". The features and labels are often used as the inputs and outputs of the classification models, respectively. The goal of the classification is to learn a function (or mapping) from "features" (inputs) to "labels" (outputs). The various machine-learning techniques that implement the function are known as classifiers. Based on the number of labels to be predicted for each sample, the classification can be categorized into *single-label* classification and *multi-label* classification. *Single-label* classification is to assign a single label to an object, while *multi-label* classification is to assign more than one label (i.e. multiple labels) to an object. Compared to single-label classification, multi-label classification is more challenging. In recent years, many different methods have been developed to solve multi-label classification. Problem transformation is the simplest and most used method to deal with it, which transforms the multi-label learning task into one or more single-label learning tasks. Then the machine learning algorithm learns the single-label classifier from the transformed data (Liu & Chen, 2015).

As will be explained in the following sections, a project instance in the dataset indicates an object, and various project indicators and solutions of a branch-and-bound procedure are used as features and labels of the learning model. Obviously, features (inputs) that are used to describe the characteristics of the project instance, labels (outputs) that are used to indicate the quality of solutions given by different branch-and-bound procedures. In our learning task, it is possible that each project instance may have one or more branch-and-bound configurations that can generate the best-possible makespan, therefore, the learning task to be addressed is a typical multi-label classification task. In order to solve this task, two problem transformation methods, *binary relevance* (BR) and *classifier chains* (CC), are used to transform it into multiple binary classification tasks.

BR is a straightforward method to deal with the multi-label classification task, where an independent binary classifier is built for each label. However, the BR assumes that labels are independent of each other and ignores possible dependencies among labels, which may affect the performance of the classification models. In order to overcome the shortcomings of the BR method and maintain the advantages of its simplicity. This is also why the CC is used to solve the machine learning classification task in the current work.

CC is to transform the multi-label learning problem into a chain, which considers label dependencies effectively. The difference between BR and CC can be that BR ignores possible dependencies among labels and assumes that labels are independent of each other, which transforms the learning task into multiple *independent* binary classification tasks. CC is an extension of the BR and aims to overcome BR's issue by modeling label correlations, which transforms the learning task into a *chain* of binary classification tasks. After the learning task is transformed, machine learning algorithms (cf. six algorithms detailed in Section 3.1) are used to learn from the data to build classification models to detect the best-performing configuration to generate a feasible schedule.

The formal definition of multi-label classification is as follows: Assume $\mathcal{X}$ indicates the F-dimensional feature space, and $\mathcal{Y}$ presents the label space with Q possible labels. Multi-label learning task aims to learn a mapping h (h: $\mathcal{X} \rightarrow 2^{\mathcal{Y}}$) from the feature space to the label space based on a given multi-label training dataset $D = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq |D|\}$, where i and $|D|$ denote each instance and the total number of instances in the dataset, respectively. More specifically, for each instance in the multi-label data, $\boldsymbol{x}_i \in \mathcal{X}$ denotes the ith instance with a F-dimensional feature vector $(x_{i1}, x_{i2}, \ldots, x_{iF})$, and $Y_i \subseteq \mathcal{Y}$ indicates the set of all labels $\{y_{i1}, y_{i2}, \ldots, y_{iQ}\}$ ($y_{iq} \in \mathcal{Y}$, $q = 1, 2, \ldots, Q$) of the ith instance associated with $\boldsymbol{x}_i$, where q and $|Q|$ denote the qth possible label and the total number of possible labels in Y_i, respectively. The learned multi-label classifier $h(\cdot)$ predicts $h(\boldsymbol{x}_j) \subseteq \mathcal{Y}$ as the set of proper labels for any unseen instance j with feature vector $\boldsymbol{x}_j$.

2.2.1. Identify input features

In order to build the classification model, features (inputs) and labels (outputs) of machine learning algorithms must be specified. In the work on the automatic detection of the best-performing priority rules for the resource-constrained project scheduling problem, Guo et al. (2021) compare the multiple binary classification models with the single-label classification model. More specifically, the 12 features (project indicators, $|F| = 12$) are selected in the model, which measure either the activity characteristics, the network structure or the resource indicators of the instance. As the current work is inspired by their work, we also follow their feature selection to retain 12 project indicators as the input to the classification model. The labels were represented by a list of 39 priority rules ($|Q| = 39$) known from the literature, each serving as an individual label for the classification models. Their classification task is clearly a *multi-label* classification task since multiple labels (priority rules) might result in the same best solution (project makespan). It is obvious that the inputs and outputs of the classification model are defined as the project indicators and the solution methods used, respectively. In the current work, we follow the selection of Guo et al. (2021) and use the same 12 project indicators as inputs to the model. As we mentioned earlier, all these project indicators are defined by various characteristics that are used to describe each project instance and can be divided into three categories: the number of activities, network indicators, and resource indicators. Explanations of these project indicators are given in Table 2. However, different from the priority rules (outputs) used in the work of Guo et al. (2021), the current work will rely on the branch-and-bound (B&B) procedures as the output of the learning model to solve the RCPSP. Accordingly, the labels of the models are defined by various configurations of the B&B procedure. Since the current work builds further on the work of Guo et al. (2021), the methodology and main findings of the original work are summarized along the following lines.

2.2.2. Identify output labels

As mentioned earlier, labels can be represented by any solution method used to solve the problem instance, which now consists of different implementations of well-known branch-and-bound procedures to solve the RCPSP. The branch-and-bound procedure has been used in different ways in the literature, and Coelho and Vanhoucke (2018) have summarized and programmed them into a so-called *composite branch-and-bound procedure* (CBP). This procedure combines all well-performing components from different work in the literature, and a specific combination of different components will further referred to as a branch-and-bound *configuration*. Moreover, the B&B procedures can provide the best-possible solution for all PSPLIB instances with 30 activities to optimality (Herroelen et al., 1998), and the *composite branch-and-bound procedure* used in the current work combines all the best-performing components from previous work (Coelho & Vanhoucke, 2018). This is also why we use the CBP to solve project instances.

Since each project instance may have more than one configuration of the CBP that generates the same best makespan, the machine learning task clearly is a multi-label machine learning classification task. This is the reason why this learning task is used in the current work to solve the RCPSP. The current multi-label learning task involves the automatic detection of the best-performing B&B configurations to solve the RCPSP using six different classifiers. The purpose of this task is to rely on a set of solutions obtained after a huge computational experiment obtained from a cluster of computers to learn the mapping

Table 2
Input features for classification model.

Indicator	Definition	Literature
Activity		
Number of activities	The size of the project.	–
Network		
Coefficient of Network Complexity (CNC)	The total number of arcs over the total number of activities (nodes) in the network.	Pascoe (1966)
Order Strength (OS)	The number of precedence relations (including transitive ones) divided by the theoretical maximum number of precedence relations.	A.A. (1970)
Serial/Parallel (SP)	Measuring how close a project network is to a serial or parallel graph.	Vanhoucke et al. (2008)
Activity Distribution (AD)	Measuring the distribution of the activities.	Vanhoucke et al. (2008)
Length of Arcs (LA)	Measuring the presence of short arcs and is based on the difference between the level of end and start activities.	Vanhoucke et al. (2008)
Topological Float (TF)	Measuring the topological float of a precedence relation.	Vanhoucke et al. (2008)
Length of long arcs I_5	Measuring the presence of long arcs rather than the short arcs used in the LA.	Vanhoucke et al. (2008)
Resource		
Resource Factor (RF)	The average number of resource types required by an activity.	Pascoe (1966)
Resource Use (RU)	The number of resource types needed by each activity.	Demeulemeester et al. (2003)
Resource Strength (RS)	The combination of network topology information and resource demand, it is still controversial. More details can be found in references.	Cooper (1976)
Resource Constrainedness (RC)	The average quantity of each resource type used by all activities divided by the availability of that type.	Patterson (1976)

between project indicators (features) and the performance of branch-and-bound configurations (labels). The feature space $\mathcal{X}$ is defined as 12 project indicators ($|F| = 12$) of each instance i in D (Table 2). The label space $\mathcal{Y}$ is defined as the makespan obtained by $|Q|$ configurations for the instance i. During the training phase, features and labels are used as inputs to the classification models. Then machine learning algorithms are used to learn the mapping between features and labels to build the classifiers. Once the relations have been obtained, these models can be used to predict whether a configuration performs best based on the obtained project indicator values (i.e. the calculation of various project indicators).

Selection of branch-and-bound procedures: In the work of Coelho and Vanhoucke (2018), various components are introduced to implement various B&B configurations. However, after a preliminary computational experiment, we found that the solutions obtained by some configurations are almost identical, which would not add much value to the training process. We chose not to integrate all the possible configurations in the branch-and-bound procedure. Instead, we have selected the ones that perform better among all possible configurations, which resulted in 24 configurations, and hence, $|Q| = 24$. Finally, our procedure consists of one of two search strategies, 3 branching schemes, 2 branching orders, and 4 composite lower bounds. The corresponding between 24 configurations and the combinations of different components can be found in Appendix C. An overview of different components used in the B&B is given in Table 3 and the brief explanations are given below.

- Search strategy: This search strategy determines the selection of nodes to branch from in the B&B search toward an optimal solution. In our procedure, either the upper bound strategy (UBS) or minimum lower bound strategy (LBS) is used. More specifically, for the calculation of the upper bound (UB), only the UBS was used, and for the calculation of the lower bound (LB), only the LBS was used.
- Branching scheme: In the search algorithm, this scheme is designed to determine the way the nodes at each level of the tree are constructed. In our procedure, three different branching schemes are used during the search, including the parallel branching scheme (PAR), the serial branching scheme (SER), and the activity start time branching scheme (AST).
- Branching order: In the B&B tree, once a node has been selected for branching, the order of its child nodes needs to be determined based on some criteria for further branching. In our implementation, two versions of the node selection are used: the activity ID order (AID) and the best lower bound (BLB).
- Composite lower bound strategy (CLB): This strategy is proposed by Coelho and Vanhoucke (2018) and assembles the *speed* and *quality* of the lower bounds into this composition. Most lower bounds come from the work of Klein and Scholl (1999), sometimes with minor improvements. More specifically, 13 lower bounds have been implemented, but using all of them would require too much time on each node of the B&B tree. Therefore, the authors designed an approach in which some combinations of lower bounds are used from the list of 13 lower bounds, and this list can be dynamically updated along the search. In our work, we decided to keep four well-performing CLBs, for which details can be found in the original paper, and the ID number of the CLB used in the original work indicates the number of LBs used in this strategy. These four composite lower bound strategies include CLB0, CLB4, CLB8, and CLB12. The definitions of these strategies and the motivation for their selection in our procedure are given below:

  CLB0 (LB_{cp}) consists of only the critical path lower bound, which is not only the easiest and fastest, but also the one used in all branch-and-bound procedures.

  CLB4 is defined by CLB0 plus the other four LBs (the critical sequence lower bound (LB_{cs}), the critical capacity lower bound (LB_{cc}), the basic version of the node packing lower bound (LB_{np0}), and the incompatible pairs lower bound (LB_{ip0})). CLB4 consists of the critical path lower bound extended with four other fast and efficient lower bounds, among others the resource capacity lower bound. Some of the procedures in the literature make use of this extended lower bound, and therefore, the CLB4 resembles some of the more advanced procedures in the literature.

  CLB8 is defined by CLB4 plus other LBs, including the incompatible triplets lower bound (LB_{ip1}), one parallel machine-based lower bound (LB_{pm1}), and two extensions of the node packing lower bound (LB_{np1} and LB_{np2}).

  CLB12 is defined by CLB8 plus other LBs, including the lower bound found by *reduction by time periods* (LB_{tp}), the lower bound found by *reduction by core times* (LB_{ct}), the lower bound found by *reduction by precedence* (LB_{pr}), and an extended version of the machine lower bound (LB_{pm2}).

Table 3
Output labels for classification model.

