Data-driven and Physics-based Modeling and Optimization for Smart Systems

By

Congfang Huang

A proposal submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Industrial and Systems Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2024

Committee Members:

ACKNOWLEDGMENT

The research presented in this dissertation benefited from valuable insights and support of many people. It is my pleasure to express my sincere gratitude to all of them.

First and foremost, my deepest gratitude is owed to my Ph.D. advisor, Prof. Shiyu Zhou, for his sincere guidance, mentoring, and encouragement throughout the graduate study years. His expertise and continual support holistically nurtured my research skills. It has been an honor working under his supervision. I have been so lucky to benefit from not only his expertise in academic research, but also his great personality in all other aspects. He will remain my role model as a scientist, mentor, and teacher.

To my other four committee members, Prof. Dharmaraj Veeramani, Prof. Kaibo Liu, Prof. Robert G Radwin, and Prof. Xiaoping Qian, a special thanks for their time and contribution to part of this research.

I would like to give my sincere gratitude to Prof. Jiong Tang from the University of Connecticut, Prof. Jingshan Li from Tsinghua University, China, and Prof. Yong Chen from the University of Iowa, who are the main collaborators of my research. Without their full support and guidance, my research study would not have been so smooth. I have benefited a lot from their insightful guidance and comments, which helped me a lot in my research.

To all of the industry collaborators at General Motors, Mercury Marine, and Thermastor who have continuously provided us with interesting research motivations, and of course, relevant datasets.

To my beloved parents and my elder sister, for their unconditional love and support. For being a permanent companion during this journey.

To my friends in the lab. Chao, Salman, Jaesung, Akash, Jinwen, Vipul, and Han, thank you for making the office space memorable. For helping with random questions regarding random variables. For introducing me to many new kinds of cuisines.

To my friends in Madison. Yiqin, Wenjun, Ziqian, Sujee, Abhijeet and Yekwon, thank you for your companionship in not only the happy days but also the tough days when going out is hard.

To my lifelong friends. Chuyan, Yuhong, Xinru, Yinan, it is always good to be with you and you are always shining people in my world.

DEDICATION

To Mother, Father and Sister

Contents

List of Figures

List of Tables

ABSTRACT

Smart systems are systems embedded with various types of sensors and smart components (e.g., a robot) for superior systems performance and optimal utilization of resources. Due to the fast development of sensing and information technology, smart modern systems grow rapidly in all fields of engineering applications. Along the rapid development, emergent challenges are posed to systems engineering. As the machines and workstations in modern engineering systems become smart and connected, advanced sensor techniques make it possible for information collection in a quick and accurate manner. Data collected from the smart systems are always high-dimensional and of various formats. Information selection or feature extraction is necessary to analyze the abundant data and avoid wasting computation efforts on non-beneficial information. In addition, the involvement of smart components, such as collaborative robots and automated tools, requires changes in the design of the entire system. While the utilization of smart components can improve both the ergonomics and the productivity of the engineering systems, the resources of available smart components are always limited and the redesign of the existing systems needs the support of optimal decision-making strategies. To address those issues listed above, three tasks are investigated in this report. Specific contributions are made to data-driven and physics-based modeling and optimization for smart engineering systems.

- **A Deep Learning Model for Engine Event Logs Monitoring.** Event logs contain abundant information on manufacturing machines, with time information. A Recurrent Neural Network model using time-to-event data from event logs was established not only to predict the time of the occurrence of a target event of interest but also to interpret, from the trained model, significant events leading to the target event. To improve the performance of the model, sampling techniques and methods dealing with censored data are utilized. The real-world case study shows that the model interpretation algorithm proposed in this work can reveal the underlying physical relationship among events.

- **Structural Fault Diagnosis using Bayesian Optimization.** A Bayesian Optimization method using a multi-output Gaussian process was introduced to solve the

structural fault diagnosis problem. This method utilizes a physics-based high-fidelity finite element model (FE) of the structure and the impedance/admittance measurements from the structure to identify the location and severity of the damage. Thompson sampling approach was used to guide the search for the structural damage in the Bayesian optimization. The proposed method outperformed other benchmark methods on both simulated functions and a real-world structural damage identification problem.