Component	Abbreviation	Description	Study
Search strategy			
Minimum lower bound strategy (LBS)	L	Starts with an initial lower bound (LB) that is the best-found value for the 13 lower bound calculations discussed in Coelho and Vanhoucke (2018). A virtual UB set to the current LB+1, and the upper bound strategy (UBS) is used to find a solution. If a solution is found, it is the optimal solution. If the search tree is fully explored without finding a solution, a new lower bound LB +1 is found, and the process is repeated.	Coelho and Vanhoucke (2018), Patterson and Huber (1974)
Upper bound strategy (UBS)	U	Enumerates all non-dominated nodes of the search tree until no more nodes can be explored.	Patterson and Huber (1974)
Branching scheme			
Activity start time (AST)	A	Selects one activity to branch, and then branches on the several activity start times in its time window.	Talbot and Patterson (1978)
Parallel branching (PAR)	P	Selects the activities that can be scheduled in parallel at each time instant and relies on a time incrementation approach that iteratively increases the time pointer in the schedule and searches for sets of activities that can be scheduled.	Christofides et al. (1987), Stinson et al. (1978)
Serial branching (SER)	S	Selects the activity that can be scheduled at its earliest start and relies on a so-called activity incrementation scheme that iteratively schedules eligible activities in the partial schedule.	A. (2000)
Branching order			
Activity ID (AID)	A	Selects the activity with the lowest number first.	A. (2000)
Best lower bound (BLB)	B	Selects the lowest lower bound values as the node for further branching.	Demeulemeester and Herroelen (1992)
Composite lower bound			
CLB0	0	The critical path lower bound LB_{cp}.	Almost all individual lower bounds are reviewed by Klein and Scholl (1999) and their composite strategies are proposed by Coelho and Vanhoucke (2018)
CLB4	4	Extends CLB0 with LB_{cp}, LB_{cp}, LB_{cp}, and LB_{cp}.	
CLB8	8	Extends CLB4 with LB_{cp}, LB_{cp}, LB_{cp}, and LB_{cp}.	
CLB12	12	Extends CLB8 with LB_{cp}, LB_{cp}, LB_{cp}, and LB_{cp}.	
#B&B configurations	24		

It is obvious that CLB8 and CLB12 make use of 8 and 12 lower bounds, respectively. As far as we know, no existing procedures in the literature make use of all these configurations, despite the high quality of the lower bounds. One of the reasons is that the calculations of these bounds take too much time, although (Coelho & Vanhoucke, 2018) have shown that it is sometimes worth calculating them as some, but not all nodes of the branch-and-bound tree. Given their performance, we also have added them to our procedures.

2.2.3. Classifier chains method

Binary Relevance is a well-known and popular problem transformation method to convert the multi-label classification into multiple independent binary classification tasks. Such conversion is relatively easy and shows low computational complexity, but it assumes that labels are not correlated. An alternative method, known as the Classifier Chains (Read et al., 2011) works in a very similar way but takes into account the correlations between labels and converts the task into a *chain* of binary learning tasks, i.e. each label still gets one classifier, but the classifiers are linked along a *chain* to include label correlations. The feature space for each binary model is extended with the "+ "/"−" label relevance of all previous binary classifiers to form a classifier chain. Consequently, each classifier in the chain is responsible for learning and predicting the binary association of the qth label given the feature space, augmented by all prior binary relevance predictions in the chain. In doing so, CC models label correlations by passing information between classifiers along a *chain*. Read et al. (2011) have shown that CC improves convincingly over BR, and overcomes the label independence problem of BR while retaining the relatively low time complexity of BR (cf. Section 2.2), the results indicate that label correlation can improve the performance of the classification model and obviously has a place in multi-label classification. Overall, the difference between BR and CC methods is that the former considers as an independent binary problem, while CC incorporates label correlation by assuming an order for labels and inserting previous label outputs in feature space and achieves higher performance.

Classifier Chains method used in the current work is based on the work of Read et al. (2011), where it is mentioned that the predictions of all binary classifiers in CC are cascaded along a *chain* as additional features to model inter-label dependencies, and the order of the chain itself (determined by the order of the label in each label vector) has an impact on the model. Therefore, in order to investigate the impact of different orders on the proposed classification models, we propose three scenarios to determine the order of different labels in the CC-based method. These three scenarios are related to the quality of the solutions generated by different B&B configurations, and their definitions are given along the following lines.

- Scenario 1 (S1). Random: The order of different configurations in the *chain* is based on their ID number (i.e. randomly), where each configuration is treated as a label for each classifier.
- Scenario 2 (S2). Sum: The order of configurations in the *chain* is based on the sum of the makespan of each specific configuration over all instances, giving priority to high-quality configurations.
- Scenario 3 (S3). Best: The order is based on the percentage of the best makespan of each configuration on all project instances. The best makespan is obtained by comparing the makespans of 24 configurations for a specific project. This scenario also gives priority to the high-quality configurations, but rather than taking average quality into account (Scenario 2), it ranks the configurations based on whether they are able to provide the best solution.

The comparative analysis of binary relevance and classifier chains method, as well as the three scenarios used in the classifier chains method, are given in Appendix B.

2.3. Foundation of the current work

The current work is inspired by our earlier work (Guo et al., 2021), where machine learning classification models are built to detect the best-performing priority rules for the RCPSP. Specifically, the learning task is transformed by binary relevance (BR) and multi-class (MC) methods, the former transforms it into multiple single-label learning tasks (multiple labels), while the latter transforms it into a multi-class learning task (single-label) by defining new classes. Then, the classification models built based on these two methods were compared to select the best one for the best-performing priority rule detection. In order to have a better understanding on the difference between current work and our previous work, we made use of *decision tree* as the main base classifier to implement both classification models in the illustrative example (Fig. 1). To the best of our knowledge, this example of decision trees built on BR and CC is also the first time that it has been used to detect the best-performing branch-and-bound procedures.

- Binary classification models with multiple labels (referred to as the J48ML model): In this model, BR is used to decompose the multi-label classification task into multiple *independent* binary classification tasks, which means that for each label (priority rule), a separate decision tree is built. This approach results in exactly Q decision trees during the training process, and the ultimate desire is that only a small number of decision trees in a set of Q decision trees result in a positive prediction. If that is the case, each test instance must be solved by only a small number of priority rules, i.e. only the decision tree with the best-performing priority rules that return a positive prediction among the Q decision trees.
- Multi-class classification model (referred to as the J48MC model): In this model, the multi-label classification task is converted into a multi-class (single-label) classification task by integrating the relevant labels as new classes, similar to Kazawa et al. (2005). More specifically, if multiple labels can generate the best possible makespan for a given instance, and this instance is labeled with the new class containing all these positive labels such that this instance can get a single newly defined class label. The obvious advantage is that this approach only results in one decision tree, but the number of created classes during the model training is not known in advance and can result in a huge decision tree with many newly created classes. Nevertheless, during the testing phase, each instance will make use of the one (huge) decision tree, and with the known feature values of each instance, only one leaf node will be selected, resulting in only one priority rule that will be used.

In order to highlight the difference between single-label (MC) and multiple-label task (BR and CC), the decision tree classifiers constructed based on these three approaches is given in Fig. 1. It should be noted that the three decision trees in this figure are built based on the example training dataset given in Appendix A. In this example data D, there are 16 instances ($|D| = 16$) with 3 features ($|F| = 3$) and 3 labels ($|Q| = 3$). In Fig. 1, the first (Fig. 1(a)) and second constructed decision trees (Fig. 1(b)) are based on the J48MC and the J48ML models introduced in the work of Guo et al. (2021). However, the third decision tree (Fig. 1(c)) is based on the classifier chains (CC) method that can pass label information between classifiers, which is also the topic of the current work and will be explained in Section 2.2.3. In this example, the third decision tree was constructed based on Scenario 1 (cf. Section 2.2.3).

From Fig. 1, it is obvious that the number of decision tree classifiers generated by MC (i.e. J48MC model) is only one, whereas the number of decision trees generated by BR and CC are 3 (is equal to the number of all possible labels).

In Fig. 1(a), we can see that all the leaf nodes in the decision tree represent the newly defined classes. However, in Fig. 1(b) and Fig. 1(c), the leaf nodes of each decision tree represent the binary predictions (positive "+ " or negative "−") for each label. For a specific label, if it can produce the best makespan among all labels, it is labeled as "+ ", otherwise it is labeled as "−". By comparing the decision trees in Fig. 1(b) and Fig. 1(c), we can note that the constructed classifier 1 is the same, however, the constructed classifiers 2 and 3 are different. For classifiers 2 and 3 generated by CC, their non-leaf nodes contain the predictions given by the previous classifiers for Label 1 (e.g., non-leaf node "L1" in classifier 2). Specifically, since the number of labels in the illustrative example is 3, the number of decision tree constructed based on CC is also equal to 3 (i.e., classifiers 1, 2, and 3). The order of labels in the chain is based on the ascending order of the IDs (i.e. L1, L2, and L3).

It can be easily seen that except for the first classifier (Fig. 1(a)), the results of the other classifiers (classifiers 2 and 3) are correlated with the previous ones (Fig. 1(b) and Fig. 1(c)). Once the classifiers are built based on the training dataset, then for any new problem instance, we only need to calculate their features and input the values to the ordered decision tree classification model, then, we can obtain the best-performing solution set generated by each classifier in the chain. From this example, we can know that the classifier built based on CC can model label correlations by passing the label information in the chain. Therefore, the label correlations have an impact on the prediction results. More details about the decision trees constructed by these three different methods are provided in Appendix A.

Computational results: (Guo et al., 2021) showed that the multiple labels models (multiple binary classifiers, i.e. J48ML) outperformed the single-label model (multi-class classifier, i.e. J48MC), although they observed that the former relies on more than one priority rule to schedule the project (and hence is more computationally costly during the testing), while the latter makes use of only one (i.e. newly defined class). Recognizing that such comparison is not completely fair, they restricted the number of times the priority rules are used in the model and observed that the multiple-label method still outperformed the single-label method, even when the number of positive labels (i.e. priority rules) was restricted to only a few. As a result, the multiple-label method is used in the current work.

2.4. Comparison of current and previous work

While the current work builds further on the results and insights obtained in the original work on priority rule detection automatically (Guo et al., 2021), we believe we have extended the previous work in three different ways: First and foremost, the solution methods used to solve the RCPSP are branch-and-bound procedures rather than priority rules used in the original work. This switch to an exact procedure not only makes the work more challenging (since B&B procedures often consume a lot of time before a feasible or optimal solution is reported) but also more relevant. It is true that the priority rules are very fast, while the automatic selection of the best-performing rule is only important for very big projects. However, the branch-and-bound procedures aim at finding optimal (or near-optimal in case of truncation) solutions, and the suitable selection of the best-performing B&B configuration is, therefore, more important. The details of these B&B configurations are discussed in Section 2.2.2. Second, the current work will rely on the extension of the binary relevance-based method (i.e., classifier chains) since it has been proved that the BR-based method outperform multi-class models (Guo et al., 2021). More specifically, the motivation for the selection of classifier chain method is that it not only overcomes the shortcoming of BR's label independence assumption, but still retains the advantages of the simplicity (Read et al., 2011). More details can be found in Section 2.2.3. Finally, two problem transformation methods (BR and CC) will be compared based on six different machine learning algorithms respectively. Moreover, these six algorithms are serve as base classifiers to determine how various classification models could be best implemented.

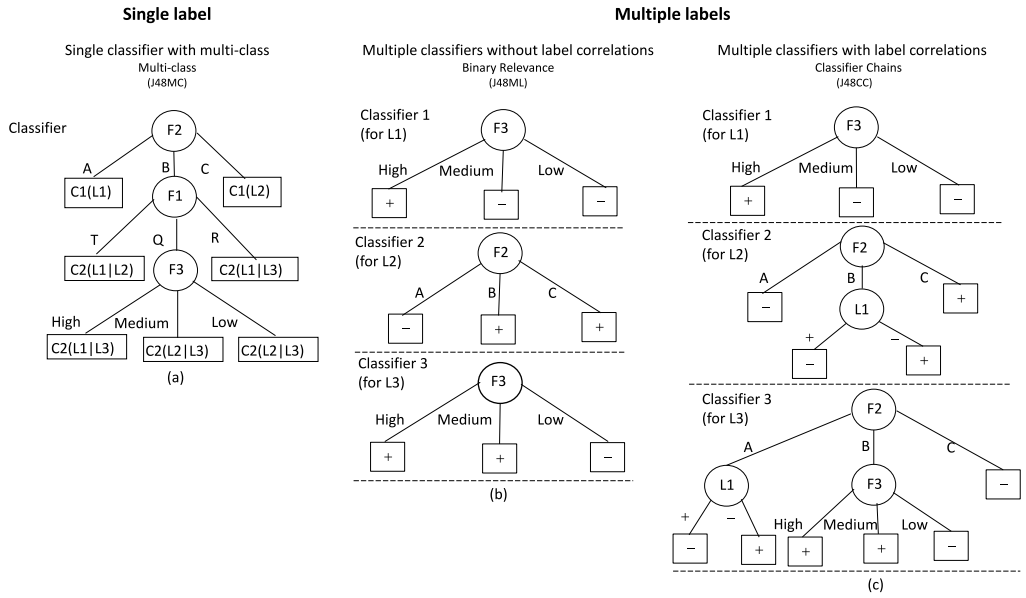

Fig. 1. Three decision tree classifiers built for three classification tasks (single-label) (a) and multiple-label (Binary Relevance (b) and Classifier Chains (c)) based on examples dataset in Appendix A.

Table 4
Differences between current work and our previous work.

Studies	Machine learning relevant		Solution methods used for the RCPSP
	Task	Method	
Guo et al. (2021)	Classification	BR[a] and MC[b]	Priority rule-based heuristics
Guo et al. (2023)	Ranking	Structured output	Branch-and-bound procedures
The current work	Classification	BR and CC[c]	Branch-and-bound procedures

[a] BR indicates Binary Relevance method.
[b] MC means Multi-Class classification.
[c] CC indicates Classifier Chains method.

As we introduced in Section 1, the current work relies on machine learning classification model to predict the best-performing B&B configuration for the RCPSP, which is different from the machine learning ranking model used in the Guo et al. (2023) and priority rules used in the work of Guo et al. (2021), respectively. In order to provide a better understanding on how the classifier chains method extends previous work, and highlight these differences between current and previous work, we summarize the differences in Table 4.

3. Methodology

In this section, we explain in detail how to construct two types of classification models based on BR and CC-based methods respectively. In Section 3.1, six machine learning algorithms are introduced, which are used as base classifiers to build classification models respectively. In Section 3.2 details the different phases of the proposed machine learning framework.

3.1. Machine learning algorithms

Once the multi-label classification task is transformed by BR or CC (Section 2.2.3), machine learning algorithms are used as to build classification models based on the transformed data. In the overall proposed machine learning framework, these algorithms play a key role to find the mapping between various project indicators (features, inputs, Section 2.2.1) and the performance of various B&B configurations (labels, outputs, Section 2.2.2). The motivation for the selection of six algorithms have been given below, in this section, we focus on providing more details on their working mechanism and their roles within the proposed machine learning framework.

Selection of six machine learning algorithms: In order to build the classification model based on BR and CC, the six popular and classic algorithms are used: Decision Tree (DT), Random Forest (RF), Back-Propagation Neural Network (BPNN), Support Vector Machine (SVM), *k*-Nearest Neighbor (kNN), and Naïve Bayes (NB). The motivation for their choice is based on the literature studies and the following reasons: First, they are the most popular and commonly used supervised learning algorithms for classification tasks. More specifically, DT has the following advantages (Farid et al., 2014): It is simple to understand and easy to implement, and it can handle large and noisy data sets. RF is one of the most powerful and popular ensemble methods and has a wide range of applications in machine learning (Nadi & Moradi, 2019). BPNN is one of the most frequently used algorithms for machine learning classification and prediction (Wu et al., 2006). SVM is a widely used machine learning technique because of its accuracy (Chen & Hsieh, 2006). *k*NN is a widely used classifier for classification because of its simplicity and efficiency (S., 2006). NB has several advantages: It is easy to use and can handle missing feature values (Farid et al., 2014). Second, they can easily be applied to our learning task transformed by CC (Read et al., 2021). Third, in the existing literature work, these six algorithms are used in their intelligent phishing website detection models (Ali & Malebary, 2020), and we follow their choice to use these algorithms in our models. More details about the mechanism of six algorithms can be found below. Since each of these six algorithms has its own characteristics, the model can be extended to solve more problems in practical applications.

Such work has, to the best of our knowledge, never been done before in the project scheduling literature. The classification model

constructed by Guo et al. (2021) is the foundation of the current work but is extended in three different ways as stated before. Guo et al. (2023) introduce a label ranking model to obtain the rank of branch-and-bound procedures for the RCPSP. The main difference between classification and ranking is that the classification models aim to detect whether a given solution method will perform well or not, whereas the ranking models aim to predict a full ranking list of the performance of the solution methods. Although these efforts have been made, existing work cannot provide information on whether a B&B configuration will perform well or not for a specific project based on the calculated project indicators. Furthermore, the latest classification model proposed by Guo et al. (2021) is based on the assumption that labels are independent of each other. Therefore, our work fills this research gap by taking label correlations into account to build the models to predict the best-performing B&B configuration for a given project.

- Decision Tree (DT): The decision tree is a well-known tree-based supervised learning method, which has been introduced by Quinlan (1986, 1987) and can be used to solve complex classification problems. A decision tree is a directed tree that contains a root node, a number of non-leaf nodes, branches, and leaf nodes. Specifically, the root node contains all the data in the training set, each non-leaf node denotes a feature test, each branch in the decision tree represents an output of a test, and each leaf node indicates a specific class label. The decision tree is built by recursively partitioning the feature space of the training data into branches (Polat & Güneş, 2009). A decision tree works on the principle of decision-making: First, a feature is selected and the logical test is performed on the selected feature. Second, a branch is created based on each outcome of the test. Third, the process is run recursively on each child node until all instances at a node satisfy a certain criterion (Chen & Wang, 2009).
- Random Forest (RF): Random forest is an effective and powerful tree-based ensemble machine learning algorithm, which has been introduced by Jo and Cheng (2014), L. (2001). The RF consists of an ensemble of multiple decision trees and is constructed based on randomly selected subsets of the training data. The output of the RF is determined by the outcome of each decision tree in the forest. The operating principle of RF is summarized as follows: First, multiple samples are drawn from the entire training set using the bootstrap sampling method. Second, multiple decision trees are constructed based on multiple samples. Third, each decision tree votes for the most popular label (or class) for a specific input, then generates the results (Wang et al., 2015b).
- Back-Propagation Neural Network (BPNN): BPNN is one of the most widely used multilayer artificial neural networks (ANNs) (Park et al., 2010), which has been applied to solve a number of problems in many fields (Wang et al., 2015a). In general, BPNN consists of three types of layers: the input layer, the hidden layer, and the output layer. Among them, the neurons of the input layer receive the information from the input or training data and then pass the information to the hidden layer. The hidden layer performs computations based on the information and provides the output to the output layer. The output layer receives the processed information and provides the final predictions for a given input. The intermediate layers between the input and output layers are called the hidden layers. Neurons between layers are fully or partially interconnected, and each connection between neurons has a weight (Sun & Tien, 2008). The inherent mechanism of BPNN is to propagate the error between the actual output value and the target value from the output layer of the network back to the input layer and minimize the error between them by updating the weights of the connections in the neural network to obtain better performance (Wang et al., 2015a).
- Support Vector Machine (SVM): Support vector machine is a statistical learning-based machine learning technique introduced by Vapnik (1999), which has become the most popular tool for machine learning classification tasks because of its good generalization capability and high performance (Lewis et al., 2006). SVM is based on an induction principle known as the structural risk minimization principle, which maps the low-dimensional data in the input space into a high-dimensional feature space (Chandaka et al., 2009). The goal of SVM is to find the optimal separating hyperplane that maximizes the margin between the classes in the data set (Zhang & Wang, 2008).
- *k*-Nearest Neighbor (*k*NN): The standard k-nearest neighbor rule is introduced to solve the classification tasks by Cortes and Vapnik (1995) and has attracted considerable interest from researchers in pattern recognition and machine learning. kNN has been frequently used in many fields because of its simplicity, intuitiveness, and effectiveness. It is based on the distance or similarity function and calculates the distance between a test sample and the training samples, then assigns the most common class of its k nearest neighbors to the test sample (Jiang et al., 2007). The working principle of kNN is to find the shortest distance between the data to be evaluated and the k nearest neighbors in the training data (Mukhlishin et al., 2017).
- Naive Bayes (NB): Naive Bayes is a simple probabilistic-based algorithm that is based on Bayes theorem. The general principle of NB is the assumption that the features are independent of each other given the class label (i.e. conditional independence assumption) (Hagenauer & Helbich, 2017). Therefore, a change in one feature does not affect other features in the classification tasks (Pawar & Gawande, 2012). Due to this assumption, the learning process of the Naive Bayes classifier can be significantly simplified when the number of features is very large (Jiang et al., 2016). For a given test sample, Naive Bayes calculates the posterior probability of each class label based on the Bayes theorem, then assigns the class label with the highest posterior probability to the test sample (Murphy et al., 2006).

  Generally, in order to solve the machine learning classification task, the classification algorithms must be specified, which can be used to learn patterns from labeled training data and make predictions for new, unseen data. The mechanism of these algorithms also makes it possible for classification models built can be easily extended to solve a wide range of problems across different fields.

3.2. Research framework

In this section, we give all details on the four different phases of the proposed classification models. The flowchart of the machine learning framework can be found in Fig. 2, which consists of four phases: data preparation (Phase I), data splitting (Phase II), model training (Phase III), and model testing (Phase IV). The rationale for structuring the machine learning framework into four phases is rooted in ensuring a systematic, efficient, and reliable process for building and evaluating predictive models. Detailed descriptions of these four phases are provided as follows.

Phase I: Data preparation. High-quality data is the foundation of any machine learning model. This phase ensures that the dataset is clean, consistent, and representative of the problem domain. The goal of this phase is to prepare all data that will be used in the experiments, including (i) calculating the feature values for each instance, (ii) labeling the values of each label as "positive" or "negative" according to the solutions given by 24 configurations, (iii) normalizing the data by a popular min–max normalization method, and (iv) removing all "easy" instances from the set. First, the 24 configurations of the CBP will be executed on all instances to obtain all possible solutions both in terms

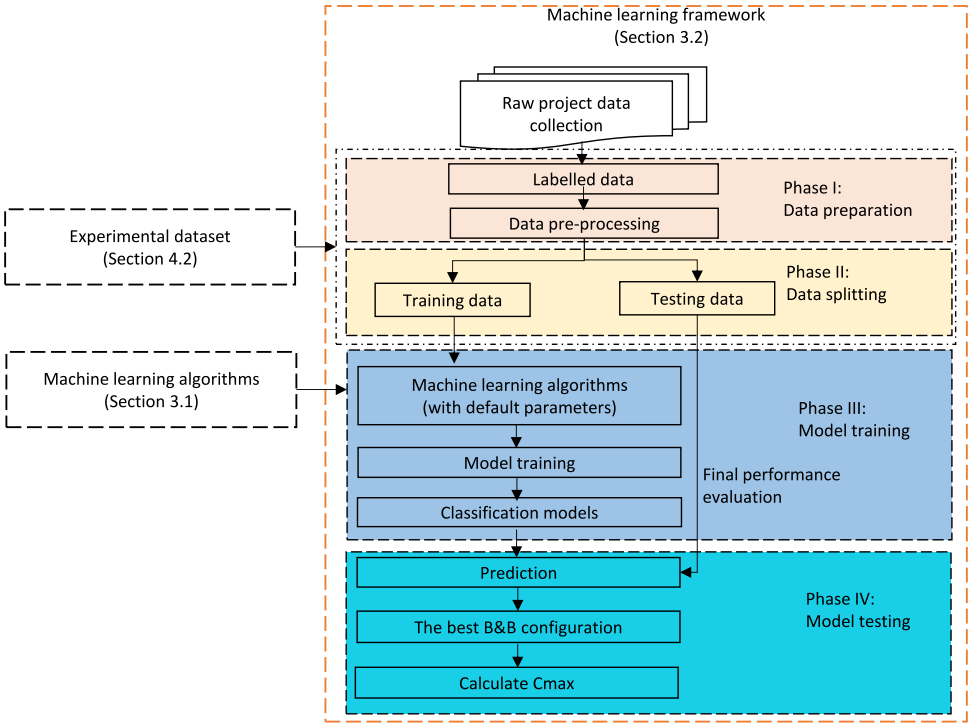

Fig. 2. Framework of the machine learning model.

of lower bound (LB) and upper bound (UB) (truncated after 1 min or 1 h). Second, for all solutions, the best-found LB and UB are calculated for each instance as the best value of all 24 configurations. These best-found solutions (LB or UB) will be used as the calibration value (further abbreviated as C*) and the results given by a specific configuration will be compared with this value. In case a specific configuration generates this best-found solution, it is labeled as "positive", and "negative" otherwise. Third, all instances with the same solutions (LB or UB) given by 24 configurations for the given time limit are removed from the experimental data.

Phase II: Data split. Splitting the data into training and testing sets ensures the model's performance is evaluated on unseen data, which mimics real-world scenarios. Based on all "hard" instances prepared in Phase I, this stage aims to split the data used in different phases of our model. Specifically, the dataset is divided into the training set (80%) and the testing set (20%) similar to Chang et al. (2011), where the training set is used to construct the classification model (Phase III) and the testing set is used to evaluate the final performance of the trained classification models (Phase IV).

Phase III: Model training. This step focuses on the core objective of machine learning—learning patterns, relationships, and dependencies from the training data. As can be seen from Fig. 2, this phase aims to build classification models based on the training data obtained in Phase II, which consists of two sub-phases: (1) Problem transformation. This sub-phase aims to transform the multi-label learning into multiple single-label learning by BR or CC (Sections 2.2 and 2.2.3). Both BR and CC involve Q binary transformations – one for each configuration (label). (2) Model building. This sub-phase aims to learn the mapping between features and labels to build classification models based on the data transformed by BR or CC through six machine-learning techniques (Section 3.1). As introduced in Section 2.2.3, any project instance in the multi-label training set will be involved in the learning process of Q binary classifiers, once the relation has been learned, the trained model can be used to generate the predictions for unseen instances and is expected to provide the best predictive performance. For a fair comparison, all six techniques are tested with their default parameter settings.