- **Task Allocation in Collaborative Production Systems with Cobots/Robots.** Collaboration between humans and robots has great promise in manufacturing systems. Previous research on cobots/robots allocation focus on the decomposition of tasks for a single workstation into multiple work elements and split them between humans and robots, rather than studying multi-machine systems. To bridge the gap, we handled the allocation problem of limited cobots/robots to manufacturing systems with multiple workstations, considering the precedence relationship between different tasks and the ergonomics concerns. The problem was formulated into a constraint integer programming problem, and the optimal allocation of cobots/robots was obtained in a simulated production system with a reinforcement learning guided evolutionary algorithm.

While the methodologies have been developed in the context of industrial systems, they can be generalized and implemented to similar problems in other fields.

Chapter 1

Introduction

1.1 Background and Motivation

Industrial systems embedded with various types of sensors and smart components are called smart systems. The objectives of embedding sensors and smart components, such as robots, are to improve the performance of the system and to optimize the utilization of resources. Equipping sensors has shown great benefits in a wide range of modern systems, such as power systems, traffic systems, and manufacturing system[3, 66, 125]. Low-cost advanced sensor technologies enable rapid and accurate data collection in modern engineering systems, especially with the development of Industry 4.0 and the Internet of Things (IoT)[89]. In the meantime, the implementation and involvement of smart components also promote the development of modern engineering systems. Taking robots as an instance, the application of robots has led to significant productivity improvement [100] and reduced the ergonomic strain on human workers substantially [47].

Given the unprecedented amount of data collected from the sensors, cutting-edge data-driven models or machine learning models can be implemented to solve a wide variety of tasks, including systems monitoring, diagnosis, prognosis, and optimization [28]. With prompt and accurate monitoring and diagnosis strategies, anomalies in the engineering systems can be detected in the early stage and further consequences can be prevented, which improves the reliability of the systems. Systems prognosis also contributes to a vitally important role in the new era of industrial systems [29]. Not only does it make predictions on the engineering tasks, but prognosis methods also provide useful insights and interpretations of the system structures. Furthermore, the development of smart systems also gives rise to the demand for innovative optimization models of the systems design or decision-making tasks, which should be adaptive to the newly involved smart components in the systems. Moreover, physical information and domain knowledge can be applied or combined into data-driven models to improve the explanation of the models and reveal the underlying mechanisms of the systems.

Despite the profound potential of data-driven and physics-based modeling on smart industrial systems, challenges are also posed to the researchers due to the large amount of information from data collection and the restrictions of applying smart components in the existing systems:

- Data collected from real world systems are not always well-organized or even complete. Data are always in diverse formats and types (e.g. discrete vs. continuous), which may also be incomplete or noisy, requiring appropriate data preprocessing techniques before analysis and modeling or adaptive data handling methods in the model structure.

- Information selection or feature extraction is necessary to analyze the abundant data. High dimensionality and long observation range are common properties of real-world data and we should avoid wasting computation efforts on nonbeneficial data to the target problem.

- The incorporation of domain knowledge and physical information with the data analytics methods is not always trivial. Domain knowledge and physical information are derived from the mechanism of the industrial systems and require further understanding of the systems, which can be difficult to utilize in data-driven models.

- While smart components can contribute a number of benefits to industrial systems, the implementation of smart components always involves various restrictions. The feasibility of the application of smart components always needs to be taken into consideration and should be included in the modeling.

- The trade-off between the impacts of newly involved smart components should be handled. Usually, multiple objectives of one industrial system can not be improved simultaneously. When the smart components are implemented in the system, the design of the system should be capable of balancing the trade-offs between different system performances.

To handle these challenges, innovative data-driven and physics-based modeling techniques are supposed to be developed and tailored to the settings and properties of smart industrial systems.

1.2 Research Objectives

There are a considerable number of potential topics and promising directions arisen from the rapidly developing smart industrial systems and the evolving data-driven modeling techniques. Specifically, with the motivation introduced and the challenges illustrated in Section 1.1, the research objectives of this thesis are determined and focused on the following aspects:

- *Monitoring and prognosis techniques with time-to-event data.* Time-to-event data characterizes the information of industrial machines and systems by associating the occurrence time of different types of events and the type of events. It is a unique type of data structure and is commonly collected along the lifetime of a manufacturing machine or the entire timeline of a production process. Unlike longitudinal data which is always collected on continuous time points and is numerical by nature, Time-to-event data requires appropriate data transformation before the modeling procedure. One common property of time-to-event data collected in industrial systems is the large number of event types included in the dataset since various kinds of events could occur during the operation. Another property of time-to-event data is the long observation time. Instead of many short observed time-to-event data sequences, it is more ubiquitous to have a few long observed data series in a given time-to-event dataset. Though there are only two attributes in a time-to-event dataset, it is widely acknowledged that the time-to-event data contains abundant information and can contribute to a wide variety of analytics tasks. More specifically, in this dissertation, we expect to develop novel monitoring and prognosis techniques that can be adapted to the properties of time-to-event data and can take great advantage of the abundant information embedded in time-to-event data.