Phase IV: Model testing. This steps aims at testing the model on unseen data and provides an unbiased assessment of its performance and generalization capabilities. Once the mapping between project indicators and the performance of the B&B configurations is learned by the models, the classification models with default parameter values can be applied to all testing instances to estimate the final performance. For each test instance, first, the values of project indicators are calculated, and then these values are used as input to classification models to detect the best-performing configurations to solve the instance. For classification models built on CC, the relevant label of each instance is generated by querying the prediction given by each classifier and the final predictions are determined by combining the outputs of all Q classifiers. Therefore, the final result of the classification model is determined by each classifier constructed, and CC methods also have an impact on the final result (Section 2.2.3). If there are multiple positive predictions given by different binary classifiers (both for LB and UB), the solutions given by all positive predictions need to be generated, which means that multiple configurations (maximum 24) might be used to solve the instance in the test set. In a later experiment, the number of positive predictions will be restricted to avoid that too many configurations must be used, and the impact of such restriction is also analyzed.

4. Experiments and discussion

In this section, we evaluate the performance of the proposed model (Section 2.2.3). Section 4.1 provides detailed information about the experimental setup, the summary of Experiments, as well as different methods used in the experiment. Section 4.2 details various benchmark dataset used. Section 4.3 introduces different evaluation metrics used to measure the predictive performance. The results of the five experiments are sequentially presented in five subsections (from Section 4.4 to Section 4.8). In order to evaluate the performance of the proposed machine learning framework, we not only compare the models with some current mainstream and cutting-edge machine learning models but also select classic machine learning methods from the literature.

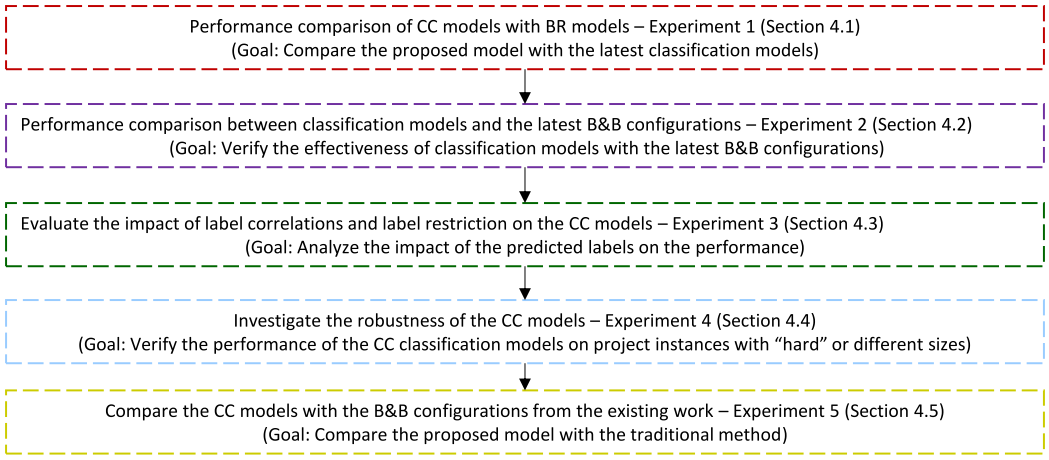

Fig. 3. Relationships of five different comparison experiments.

4.1. Experimental setup

Experimental environment. All computational experiments are conducted on a PC with Intel(R) Core(TM) i7-8550U 2G/1.8G CPU and 12.0 G RAM and implemented using the Mulan framework (Tsoumakas et al., 2009). Mulan is an open-source Java library for learning from multi-label data sets, built on top of the Weka tool (Witten & Frank, 2002).

Overview of various methods used in the experiments. To evaluate the performance of the newly proposed machine learning model, we compared it with various state-of-the-art methods from the literature, including the latest machine learning binary relevance classification method used to solve the RCPSP (Guo et al., 2021), the latest branch-and-bound procedures developed by Coelho and Vanhoucke (2018), various B&B procedures from previous work, and machine learning clustering algorithm. More specifically, five experiments with different focuses are conducted by using the proposed machine learning framework introduced in Section 3.2.

- In Experiment 1 (Section 4.4), we compare the performance of classification models built based on two problem transformation methods, BR and CC. Among them, BR is the latest machine learning classification model used to solve RCPSP in the existing literature.
- In Experiment 2 (Section 4.5), we compare the performance of the best-performing classification models with the latest branch-and-bound configurations from the literature.
- In Experiment 3 (Section 4.6), we analyze the impact of label correlation in the classifier chains-based model to explain how it affects the quality of the solution.
- In Experiment 4 (Section 4.7), we verify the robustness of the classification models on projects with different activities, hard instances, and even without known optimal solutions.
- In Experiment 5 (Section 4.8), we compare the performance of the proposed models with different types of algorithm from the literature existing work, including, various branch-and-bound procedures and clustering algorithm.

In order to have a better understanding of the goals and details of each experiment, we provide the relation of five experiments in Fig. 3.

4.2. Experimental dataset

To evaluate the performance of the proposed machine learning models, we use problem instances from two benchmark datasets, including the well-known project scheduling library PSPLIB (Sprecher & Kolisch, 1996) and CV datasets (Coelho & Vanhoucke, 2020). The PSPLIB set (Sprecher & Kolisch, 1996) contains 2040 instances with 30 (J30), 60 (J60), 90 (J90), and 120 (J120) activities, each with four resource types and different network topologies. The CV dataset is recently introduced by Coelho and Vanhoucke (2020) which is designed to only contain 623 instances that are very hard to solve to optimality. The selection of the dataset in the current work was motivated by the following: PSPLIB is the best known and most widely used set in the project scheduling literature and consists of projects from 30 to 120 activities. It is known in the literature that all 30-activity instances can be solved to optimality by most branch-and-bound procedures, but for the 60, 90, and 120 activity instances, many instances are yet unsolved. While the CV set consists of small instances with a maximum of 30 activities. Unlike the 30-activity instances in the PSPLIB set, most CV instances have no known optimal solutions and are therefore challenging for current branch-and-bound procedures. In order to verify the performance of our model with various state-of-the-art methods, we used these two data sets with different characteristics in the experiments. All datasets can be downloaded from www.projectmanagement.ugent.be/research/data.

In these two datasets, many of project instances used are really hard to be solved to optimality. For example, the CV dataset consists of hard instances that could not be solved to optimality even after 20 h of computation time. Therefore, we set two finite values to specify the truncation time for the branch-and-bound procedures, i.e. 1 min and 1 h (Coelho & Vanhoucke, 2018), and the results have been given in Table 5. In order to compare the performance of different methods, all experiments are performed on the same data (randomly partitioned, Section 3.2) and the same time limit (truncation time of the B&B procedure), therefore, the dataset and the truncation time used do not affect the final result. Table 5 summarizes the datasets used in the experiments, showing the number of whole instances (#Instances (Whole)) as well as the number of "hard" instances (#Hard instances) in each dataset. More specifically, the instances are divided into "easy" and "hard" in a simple and pragmatic way that uses the time limit as an input. From the moment the lower bound (LB) or upper bound (UB) values generated by the 24 configurations, truncated after 60 s (60 s) or 1 h (1 h), is the same, we called the project instance "easy", otherwise, we called the project the instance is said to be "easy", and otherwise, it is assumed to be "hard". The "easy" instances that have the same solution for all configurations are regarded as noisy data for our classification task, which is why they are removed from the experimental data. Only the "hard" instances in the table are kept in computational experiments for further analysis.

Table 5
Datasets with different time limits used in the experiment.

Dataset	Subset	#Instances (Whole)	#Hard instances[a]	
			Lower bound (LB) | Upper bound (UB)[b]	
			60 s[c]	1 h[d]
PSPLIB	J30	480	151|236	149|222
	J60	480	128|289	128|266
	J90	480	111|303	118|284
	J120	600	273|595	301|594
CV		623	573|623	608|623
Total		2,663		

[a] The number of "hard" instances for each dataset for a total given time limit.
[b] The number of "hard" instances generated by all configurations in terms of LB or UB.
[c] The total time limit is set to 60 s.
[d] The total time limit is set to 1 h.

4.3. Evaluation metrics

In the machine learning community, there are numerous widely accepted standards for evaluating the performance of machine learning models themselves. Similarly, in the project scheduling community, various metrics are designed to assess the quality of solutions and evaluate the performance of models used for scheduling. In the current work, our goal relies on machine learning to establish the relationship between project indicators and the performance of branch-and-bound procedures. To evaluate the effectiveness of the proposed models and achieve the highest quality of solution performance for the RCPSP, we use four widely used evaluation metrics in project scheduling, which are based on the lower bound and upper bound values of the test instances. The explanations of four evaluation metrics are given below:

- The *cumulative lower bound* (CLB) measures the quality of the lower bounds produced by different methods and calculate the sum of the makespan over all test instances.
- The *cumulative upper bound* (CUB) computes the quality of the upper bounds given by different methods and calculate the sum of the makespan over all test instances.
- The *cumulative absolute deviation* from the best-found LB (ADevLB) is used to measure the cumulative deviation between the best LB produced by a specific method and the best-found LB for all instances in the test set. First, it calculates the deviation for each instance and then takes the sum of all the deviations for all instances.
- The *cumulative absolute deviation* from the best-found UB (ADevUB) can be computed in a similar way to ADevLB, except that the LB needs to be replaced by UB. As said, the solution can be the prediction given by classification models or any other method used.

4.4. Experiment 1. Performance of two problem transformation methods

In this first experiment, we aim to compare the classifier chains method with the state-of-the-art machine learning binary relevance method used to solve the RCPSP (Guo et al., 2021), both of which are built based on six tradition machine learning algorithms (i.e. BPNN, DT, *k*NN, NB, RF, and SVM). Since BR does not require any ranking of labels, only one scenario is shown, but the results for CC are given for three different scenarios (Random (S1), Sum (S2), and Best (S3)). Table 6 displays CLB and CUB for the LB and UB calculations for six base classifiers. The solutions of various B&B configurations are obtained within a 60-second (60s) time limit.

The values in bold show the best results for each base classifier for the LB and UB calculations, which can be used to compare the performance of CC and BR for any given base classifier. The results indicate that the CC classification model performs at least as good, and often outperforms the BR classification model for all but one base classifier, which illustrates the potential of adding label correlation. More specifically, for two classifiers (BPNN and RF), there exists always at least one scenario that the CC model outperforms the BR model (both for calculating the LB and UB). Moreover, the last column also displays the average values of the six classifiers under BR and CC, the results show that incorporating solution quality into the ranking of the correlated labels helps, since the third scenario (S3) shows the average best performance, both the LB calculations (22,021) and UB calculations (40,156).

The table shows only one column in which BR performs better than CC. Indeed, the DT classifier performs slightly better for BR to calculate the lower bounds than for CC under any of the scenarios. Three other classifiers (*k*NN, NB, and SVM) show similar performance, regardless of whether CC or BR is used. From the last column of the table, we can know that the CC model outperforms the BR in terms of average performance (both in terms of LB and UB) over the whole dataset, which further indicates that label correlations affect the performance of the classification model.

The values shown in italic refer to the best performing results (highest values for LB and lowest values for UB) for all classifiers and both CC and BR. The table shows that the BPNN under Scenario 3 is the best performing model and outperforms all other algorithms. Since Scenario 3 is the best performing scenario for CC, indicating that the order of the configurations is best done according to the number of times the configuration performs best, CC-BPNN model will be used in our further experiments, and will occasionally be compared with the best performing BR models (BR-DT for LB calculations and BR-BPNN for UB calculations).

4.5. Experiment 2. Performance comparison between classification models and B&B procedure

In order to test the ability of the classification models in detecting the best-performing branch-and-bound configurations, we compared the overall performance of two classification models based on BR and CC with the latest 24 CBP from Coelho and Vanhoucke (2018) (Appendix C) as well as with the best-found solution C*. Recall that C* selects the best solution for all 24 configurations for each instance in the test set which therefore acts as the ideal scenario. Since the proposed classification models are designed to select the best-performing configuration(s), but will probably not succeed in always selecting the best-performing one, they will most likely never generate a solution similar to C*. Nevertheless, the closer a method lies to the ideal C* solution, the better it works.

Table 7 shows the results of different methods for the LB and UB calculations. All results are ranked in decreasing (for LB) and increasing (for UB) values of the sum of the makespans, therefore, displaying the best-performing methods on top. The two types of classification models are abbreviated by BR-DT (binary relevance using decision tree) or BR-BPNN (binary relevance using back-propagation neural network)

Table 6
Performance comparison of CC and BR method (in terms of CLB and CUB values)

		BPNN	DT	kNN	NB	RF	SVM	Average value
LB	CC (S1)	***22,025***	22,018	22,022	22,014	22,020	22,023	22,020
	CC (S2)	22,019	22,019	22,022	22,012	**22,023**	22,015	22,018
	CC (S3)	***22,025***	22,018	22,022	22,014	**22,023**	22,023	**22,021**
	BR	22,007	**22,023**	22,022	22,014	22,022	22,023	22,019
UB	CC (S1)	41,138	39,932	40,388	**40,018**	40,774	41,313	40,594
	CC (S2)	40,392	41,091	40,388	40,731	**39,671**	40,823	40,516
	CC (S3)	***39,628***	**39,825**	40,388	40,266	40,215	**40,614**	**40,156**
	BR	39,880	40,622	40,388	40,219	40,150	40,880	40,357

and CC-BPNN (classifier chains using back-propagation neural network) based on the results of the previous subsection. The individual configurations are abbreviated using the labels discussed in Section 2.2.2 (Table 3) as follows:

- Search strategy: L = minimum lower bound strategy (LBS) and U = upper bound strategy (UBS).
- Branching scheme: A = activity starting time (AST), P = parallel (PAR) and S = serial (SER).
- Branching order: A = activity ID (AID) and B = best lower bound (BLB)
- Composite lower bound CLB: 0 = critical path (CLB0), or 3 extensions from 4 = CLB4, 8 = CLB8 to 12 = CLB12.

The table shows that the two types of classification models outperform any single configuration, and their results sometimes lie impressively close (certainly for the LB calculations) to the ideal scenario C*. The results show that CC-BPNN lies closer to the C* than those produced by BR-DT, although only marginally for the LB, but significantly for the UB. This result illustrates that incorporating label correlations has a positive impact on the classification model. This latter observation will be further analyzed in the next subsection.

Moreover, the table also shows the importance of comparing and improving single configurations of existing branch-and-bound procedures. For example, the single best configuration is LPB12 for the LB calculations and the UPB12 for the UB calculations, but the classification models perform better. This is an interesting direction since it shows the power and ability of the proposed machine learning classification models to improve the branch-and-bound procedures by just selecting the suitable configurations for a given instance. Furthermore, the result also shows that improvements can be made on single configurations too (which then of course lead to improvements to the classification models).

4.6. Experiment 3. Impact of the hyper-parameter label correlation

The previous experiments showed that the two types of classification models outperform any single configuration, but this of course does not automatically state that these methods are good predictors to select the best-performing configuration. As a matter of fact, the classification models work as multiple binary classification models, which means that they aim at predicting for each configuration (label) whether it will lead to a high-quality solution (positive label value) or a low-quality solution (negative label value). It is common that the label imbalanced in the multi-label dataset, therefore, we will analyze the impact of positive label on the model performance. Each time a positive label value is predicted, the test instance is solved using that particular configuration. Consequently, while a single configuration only uses – by definition – one single configuration, the classification models use multiple configurations to find a solution for a test instance. In the worst case, all $|Q| = 24$ labels are positive, resulting in 24 times a search for a high-quality solution. The comparison between the single configurations and the classification models is therefore not fair.

In order to analyze this unfairness of label imbalance, Fig. 4 displays the number of positive labels (#PL) for each classification model showing minimum (minPL) and maximum values (maxPL), but also average values (AvgPL) and standard deviations (StdPL) for 248 and 409 test instances used in the experiment in terms of LB and UB, respectively. This figure shows that the average number of positive labels is comparable between BR models and CC models, with only slightly lower values for BR and CC. A lower number of positive labels indicates that the solutions can be found with less effort since only the positive label values require a call to the branch-and-bound procedure. Given the comparable values between the two types of models, CC-BPNN can be said to outperform BR-DT in terms of LB, since the effort is relatively the same, but the quality is somewhat better for CC-BPNN compared to BR-DT as explained earlier (cf. Table 7). And similar findings can also be found in the calculations of UB.

Fig. 5 shows the evolution of solution quality (y-axis) of the two classification models along the number of positive labels (x-axis) ranging from 1 to MaxPL. The graph on the left shows the results for LB calculations, while the right graph displays the UB results. The results show that CC-BPNN always outperforms BR-DT in terms of LB, the observations demonstrate that taking into account label correlation can improve the solution quality in terms of LB, which illustrates the importance of label correlation. For the UB calculations, CC-BPNN outperforms BR-BPNN when only one label is used, but the two models go into a competition mode when the number of labels ranges between 2 and 4. From 5 and more labels, CC-BPNN clearly outperforms the uncorrelated BR-BPNN model. These findings further illustrate that taking into account label correlation can also improve the solution quality in terms of UB.

Given the results shown in Fig. 5, we have redone our tests using truncated results of 1 h (instead of 60 s used in the previous experiments) in total for calculating upper bounds (UB), regardless of how many positive labels are predicted. More particularly, each instance is solved by all configurations with a positive label (PL) value such that the total limit is equal to 1 h. When #PL is used to denote the number of positive labels in the prediction, then the time spent at each configuration equals therefore $\frac{1hour(1h)}{\#PL}$. Of course, since each instance can have a different number of positive label predictions (minimum and maximum values are shown in Fig. 4), the total time for each positive label (configuration) differs for each project instance. For example, if a certain instance has 4 positive predictions, the instance is solved by each of these positive configurations for 15 min each. Each solution of one configuration is used as input for the next configuration run, and the configurations are sequentially ordered randomly. Hence, more positive predicted labels, therefore, result in a lower allowable time for each configuration, since the total time must always be equal to one hour. The results of this sequential approach as shown in the second column of Table 8. These results should be compared with the values obtained by the so-called separate runs, also shown in this table. In these runs, all positive configurations are used to obtain solutions in a separate way, and each configuration gets a total time of 1 h. Consequently, the total time for these runs equals #PL *times* 1 h, which exceeds the total time spent on the sequential runs. For these results, the column "Min" shows the best upper bound found for each positive configuration (each time truncated after 1 h), the "Max" column displays the worst solution found, while the "Avg" column displays the

Table 7
Performance comparison of classification models with the latest CBP.

Rank	Methods	CLB	ADevLB	Methods	CUB	ADevUB
1	C*	22,029	0	C*	37,827	0
2	CC-BPNN	22,025	4	CC-BPNN	39,628	1,801
3	BR-DT	22,023	6	BR-BPNN	39,880	2,053
4	LPB12	21,992	37	UPB12	41,676	3,849
5	LPA12	21,982	47	USA4	41,891	4,064
6	LPB8	21,915	114	USA0	41,900	4,073
7	LPA8	21,909	120	USA8	41,902	4,075
8	LPB4	21,901	128	UPB4	42,144	4,317
9	LPA4	21,890	139	UPB8	42,169	4,342
10	LPB0	21,889	140	UPB0	42,170	4,343
11	LPA0	21,879	150	UAA12	42,197	4,370
12	LSB12	21,424	605	UAA8	42,293	4,466
13	LSA12	21,422	607	UAA4	42,294	4,467
14	LSB8	21,325	704	UAA0	42,298	4,471
15	LSA8	21,325	704	UAB0	42,419	4,592
16	LSB4	21,320	709	UAB12	42,426	4,599
17	LSA4	21,319	710	UAB4	42,428	4,601
18	LSB0	21,291	738	UAB8	42,439	4,612
19	LSA0	21,291	738	USB4	42,710	4,883
20	LAB12	21,242	787	USB8	42,768	4,941
21	LAA12	21,233	796	USB0	42,834	5,007
22	LAB8	21,215	814	USB12	43,598	5,771
23	LAB4	21,212	817	USA12	45,397	7,570
24	LAB0	21,209	820	UPA4	51,332	13,505
25	LAA8	21,208	821	UPA12	51,354	13,527
26	LAA4	21,198	831	UPA0	51,372	13,545
27	LAA0	21,188	841	UPA8	51,389	13,562

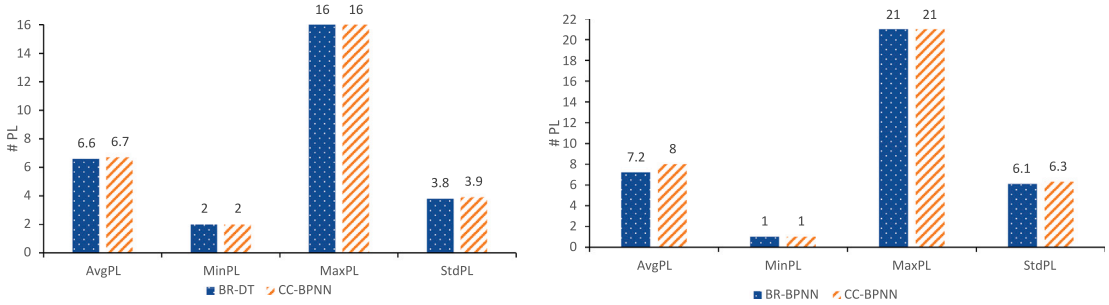

Fig. 4. Number of positive labels for the two classification models.

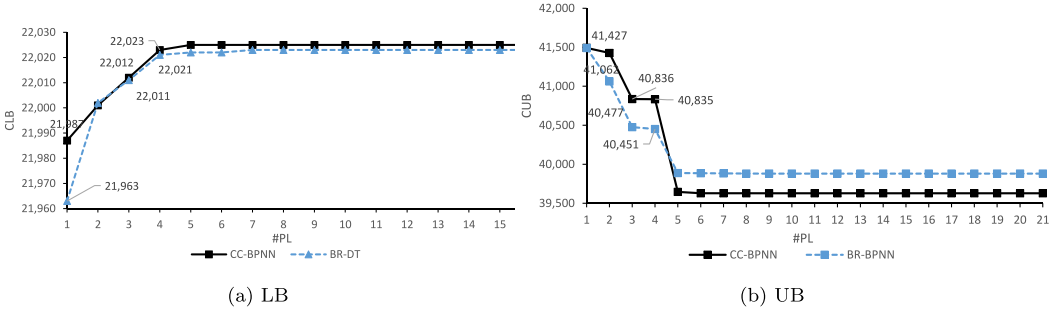

Fig. 5. Impact of restricting the number of positive labels (#PL) used.

average value of the upper bound found by all positive configurations after 1 h each. Furthermore, since it is important for the RCPSP to find high-quality solutions in a very fast way, we have compared the quality of the solutions (UB) generated by the classification models and the single-best configuration UPB12 under a stop criterion of 1 s (instead of the 60 s or 1 h used in the previous settings). In this run, all positive configurations are used to obtain the solutions in a separate way, and each configuration gets a total time of 1 s. The results given by this separate run are given in the last column of Table 8.

The results show that the sequential runs perform better than the average performance of the separate runs, and come close to the minimum value of the separated runs. Of course, the minimum value is obtained by taking each time the lowest makespan of all 1-hour runs (for each positive label), while the sequential run only runs for $\frac{1h}{\#PL}$ and is therefore much quicker. These results clearly show that our label prediction methods outperform the use of separate configuration runs, and confirm that CC-BPNN model slightly outperforms BR-BPNN model. In addition, the results also indicate that both classification

Table 8
Solutions truncated by the number of positive predicted labels.

Methods	Sequential runs	Separated runs			
	(Total time = 1 h)	(1h for each PL)			(1 s for each PL)
	($\frac{1h}{\#PL}$ for each PL)	Min	Max	Avg	
CC-BPNN	39,505	39,426	42,324	40,615	42,440
BR-BPNN	39,728	39,672	42,147	40,792	44,193
UPB12	–	41,676	–	–	54,686

Table 9
Performance of various methods on subsets.

Dataset	Methods	CLB	ADevLB	Methods	CUB	ADevUB
J30	C*	2,082	0	C*	2,736	0
	CC-BPNN	2,082	0	CC-BPNN	2,736	0
	LPB0	2,082	0	USB12	2,736	0
J60	C*	2,572	0	C*	4,704	0
	CC-BPNN	2,572	0	CC-BPNN	4,706	2
	LPA12	2,572	0	UPB12	4,707	3
J90	C*	2,632	0	C*	5,663	0
	CC-BPNN	2,632	0	CC-BPNN	5,688	25
	LPB12	2,632	0	USA12	5,823	160
J120	C*	7,735	0	C*	15,284	0
	CC-BPNN	7,735	0	CC-BPNN	16,517	1,233
	LPB12	7,734	1	USA4	16,608	1,324
CV	C*	7,082	0	C*	7,937	0
	CC-BPNN	7,081	1	CC-BPNN	8,142	205
	LPB12	7,078	4	USB12	8,363	426

models can generate higher quality solutions (42,440 and 44,193) in relatively short computational times (a truncated time limit of 1 s) than the single-best configuration UPB12 (54,686).