- *Diagnosis methods in complex structures.* Fault diagnosis plays a vital role in ensuring the durability and reliability of engineering systems, especially for systems with complex structures. Structural damage in the systems can cause performance degradation and even lead to catastrophic consequences. One of the challenges of the fault identification of complex structures is the lack of direct measurements. While the measurements of systems with simple structures are always trivial and can be derived with low expense, it is usually hard and almost impossible for direct measurements of complex structures during the operation. Conventionally, complicated physical models are applied to characterize the relationship between indirect measurements and structural damages. Nevertheless, when the structures become more complex, the physical models also become more complicated and take much more time to run. With the incredible capability of data-driven modeling techniques, we expect to establish a data-driven and physical-based model to overcome the lack of direct measurement issues and accurately identify structural damage with reduced time expense.

- *Design optimization of production systems with smart components.* Implementation of newly added smart components in industrial production systems requires the appropriate design of the systems to fully realize the capability of smart components and avoid wasting limited resources. Design optimization problems depend extensively on the specific properties of the smart components. Given collaborative robots as the smart components, corresponding concerns include the feasibility and cost of the implementation of collaborative robots on different tasks, the precedence relationship between different tasks, and the ergonomics strain reduction of human workers. Also, the change in processing time and ergonomics strain under different operation alternatives is supposed to be characterized explicitly. With the physical knowledge of the ergonomic strain of human workers, we can formulate a framework for the design optimization problem in a production system when the resources are limited. Furthermore, the optimization problem should be solved with a scalable method and easily generalized to other similar production systems.

In the following chapters of the thesis, each of the research tasks will be investigated thoroughly and the outcomes of the research tasks will be clearly illustrated and analyzed.

1.3 Outline of the Dissertation

With respect to the research objectives listed in Section 1.2, four specific research works are conducted, and future works are presented in this dissertation. The remainder of the dissertation is organized as follows.

- ***Chapter 2:*** *A Deep Learning Model for Event Logs Monitoring and Prognosis.* In this task, a data-driven model using a Recurrent Neural Network model with time-to-event data is developed. The data is collected from event logs and this work not only predicts the time of the occurrence of a target failure event but also interprets significant events leading to the target event. Sampling techniques are applied and the censored data problem is dealt with. Both simulations and real-world case studies are carried out and the proposed model is compared to the traditional survival analysis model.

- ***Chapter 3:*** *Structural Fault Diagnosis using Bayesian Optimization.* In this task, a data-driven and physics-based fault prognostic model is developed for damage in a complex structure. A Bayesian Optimization method using a multi-output Gaussian process is incorporated with a high-fidelity finite element model (FE) to solve the structural damages prognosis problem. This model takes the indirect measurements of impedance/admittance from the structure to identify the location and severity of the damage. Thompson sampling approach is used to guide the search for the structural damage in the proposed model. Simulated functions and a real-world structural damage identification problem are carried out and comparisons are made with the existing benchmark method.

- ***Chapter 4:*** *Resource Allocation in Collaborative Production Systems with Robots/Cobots.* In this task, a physic-based task allocation optimization method for a production system with smart collaborative robots is formulated and solved with a scalable algorithm. Unlike previous works on robot allocation works, a production system with multiple workstations is investigated. In this work, we try to select the workstations to which the robot/cobots are allocated. An integrated performance measure, considering both productivity and ergonomics is proposed and optimized in the proposed model. Formulated into a constrained integer programming problem, multiple simulated studies based on real-world scenarios are conducted and useful insights are concluded.

- ***Chapter 5:*** *Collaborative Production System Design with Ergonomics Concerns and Precedence Constraints.* In ***Chapter 4***, we allocate robots/cobots to selected workstations with a given set of tasks. Extending the research, we focus on the complex task allocation and production line balancing problem in this chapter: a series of unsplittable tasks are allocated to different workstations with constraints on precedence relationship and the constraints of processing cycle time. In addition, the ergonomic strains on the human workers and the implementation cost of robots/cobots are considered in the objective function. We propose to utilize a reinforcement learning guided evolutionary algorithm to solve the problem which improves the scalability of the optimization framework. A case study based on an assembly production system is conducted to demonstrate the effectiveness of our proposed method.

Lastly, in ***Chapter 6***, the current works and contributions are summarized and discussed. Also, the future directions of the thesis are overviewed.

Chapter 2

A Deep Learning Approach for Predicting Critical Events using Event Logs

Abstract

Event logs, comprising data on the occurrence of different types of events and associated times, are commonly collected during the operation of modern industrial machines and systems. It is widely believed that the rich information embedded in event logs can be used to predict the occurrence of critical events. In this paper, we propose a Recurrent Neural Network model using time-to-event data from event logs not only to predict the time of the occurrence of a target event of interest, but also to interpret, from the trained model, significant events leading to the target event. To improve the performance of our model, sampling techniques and methods dealing with censored data are utilized. The proposed model is tested on both simulated data and real-world datasets. Through these comparison studies, we show that the Deep Learning approach can often achieve better prediction performance than the traditional statistical model, such as, the Cox Proportional Hazard (PH) model.

This chapter is based on the paper: **Huang, C.**, Deep, A., Veeramani, D., & Zhou, S. (2021). A Deep Learning Approach for Predicting Critical Events using Event Logs. Quality and Reliability Engineering International, 37(5), pp.2214-2234.

The real-world case study also shows that the model interpretation algorithm proposed in this work can reveal the underlying physical relationship among events.

2.1 Introduction

With the rapid advances in information and communication technology, event logs, which are temporal records of the occurrence of various types of events, are commonly available at both the machine level and the system level in industrial enterprises. For example, the service department of most manufacturers keeps a record of their after-sales service provided for products during the warranty period. Such records contain the occurrence of malfunction/failure events and corresponding repair actions taken for the products over time. Many modern machines are numerically controlled by embedded computers, and various events such as machine activities, critical failures, operator/user actions and task status, are recorded in real time during the machine operation. For example, for material handling equipment, we can obtain an event log containing various event types such as battery status, occurrences of accidents, internal communication error across subsystems, etc. A typical event log is shown in Figure 2.1. The horizontal line in the figure is the time line; the vertical bars with arrowheads represent the occurrence of events, and the letters indicate event types.

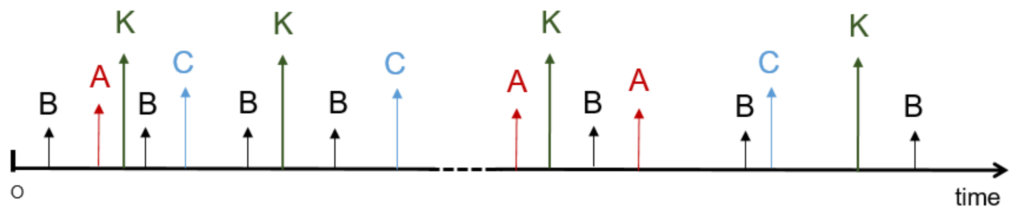


Figure 2.1: Time-to-Event Sequence Data

The event logs often contain rich information regarding system operation and can be used for condition monitoring, event prediction, and maintenance decision making. For example, although some machine failures could occur unexpectedly, it is common for some precursor events to happen before a critical failure event. By modeling the relationship between the precursor events and the critical event, we could predict the occurrence of the critical event when we observe the precursor events. Another example is that by analyzing the service records of a particular product, we can find the occurrence frequency of certain failure modes and the potential relationship between different failure modes. For instance, two different failure types may be related due to an underlying physical linkage between the two types of failure or due to the fact that these two failure modes are caused by a common root cause. Information regarding the relationship between events, when combined with the ability to accurately predict critical events, is very helpful for identifying the root causes of failures and for designing optimal preventive maintenance policies that can reduce unexpected machine downtime and maintenance cost. Thus, establishing a event log based modeling and event prediction method can be highly valuable in industrial practice.