4.7. Experiment 4. Effectiveness of the model on five different project sizes

The previous experiments demonstrate that the proposed classifier chains method outperforms the binary relevance method and that CC-BPNN performs best in Scenario 3. In this experiment, we further retrain and retest the proposed CC-BPNN model in Scenario 3 to evaluate the robustness of the model on each subset with a different number of activities in the time limit of 1 h. Specifically, the number of activities in the subset varies between a minimum of 20 (in the CV dataset) and a maximum of 120 (in the J120 dataset), and the experiment is conducted on all "hard" project instances in each subset (Section 4.2).

Table 9 presents the results given by various methods on each subset, including the proposed classifier chains method, the single-best configuration, and C*. It should be noted that C* and the single best configuration in this table is obtained by comparing the results given by 24 configurations on each subset. From this table, it can be observed that the proposed classification model preserves robustness for different number of activities. The results indicate that the classifier chains models are superior to the single best B&B configuration in both LB and UB calculations, although the best-performing configuration may vary in different subsets. For the LB calculations, the deviation between the solutions produced by the single-best B&B configuration and that produced by C* is very small, however, the classification still performs best. For subsets J30, J60, and J90, the results given by these two methods are the same on the test instances. For the other two subsets J120 and CV, the deviations given by the single-best B&B configuration LPB12 and C* are 1 and 4, respectively. For the UB calculations, the differences between the classification method and the single-best B&B configuration are more outspoken. These results show that the proposed classification model generates results closer to C* than the single-best B&B configuration. As a result, we can conclude that the classifier chains method is still robust in practical applications when the number of activities in the subsets varies.

4.8. Experiment 5. performance comparison between classification and clustering

In this experiment, our goal is to compare the performance of classification model and traditional clustering algorithm (i.e., K-means) on the whole dataset to evaluate the generalization capacity of the proposed CC model. Different from the previous 4 experiments, in which 24 configurations are used, in this experiment, only 9 configurations obtained from the literature that could be matched to our 24 configurations are used. The K-means algorithm is the most commonly used clustering algorithm because it is simple and efficient in terms of execution time (Yu et al., 2018). The goal of K-means is to group data into K clusters based on the distance measure between each data point and the cluster centers, where K indicates the number of clusters used. Euclidean distance is widely used in clustering and is also the default distance measure used with the K-means (Huang et al., 2008; Yiakopoulos et al., 2011), which is why we use it in our experiment. More specifically, we retrain and retest our model in this new group consisting of 9 configurations, and the experiment is carried out on all "hard" instances in the dataset. For more details on these matches, the reader is referred to the work of Guo et al. (2023).

Table 10 shows the performance comparison of four different methods on the whole dataset with a time limit of 1 h, including CC-BPNN, K-means, 9 configurations from the existing literature, and C*(9). Since the hyper-parameters of K-means (i.e., number of clusters K) has an impact on the clustering results, to gain further insight into the impact of this hyper-parameters on the performance, we change the cluster numbers from 2 (default value) to 9 (the number of configurations used), i.e. $K \in \{2, 3, 4, 5, 6, 7, 8, 9\}$. In Table 10, we only keep the best results of K-means, and the results of K-means with different K are shown in Fig. 6. In addition, in order to have a better understanding of the performance of 9 configurations, we list not only the results of these 9 configurations but also the references for each configuration. It should be noted that C*(9) is obtained by comparing the results given by nine configurations on the test instances in terms of LB and UB, respectively. The single-best configuration is also obtained by comparing the results of these nine configurations. The results indicate that CC-BPNN models are competitive with K-means and 9

Table 10
Comparing the classification models with methods from the literature.

Literature	Abbreviation	CLB	ADevLB	Literature	Abbreviation	CUB	ADevUB
	C*(9)	26,083	0		C*(9)	37,984	0
The current work	CC-BPNN	26,083	0	The current work	CC-BPNN	38,850	866
Clustering algorithm	*K*-means	26,083	0	Clustering algorithm	*K*-means	38,860	876
Stinson et al. (1978)	LPB4	26,083	0	A. (2000)	USA4	38,915	931
Bell and Park (1990), Stinson et al. (1978)	LPB0	26,073	10	A. (2000)	USA0	38,943	959
Nazareth et al. (1999)	LPA0	26,068	15	Patterson and Huber (1974), Talbot and Patterson (1978)	UAA0	40,728	2,744
Patterson and Huber (1974)	LAA0	25,406	677	Christofides et al. (1987), Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998)	UPA4	49,062	11,078
Christofides et al. (1987), Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998)	UPA4	24,852	1,231	Christofides et al. (1987), Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998)	UPA0	49,093	11,109
Christofides et al. (1987), Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998)	UPA0	24,829	1,254	Stinson et al. (1978)	LPB4	106,118	68,134
A. (2000)	USA4	24,764	1,319	Bell and Park (1990), Stinson et al. (1978)	LPB0	107,507	69,523
A. (2000)	USA0	24,753	1,330	Nazareth et al. (1999)	LPA0	113,441	75,457
Patterson and Huber (1974), Talbot and Patterson (1978)	UAA0	24,436	1,647	Patterson and Huber (1974)	LAA0	141,230	103,246

configurations. For LB calculations, the result given by the single-best configuration LPB4 and *K*-means is equal to that produced by C*(9), in which case it is clear that CC-BPNN can also generate the same result. For UB calculations, the deviations between the results produced by the single-best configuration USA4, *K*-means and C*(9) are 931 and 876 respectively, however, the deviation between the results of CC-BPNN and C*(9) is 866. The result further confirms the findings in previous experiments that the proposed classification model outperforms any single best B&B configuration or machine learning method and that it is closer to C*(9) than the single best one.

Based on the results of the K-means, we found that the LBs produced by *K*-means with different *K* are equal, and we can conclude that *K* does not affect the quality of LBs. Therefore, we only analyze the effect of hyper-parameters *K* of *K*-means on UB, and the results are shown in Fig. 6. For the sake of comparison, we also present the results of CC-BPNN and the single-best configuration USA4. From Fig. 6, we can see that the worst (39,050) and best (38,860) clustering results are attained when the number of clusters is 6 and 7, respectively. The worst result is even worse than the one produced by USA4, while the best result is very competitive as it is close to the result given by CC-BPNN. Furthermore, the results produced by *K*-means and USA4 are equal when *K* is equal to 2 or 3, which indicates that *K*-means cannot improve the quality of UB when the number of clusters is small. The results show that the proposed classification model not only outperforms nine configurations from the project scheduling literature, but also outperforms *K*-means clustering algorithm from different categories in machine learning.

5. Conclusions and future research

In this paper, we have proposed two problem transformed-based machine learning classification models (binary relevance and classifier chains) to detect the best-performing B&B configurations for solving the RCPSP, aiming to find the mapping between various project indicators and the quality of the solution (i.e. makespan) given by different B&B configurations. Specifically, both binary relevance and classifier chains are used to transform the learning task into multiple binary learning tasks. The main difference between BR and CC methods is that the latter considers the possible correlation between labels.

We have compared the model with various state-of-the-art methods from the literature to evaluate its effectiveness in 5 experiments. The computational results show that CC works best with the Back-Propagation Neural Network algorithm, as well as outperforms the latest BR method and other classification models in the field of project scheduling. In a second experiment, we not only listed the performance of different methods but also compared the proposed models with various latest B&B configurations from the literature. The results show that CC-BPNN not only generates better overall performance than any single-best B&B configuration but also shows performance close to that of the best-chosen B&B configuration. The third experiment investigated the contribution of label correlation by restricting the maximum number of positive labels used in the models. Moreover, we also have added a sub-experiment where the model is tested under a high stop criterion to show that the models can help find high-quality solutions for the RCPSP. The fourth experiment retrains and retests the models to verify the robustness of the models on different subsets. In the last experiment, we used the model for the existing B&B procedures in the field of project scheduling, in addition, we also ran the *K*-means clustering algorithm to show how the models can be used to improve the current state-of-the-art knowledge.

Based on the experimental results obtained and comparison classification performance presented, we are of the opinion that the current work has mainly academic value in the world of project scheduling, as it showed that the quality of solutions can be improved in different ways. Rather than changing and expanding the existing solution

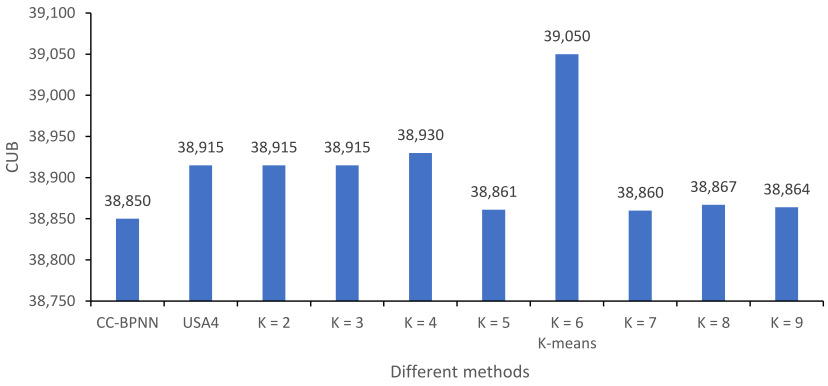

Fig. 6. Impact of hyper-parameters of K-means with different clusters.

methods (which is normally the topic of many research work), the existing algorithms can be better (and automatically) combined using known machine learning methods. However, the work showed that this combination must be done well, with the correct use of a carefully selected diverse dataset, a good insight into the existing project indicators, and a good implementation of the classification models being central. First and foremost, the work has shown that the network and resource indicators used as features adequately describe the characteristics of the project. Although we have used existing indicators in the research that have already proven their usefulness in other studies, this research is not standing still either. For example, Van Eynde and Vanhoucke (2022) recently proposed new indicators, and it would be useful to investigate in a follow-up work whether these indicators can also be used in the machine learning work such as ours. In addition, the work also showed that a classification task works better when a classifier chain method is used instead of the classic binary relevance. These insights seem like small adjustments from a machine learning framework but nevertheless provide insights on how best to use the machine learning methods for project scheduling.

In light of the increasing interest in the application of machine learning methods to investigate various project scheduling problems, it is important for researchers and practitioners to have a greater awareness of the potential of machine learning. For example, how to expand the proposed model to solve more problems or understand its implementation. As future research, the current work can be expanded in several directions: First, the B&B procedures used can be replaced by any possible solution methods, such as priority rules or meta-heuristics. Second, the proposed models are not limited to solving the resource-constrained project scheduling problem but can be extended to different variants of this challenging problem, provided the necessary adjustments are made. Currently, a number of attempts have already been made for this (e.g. Guo et al., 2023, 2021 and Messelis & De Causmaecker, 2014), but the research on this theme still needs more efforts.

The practical significance of our work lies in its ability to enhance decision-making in complex project scheduling scenarios. By leveraging machine learning techniques, our models offer a systematic way to select the best B&B configurations based on project-specific indicators, significantly reducing trial-and-error efforts in B&B configuration selection. These models can be utilized in various industries, such as construction, manufacturing, aerospace, and logistics, where efficient resource-constrained scheduling is critical for operational success.

Moreover, the ability of the proposed models to handle label correlations and adapt to different project characteristics ensures their applicability across diverse scheduling environments. For instance, project managers can use these models to predict optimal scheduling configurations, improving project completion times and resource utilization. In addition, the robustness and scalability of our approach suggest that it can be extended to other combinatorial optimization problems beyond RCPSP, broadening its potential applications.

CRediT authorship contribution statement

Weikang Guo: Conceptualization, Data curation, Formal analysis, Writing – original draft. **Mario Vanhoucke:** Formal analysis, Funding acquisition, Project administration, Resources, Supervision, Validation, Writing – review & editing. **José Coelho:** Data curation, Formal analysis, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Weikang Guo reports financial support was provided by China Scholarship Council.

Acknowledgments

The authors would like to thank the China Scholarship Council, China for providing financial support (Grant No. 201706050166).

Appendix A. Illustrative example for BR, CC, and MC

To have a better understanding of the difference between BR, CC, and MC, as well as highlight the impact of label order used in the CC, we provide the detailed process to explain how the decision tree models built based on the small given example training dataset D. From the example given in Table A.1, we can see that there are 16 project instances ($|D| = 16$) with 3 features ($|F| = 3$) and 3 labels ($|Q| = 3$). This dataset can be used to build the classifiers for multiple labels (three columns titled "Multiple labels (ML)") or multi-class (last column titled "Multi-class (MC)") learning tasks and a more detailed explanation is given below.