A popular approach for event prediction is the data-driven rule-based method. This method uses temporal mining to identify the event combinations that occur frequently together. Then, these combinations are viewed as event patterns and these patterns are used to predict future events. [49, 51, 52, 73, 78, 142] Most of these works focus on how to find the frequent event patterns in a computationally efficient way. The prediction rule obtained by such methods is often in the following form:

If events A and B occur in the system, *then* event K will occur with confidence (c%)

Although such rules are easy to use, it is difficult for the rule-based method to incorporate the information of time interval between events because such intervals are continuous variables and potentially have infinite number of combinations. It is also not straightforward to combine rules when multiple rules are simultaneously applicable. For example, if both event patterns (A, B), and (C, D) occur and we have two rules linking (A, B) and (C, D) with event K, it is not clear how to adjust the prediction using these two rules simultaneously.

In the realm of statistics, reliability models with covariates provide a mathematical framework for event prediction. [44, 87] Essentially, we can view the occurrence of the critical event as the 'death' of the system, and encode the precursor events as covariates in the system that impact the survival probability of the system. Using such a model, we can link the precursor events with the critical event and predict the occurrence probability of the critical event. Li *et al* adapted this idea and proposed a method for modeling event log data using the Cox proportional hazard (PH) model, a popular reliability model with covariates.[79] The output of the model is the hazard of the critical event, and the covariates of the model include the selected precursor events. One interesting contribution of their work is that they utilize the temporal mining approach to identify frequent event patterns as in the rule-based method, and then encode these identified patterns as covariates in the Cox PH model. However, how to accurately predict the event occurrence based on the model is not discussed. This problem is addressed by Yuan *et al.* [144] They established a formal event prediction scheme based on the fitted Cox PH model and developed a method for systematically evaluating the prediction performance. The methods using reliability models with covariates for event prediction are statistically rigorous. However, to employ these statistical models, certain conditions need to be satisfied. For example, proportional hazard is one condition required by the Cox PH model. However, in practice, these conditions may not be met. Furthermore, the scalability of some statistical models is limited. For example, the most commonly used method in estimating Cox-PH model parameters is the maximum likelihood estimation method. The likelihood function of the Cox-PH model is a complex nonlinear function. If there are a large number of covariates, we may face convergence issue in maximizing the likelihood function.

Significant progress has been made recently in machine learning and artificial intelligence. [5, 88] Many new generic data-driven modeling approaches have been developed, among which deep learning methods has shown to be quite flexible and powerful. Deep learning is a generic nonlinear function approximation method using a neural network framework, which can learn, from the data, relationship between high dimensional inputs and an output. The effectiveness of deep learning comes from its flexible structure. Recent advances in stochastic gradient descent optimization and in GPU-based parallel computing make very large scale deep learning models possible, thereby enhancing the flexibility and effectiveness of deep learning models. Indeed, deep learning methods have been applied to a variety of fields, including computer vision, natural language processing, bioinformatics, medical image analysis, where they have produced results superior to existing methods in many cases. There also exist some works using deep learningfor survival analysis and event log modeling. Katzman *et al* proposed to use the flexibility of deep learning network to extend the conventional Cox PH model.[65] The "Deep Survival" model that uses the basic idea to replace the exponential function of the linear combination of the predictors with the exponential function of a generic nonlinear function of the predictors, where the nonlinear function is represented by a deep neural network. This model relaxed certain restrictions on the Cox PH model, but it does not deal with event log data. In the biomedical field, Choi *et al* proposed a Recurrent Neural Network (RNN) to model the event sequence recorded in Electronic Health Records(EHR). [22] They used the RNN model to predict heart failure onset within a fixed prediction interval based on EHR records in a fixed length preceding window. The method showed promising results but has some limitations in the approach. The method can predict the occurrence of the event within a window, but not the exact occurrence time. The censoring in the data (i.e., the interested event is not observed in the sampled event sequence) is not considered. Furthermore, the interpretation of the learned network is not discussed in their work.

Considering the great potential of deep learning and the limitations in existing literature, we propose a comprehensive event log modeling and critical event prediction approach using deep neural networks. Both the probability of event occurrence in a fixed time window and the exact occurrence time are predicted. Several neuron structures, such as Stacked GRU and Bi-directional GRU, will be used and compared. Data censoring is considered in the deep learning by using Inverse Probability of Censoring Weights (IPCW). An interesting network interpretation algorithm is also developed to identify the significant precursor events using the learned neural network. The deep learning results are compared with the Cox PH model based statistical method in a comprehensive numerical study. The deep learning method has very good scalability due to the gradient descent optimization method used. A case study based on real-world event logs is conducted as well. The study shows superior performance of the deep learning approach in most cases.