Two problem transformation methods BR and CC based on multiple labels:

- Binary Relevance (BR): From Section 2.3, we know that the basic idea of this method aims to transform the multi-label learning into multiple *independent* binary classification tasks, each binary classifier is constructed for each possible label. In BR method, the number of features and labels used in the training phase are both 3. The classifiers are then used to find the map between 3 features and each of the three labels (i.e. for L1, L2, and L3 independently), for which there are only two possible values "+ " or "−". Fig. 1(b) shows three different decision trees built on BR respectively, three classifiers corresponding to three different labels in the multiple labels columns of Table A.1. Since this BR method ignores the correlation between labels, it is impossible for one label to appear in the classifier built for the other.

Table A.1
Example training dataset D.

ID	Feature 1 (F1)	Feature 2 (F2)	Feature 3 (F3)	Classification			
				Multiple labels (ML)			Multi-class (MC)
				Label 1 (L1)	Label 2 (L2)	Label 3 (L3)	Newly defined class
1	Q	B	Medium	–	+	+	C2(L2|L3)
2	Q	B	High	+	–	+	C2(L1|L3)
3	Q	B	Medium	–	+	+	C2(L2|L3)
4	T	A	High	+	–	–	C1(L1)
5	Q	A	High	+	–	–	C1(L1)
6	F	B	Medium	+	–	+	C2(L1|L3)
7	Q	B	High	–	+	+	C2(L2|L3)
8	Q	A	Medium	+	–	–	C1(L1)
9	F	A	Medium	–	–	+	C1(L3)
10	Q	B	High	+	–	+	C2(L1|L3)
11	T	B	Low	+	+	–	C2(L1|L2)
12	Q	A	Low	–	–	+	C1(L3)
13	T	C	Low	–	+	–	C1(L2)
14	F	B	Low	–	+	–	C1(L2)
15	T	A	High	+	–	+	C2(L1|L3)
16	Q	C	Low	–	+	–	C1(L2)

"+" indicates that the makespan given by one label is equal to the best-found solution produced by comparing all labels.
"–" indicates the makespan given by one label is not equal to the best-found solution produced by comparing all labels.

Table B.1
A small set D' of the example data set D.

Project ID	Features (F)			Labels (L)(UB)			Labeled labels		
	F1	F2	F3	L1	L2	L3	L1	L2	L3
1	Q	B	Medium	10	8	8	–	+	+
2	Q	B	High	15	19	15	+	–	+
3	Q	B	Medium	13	12	12	–	+	+

- Classifier Chains (CC): As explained in Section 2.2.3, the basic idea of this method is to transform the multi-label classification into a *chain* of binary classification tasks, and subsequent binary classifiers in the chain are built based on the predictions given by the previous ones, which overcomes the disadvantages of the BR while keep its advantages. More specifically, during the training phase of the classification, the CC method augments the feature space with extra features from the relevance of labels for training instances.
  From this example, it can be seen that the number of labels is 3 (that is the same as the number of labels in the BR method), while the feature space can be extended with the "+ "/"–" label relevances of all previous classifiers (which is different from the BR method). This means that the label used in the previous classifier built may become non-leaf nodes in its subsequent classifiers built based on this feature space. Therefore, the order of labels determined by the classifier chain method may lead to changes in the final classifier developed (Section 2.2.3).
  As explained in Section 2.2.3, we know the order of three labels in the chain based on Scenario 1 is L1, L2, and L3. Fig. 1 shows three different decision trees constructed by CC based on Scenario 1. It can be observed that L1 appears in the non-leaf nodes of classifiers 2 and 3, in this case, two classifiers built for L2 and L3 are related to L1.

Single-label method (MC):

- *Multi-class classification:* The basic idea of this method is to integrate all labels with positive value into a newly defined class for each training instance. Therefore, each instance is only associated with these newly defined classes, and the classifier is built to find the relation between features and all possible new classes. The definition of the new class can be divided into two cases, one is that the total number of relevant labels equals 1 (Case 1), and the other is that the total number of relevant labels is greater than 1 (Case 2). In Case 1, the new class for each instance is defined as $C_{\#1}(L_{\#2})$, where the capital C indicates the Class, "#1" denotes the number of relevant labels, and "#2" represents the ID of the only relevant label. Obviously, in this case, "#1" = 1. In Case 2, the new class of each instance is defined as $C_{\#L}(L_{\#2}|L_{\#3}|...)$, where the capital C also denotes the Class, "#L" indicates the total number of relevant labels and it is always greater than 1, and the "$L_{\#2}|L_{\#3}|..$" between the brackets represent the set of IDs of all the relevant labels. As can be seen from Fig. 1(a), there is only one decision tree classifier constructed by this method, where non-leaf nodes represent different features and leaf nodes indicate newly defined classes. For a more detailed description of the multi-class classification, the reader is referred to Guo et al. (2021).

Appendix B. Comparative analysis of BR and CC

In order to have a better understanding of the differences between BR and CC, we keep the first three project instances in Table A.1 (Appendix A) as a small set D' (Table B.1) to demonstrate their processes. Table B.1 shows the values of the three features ($|F| = 3$) and three labels ($|Q| = 3$) for the three instances ($|D'| = 3$). In addition, we also list the makespan given by each label to highlight the differences between the three scenarios used in CC. In Table B.1, the makespan of the label represents the quality of the upper bound (UB) found by the specific configuration for the project, ranging from 8 time units to 19 time units. Moreover, for each label q (L1 or L2, or L3) in the last three columns, only two possible labeled values are assigned, indicating whether a certain configuration yields the best possible makespan (labeled as "+") among 3 configurations or not (labeled as "–"). In this example, both BR and CC are used to deal with multi-label learning, and once the learning task has been transformed, machine learning algorithms can be used to learn the relation between features and labels to build classifiers. The transformation process of both BR and CC is explained as follows.

Table B.2 shows the transformed data generated by BR and CC on the three instances of Table B.1, as well as the order of the labels used in CC for three scenarios. Since BR assumes that the labels are independent and would not pass any information to one another, it

Table B.2
The transformed data generated by BR and CC on D'.

	(a) BR method				(b) CC method																	
					(b1) Scenario 1						(b2) Scenario 2						(b3) Scenario 3					
Order of labels	-				L1 ≻ L2 ≻ L3						L3 ≻ L1 ≻ L2						L3 ≻ L2 ≻ L1					
	F1	F2	F3	L L1	F1	F2	F3	L L1			F1	F2	F3	L L3			F1	F2	F3	L L3		
Classifier 1 (h_1)	Q	B	Medium	−	Q	B	Medium	−			Q	B	Medium	+			Q	B	Medium	+		
	Q	B	High	+	Q	B	High	+			Q	B	High	+			Q	B	High	+		
	Q	B	Medium	−	Q	B	Medium	−			Q	B	Medium	+			Q	B	Medium	+		
	F1	F2	F3	L L2	F1	F2	F3	L1	L L2		F1	F2	F3	L3	L L1		F1	F2	F3	L3	L L2	
Classifier 2 (h_2)	Q	B	Medium	+	Q	B	Medium	−	+		Q	B	Medium	+	−		Q	B	Medium	+	+	
	Q	B	High	−	Q	B	High	+	−		Q	B	High	+	+		Q	B	High	+	−	
	Q	B	Medium	+	Q	B	Medium	−	+		Q	B	Medium	+	−		Q	B	Medium	+	+	
	F1	F2	F3	L L3	F1	F2	F3	L1	L2	L L3	F1	F2	F3	L3	L1	L L2	F1	F2	F3	L3	L2	L L1
Classifier 3 (h_3)	Q	B	Medium	+	Q	B	Medium	−	+	+	Q	B	Medium	+	−	+	Q	B	Medium	+	+	−
	Q	B	High	+	Q	B	High	+	−	+	Q	B	High	+	+	−	Q	B	High	+	−	+
	Q	B	Medium	+	Q	B	Medium	−	+	+	Q	B	Medium	+	−	+	Q	B	Medium	+	+	−

Table C.1
The 24 configurations corresponding to all combinations of components.

LB/UB	Abbreviation	Search strategy	Branching scheme	Branching order	Composite lower bound
		LBS (L)/UBS (U)	AST (A)/PAR (P)/SER (S)	AID (A)/BLB (B)	CLB0 (0)/CLB4 (4)/CLB8 (8)/CLB12 (12)
UB	UAB0	UBS	AST	BLB	0
	USB0	UBS	SER	BLB	0
	UPB0	UBS	PAR	BLB	0
	UAB4	UBS	AST	BLB	4
	USB4	UBS	SER	BLB	4
	UPB4	UBS	PAR	BLB	4
	UAB8	UBS	AST	BLB	8
	USB8	UBS	SER	BLB	8
	UPB8	UBS	PAR	BLB	8
	UAB12	UBS	AST	BLB	12
	USB12	UBS	SER	BLB	12
	UPB12	UBS	PAR	BLB	12
	UAA0	UBS	AST	AID	0
	USA0	UBS	SER	AID	0
	UPA0	UBS	PAR	AID	0
	UAA4	UBS	AST	AID	4
	USA4	UBS	SER	AID	4
	UPA4	UBS	PAR	AID	4
	UAA8	UBS	AST	AID	8
	USA8	UBS	SER	AID	8
	UPA8	UBS	PAR	AID	8
	UAA12	UBS	AST	AID	12
	USA12	UBS	SER	AID	12
	UPA12	UBS	PAR	AID	12
LB	LAB0	LBS	AST	BLB	0
	LSB0	LBS	SER	BLB	0
	LPB0	LBS	PAR	BLB	0
	LAB4	LBS	AST	BLB	4
	LSB4	LBS	SER	BLB	4
	LPB4	LBS	PAR	BLB	4
	LAB8	LBS	AST	BLB	8
	LSB8	LBS	SER	BLB	8
	LPB8	LBS	PAR	BLB	8
	LAB12	LBS	AST	BLB	12
	LSB12	LBS	SER	BLB	12
	LPB12	LBS	PAR	BLB	12
	LAA0	LBS	AST	AID	0
	LSA0	LBS	SER	AID	0
	LPA0	LBS	PAR	AID	0
	LAA4	LBS	AST	AID	4
	LSA4	LBS	SER	AID	4
	LPA4	LBS	PAR	AID	4
	LAA8	LBS	AST	AID	8
	LSA8	LBS	SER	AID	8
	LPA8	LBS	PAR	AID	8
	LAA12	LBS	AST	AID	12
	LSA12	LBS	SER	AID	12
	LPA12	LBS	PAR	AID	12

learns one independent binary classifier for each label. Each classifier ($h(\cdot)$) maps the known feature vector $\boldsymbol{x}_i$ in the set D' into a single label y_{iq} with binary values "+ ", "−". A given instance is associated with one label if the value is "+ " and not associated if the value is "−". In this small set D', the number of labels is 3, therefore, the total number of classifiers constructed based on BR is also 3. In addition, the three classifiers are independent of each other. From Table 4 (the left column), it is easy to observe that the three classifiers (h_1, h_2, and h_3) are constructed for three different labels and they are also independent. For example, Classifier 2 (h_2) is built to find the relation between the three features and Label 2 (L2).

However, CC transforms the learning task into a *chain* of binary classification tasks to model possible dependencies among labels (i.e. label correlations). Each classifier in the *chain* deals with a binary relevance classification task associated with a unique label, and its feature space is augmented with all the predictions of the previous classifiers in the *chain*. The difference with respect to BR is that the feature space used to induce each classifier is extended by previous labels in the *chain*, and these labels are treated as additional features. The classifier is constructed to learn the relation between the extended feature space and each label.

From Table B.2 (the left column), we can see the labels in the data transformed by BR are independent of each other, and therefore, the classifiers built for the labels in the transformed data are also independent of each other. From Table B.2 (the right column), we can see that the labels in the data transformed by CC in three different scenarios (Scenario 1, Scenario 2, and Scenario 3) are related to each other. Obviously, the order of the labels in the *chain* for three scenarios is: L1 ≻ L2 ≻ L3 (S1), L3 ≻ L1 ≻ L2 (S2), and L3 ≻ L2 ≻ L1 (S3). For example, in Scenario 2, the sum of the makespan of the three labels is 38 (L1), 39 (L2), and 35 (L3), respectively, resulting in their order in the *chain* being L3 ≻ L1 ≻ L2. In Scenario 3, Label 3 (L3) provides 3 times the best solutions, compared to 2 times for L2 and only 1 time for L1, resulting in the order of the three labels being L3 ≻ L2 ≻ L1. By comparing the data transformed by BR and CC, it can be noted that the feature space of Classifiers 2 and 3 constructed based on CC is extended by one and two labels in the previous classifiers, respectively. For example, in Scenario 1, the feature space of Classifier 2 (h_2) is extended by L1 in Classifier 1 (h_1). Finally, for each scenario, a *chain* of binary classifiers ($h(\cdot) = \{h_1, h_2, h_3\}$) is formed, and each classifier in the *chain* is responsible for learning and predicting the binary association of the qth label given the feature space.

To have a better understanding the working mechanism of the CC-based models, we explain it using the classifiers constructed based on S1 (b1 in Table B.2): From this table, it is obvious that the CC model contains three classifiers (h_1, h_2, and h_3) and the order of labels in the chain is L1 ≻ L2 ≻ L3. Therefore, the final result of the model is determined by the set of predicted values of all three classifiers. For any new instance, we have to input the corresponding values into each classifier to generate the prediction. For example, we need to input the project characteristics of a specific instance to Classifier 1 h_1 to obtain the prediction. Then, we need to input the project features and the results given by h_1 to get the prediction of Classifier 2 h_2 since the information of L1 is included in h_2. Finally, we need to input the project features and the results given by h_1 and h_2 to get the prediction of Classifier 3 h_2 since the information of L1 and L2 is included in the h_3. This example further shows how label information is passed between different classifiers (from h_1 to h_3) and how the order of labels (Label correlations, L1 ≻ L2 ≻ L3) in the chain affects the performance.

The differences between BR and CC can also be observed from the decision tree classifiers constructed in Fig. 1. From Fig. 1(b) and Fig. 1(c), we can see that the Classifier 2 constructed based on CC contains a non-leaf node L1, which can pass the information of L1 to the classifier built for L2. And similar findings can also be observed in Classifier 3. This further illustrates that the three different scenarios in Table B.2 for determining the order of labels in the *chain* affect the final performance of the model. In the next section, it will be shown that not only the decision tree but also five other classical machine learning algorithms will be used as base classifiers to construct classification models.

Appendix C. All 24 configurations

See Table C.1.

Data availability

The data is available from a known website.

References

A., Sprecher (2000). Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, *46*(5), 710–723.

A.A., Mastor (1970). An experimental investigation and comparative evaluation of production line balancing techniques. *Management Science*, *16*(11), 728–746.

Adamu, P. I., Agarana, M. C., & Okagbue, H. I. (2018). Machine learning heuristic for solving multi-mode resource-constrained project scheduling problems. *vol. 1*, In *Proceedings of the international multiConference of engineers and computer scientists* (pp. 4–16).

Ali, W., & Malebary, S. (2020). Particle swarm optimization-based feature weighting for improving intelligent phishing website detection. *IEEE Access*, *8*, 116766–116780.

Bahroun, Z., Tanash, M., As'ad, R., & Alnajar, M. (2023). Artificial intelligence applications in project scheduling: A systematic review, bibliometric analysis, and prospects for future research. *Management Systems in Production Engineering*, *31*(2), 144–161.

Bell, C. E., & Park, K. (1990). Solving resource-constrained project scheduling problems by a* search. *Naval Research Logistics*, *37*(1), 61–84.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, *5*(1), 11–24.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*(1), 3–41.

Cai, H., Bian, Y., & Liu, L. (2024). Deep reinforcement learning for solving resource constrained project scheduling problems with resource disruptions. *Robotics and Computer-Integrated Manufacturing*, *85*, Article 102628.

Chandaka, S., Chatterjee, A., & Munshi, S. (2009). Cross-correlation aided support vector machine classifier for classification of eeg signals. *Expert Systems with Applications*, *36*(2), 1329–1336.

Chang, K.-Y., Chen, C.-S., & Hung, Y.-P. (2011). Ordinal hyperplanes ranker with cost sensitivities for age estimation. *vol. 58*, In *CVPR 2011* (pp. 5–592).

Chen, R.-C., & Hsieh, C.-H. (2006). Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications*, *31*(2), 427–435.

Chen, S., & Wang, W. (2009). Decision tree learning for freeway automatic incident detection. *Expert Systems with Applications*, *36*(2), 4101–4105.

Christofides, N., Alvarez-Valdés, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*(3), 262–273.

Coelho, J., & Vanhoucke, M. (2018). An exact composite lower bound strategy for the resource-constrained project scheduling problem. *Computers & Operations Research*, *93*, 135–150.

Coelho, J., & Vanhoucke, M. (2020). Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research*, *121*, Article 104976.

Cooper, D. F. (1976). Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, *22*(11), 1186–1194.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, *38*(12), 1803–1818.

Demeulemeester, E. L., & Herroelen, W. S. (1997). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, *43*(11), 1485–1492.

Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, *6*(1), 17–38.

Farid, D. M., Zhang, L., Rahman, C. M., Hossain, M. A., & Strachan, R. (2014). Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert Systems with Applications*, *41*(4), 1937–1946.

Golab, A., Gooya, E., Falou, A., & Cabon, M. (2023). A convolutional neural network for the resource-constrained project scheduling problem (rcpsp): A new approach. *Decision Science Letters*, *12*(2), 225–238.

Guo, W., Vanhoucke, M., & Coelho, J. (2023). A prediction model for ranking branch-and-bound procedures for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *306*(2), 579–595.

Guo, W., Vanhoucke, M., Coelho, J., & Luo, J. (2021). Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem. *Expert Systems with Applications*, *167*, Article 114116.

Hagenauer, J., & Helbich, M. (2017). A comparative study of machine learning classifiers for modeling travel mode choice. *Expert Systems with Applications*, *78*, 273–282.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207*(1), 1–14.

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*(1), 1–14.

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, *25*(4), 279–302.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (1999). A classification scheme for project scheduling. In *Project scheduling* (pp. 1–26). Springer.

Huang, A. (2008). Similarity measures for text document clustering. *vol. 4*, In *Proceedings of the sixth New zealand computer science research student conference (NZCSRSC2008)* (pp. 9–56). Christchurch, New Zealand.

Jiang, L., Cai, Z., Wang, D., & Jiang, S. (2007). Survey of improving k-nearest-neighbor for classification. *vol. 1*, In *Fourth international conference on fuzzy systems and knowledge discovery (FSKD 2007)* (pp. 679–683). IEEE.

Jiang, L., Wang, S., Li, C., & Zhang, L. (2016). Structure extended multinomial naive bayes. *Information Sciences*, *329*, 346–356.

Jo, T., & Cheng, J. (2014). Improving protein fold recognition by random forest. *vol. 15*, In *BMC bioinformatics* (pp. 1–7). BioMed Central.

Johnson, D. S. (1985). The np-completeness column: An ongoing guide. *Journal of Algorithms*, *6*(3), 434–451.

Klein, R., & Scholl, A. (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, *112*(2), 322–346.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, *174*(1), 23–37.

L., Breiman (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Lewis, D. P., Jebara, T., & Noble, W. S. (2006). Support vector machine learning from heterogeneous data: An empirical analysis using protein sequence and structure. *Bioinformatics*, *22*(22), 2753–2760.

Li, X., He, Z., & Wang, N. (2024). A branch-and-bound algorithm for the proactive resource-constrained project scheduling problem with a robustness maximization objective. *Computers & Operations Research*, *166*, Article 106623.

Liu, S. M., & Chen, J.-H. (2015). A multi-label classification based approach for sentiment classification. *Expert Systems with Applications*, *42*(3), 1083–1093.

Liu, Y., Jin, S., Zhou, J., & Hu, Q. (2023). A branch-and-bound algorithm for the unit-capacity resource constrained project scheduling problem with transfer times. *Computers & Operations Research*, *151*, Article 106097.

Messelis, T., & De Causmaecker, P. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, *233*(3), 511–528.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation. *Management Science*, *44*(5), 714–729.

Mukhlishin, M. F., Saputra, R., & Wibowo, A. (2017). Predicting house sale price using fuzzy logic, artificial neural network and k-nearest neighbor. In *2017 1st international conference on informatics and computational sciences (iCICoS)* (pp. 171–176). IEEE.

Murphy, K. P. (2006). Naive bayes classifiers. *University of British Columbia*, *18*(60), 1–8.

Nadi, A., & Moradi, H. (2019). Increasing the views and reducing the depth in random forest. *Expert Systems with Applications*, *138*, Article 112801.

Nazareth, T., Verma, S., Bhattacharya, S., & Bagchi, A. (1999). The multiple resource constrained project scheduling problem: A breadth-first approach. *European Journal of Operational Research*, *112*(2), 347–366.

Park, S. C., Chapman, B. E., & Zheng, B. (2010). A multistage approach to improve performance of computer-aided detection of pulmonary embolisms depicted on ct images: Preliminary investigation. *IEEE Transactions on Biomedical Engineering*, *58*(6), 1519–1527.

Pascoe, T. L. (1966). Allocation of resources cpm. *Revue FranÇ*, *10*(38), 31.

Patterson, J. H. (1976). Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly*, *23*(1), 95–123.

Patterson, J. H., & Huber, W. D. (1974). A horizon-varying, zero–one approach to project scheduling. *Management Science*, *20*(6), 990–998.

Pawar, P. Y., & Gawande, S. (2012). A comparative study on different types of approaches to text categorization. *International Journal of Machine Learning and Computing*, *2*(4), 423.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416.

Polat, K., & Güneş, S. (2009). A novel hybrid intelligent method based on c4. 5 decision tree classifier and one-against-all approach for multi-class classification problems. *Expert Systems with Applications*, *36*(2), 1587–1592.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27*(3), 221–234.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2021). Classifier chains: A review and perspectives. *Journal of Artificial Intelligence Research*, *70*, 683–718.

S., Tan (2006). An effective refinement strategy for knn text classifier. *Expert Systems with Applications*, *30*(2), 290–298.

S., Creemers (2019). The preemptive stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, *277*(1), 238–247.

Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J. (2021). A reinforcement learning based multi-method approach for stochastic resource constrained project scheduling problems. *Expert Systems with Applications*, *169*, Article 114479.

Sprecher, A., & Kolisch, R. (1996). Psplib-project scheduling problem library. *European Journal of Operational Research*, *96*, 205–216.

Stinson, J. P., Davis, E. W., & Khumawala, B. M. (1978). Multiple resource–constrained scheduling using branch and bound. *AIIE Transactions*, *10*(3), 252–259.

Strahl, W. R., & Gounaris, C. E. (2023). A priority rule for scheduling shared due dates in the resource-constrained project scheduling problem. *Computers & Industrial Engineering*, *183*, Article 109442.

Sun, T.-H., & Tien, F.-C. (2008). Using backpropagation neural network for face recognition with 2d+ 3d hybrid information. *Expert Systems with Applications*, *35*(1–2), 361–372.

Talbot, F. B., & Patterson, J. H. (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, *24*(11), 1163–1174.

Tian, Y., Mei, Y., & Zhang, M. (2024). Learning heuristics via genetic programming for multi-mode resource-constrained project scheduling. In *2024 IEEE congress on evolutionary computation* (pp. 01–08). IEEE.

Tillmann, A. M. (2014). On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, *22*(1), 45–49.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. *vol. 66*, In *Data mining and knowledge discovery handbook* (pp. 7–685). Springer.

Van Eynde, R., & Vanhoucke, M. (2022). New summary measures and datasets for the multi-project scheduling problem. *European Journal of Operational Research*, *299*(3), 853–868.

Vanhoucke, M., & Coelho, J. (2024a). A matheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *319*(3), 711–725.

Vanhoucke, M., & Coelho, J. (2024b). Reducing the feasible solution space of resource-constrained project instances. *Computers & Operations Research*, *165*, Article 106567.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, *187*, 511–524.

Vapnik, V. (1999). *The nature of statistical learning theory*. Springer science & business media.

Wang, Z., Lai, C., Chen, X., Yang, B., Zhao, S., & Bai, X. (2015b). Flood hazard risk assessment model based on random forest. *Journal of Hydrology*, *527*, 1130–1141.

Wang, L., Zeng, Y., & Chen, T. (2015a). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Systems with Applications*, *42*(2), 855–863.

Witten, I. H., & Frank, E. (2002). Data mining: practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, *31*(1), 76–77.

Wu, D. D., Yang, Z., & Liang, L. (2006). Using dea-neural network approach to evaluate branch efficiency of a large canadian bank. *Expert Systems with Applications*, *31*(1), 108–115.

Yiakopoulos, C. T., Gryllias, K. C., & Antoniadis, I. A. (2011). Rolling element bearing fault detection in industrial environments based on a k-means clustering approach. *Expert Systems with Applications*, *38*(3), 2888–2911.

Yu, S.-S., Chu, S.-W., Wang, C.-M., Chan, Y.-K., & Chang, T.-C. (2018). Two improved k-means algorithms. *Applied Soft Computing*, *68*, 747–755.

Zhang, M.-L., Li, Y.-K., Liu, X.-Y., & Geng, X. (2018). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, *12*, 191–202.

Zhang, J., & Wang, Y. (2008). A rough margin based support vector machine. *Information Sciences*, *178*(9), 2204–2214.

Zheng H.-y, Wang, L., & Zheng, X.-l. (2017). Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. *Soft Computing*, *21*, 1537–1548.