The rest of the paper is organized as follows. In Section 2.2, we introduce the proposed deep learning prediction model and illustrate the interpretation algorithm for the trained model. In Section 2.3, simulation results using different model structures are compared on simulated datasets. In Section 2.4, we implement our model on a real-world dataset and show its effectiveness.

2.2 Deep Learning Method for Event Prediction

In this section, we first show the data preparation step that transforms the raw data from the event into a training dataset for the deep learning neural network. After the data preparation, we introduce the deep learning network architecture. Finally, both the prediction and the trained model interpretation are explained.

2.2.1 Data Preparation from Event Logs

As shown in Figure 2.2, we consider multiple event logs, each collected during the operation of one specific unit (e.g., a car, a machine, etc.) in the field. We assume that these units are of the same type, and thus the event sequences contain some common information. However, since the units are operated independently, these event sequences are assumed to be independent of each other.

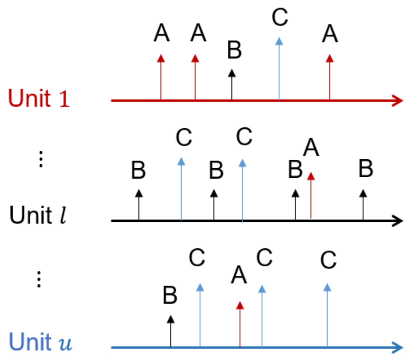


Figure 2.2: Example of Time-to-event Sequence Data

In practice, an event log for a specific unit could be very long covering the events from years of operations. However, in most cases, we only need to consider a certain number of recent events (or events within a certain time window prior to the current time instance) for the purpose of prediction. To prepare the event logs for training the deep learning network, we apply a windowed sampling approach to the raw data. As shown in Figure 2.3, the windowed sampling method selects windows from a long event sequence with certain rules. For example, the 'n-gram' method, which is often used in Natural Language Processing, uses a window containing a fixed number of events and shifts the window from the beginning to the end of the event sequence. In this way, multiple sub-event sequences can be obtained from an original long event sequence. Other rules of window selection can also be used, such as fixed time duration for the window, random selection of window locations, etc. In this work, we follow the 'n-gram' method, and use different window lengths, m, and compare the results.

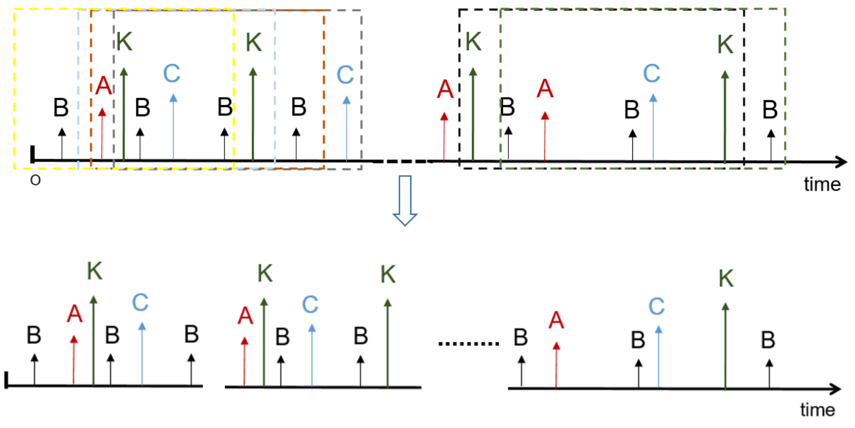


Figure 2.3: Illustration of Window Technique

We use the symbol $X \equiv \{Y, H\}$ to denote a single event, which consists of the event type Y and the time from prior event H (H is zero for the first event in the sequence). We use $\mathbf{X}$ to denote a event sequence consisting of multiple events. For example, we can denote the event sequence from the lth unit as $\{\mathbf{X}^l\}_{l=1}^{u}$, where u is the total number of units. By applying the windowed sampling approach, we can obtain multiple event sequences from $\mathbf{X}^l$ as $\{\mathbf{X}_i^l = (X_{i,1}^l, X_{i,2}^l, \ldots, X_{i,m}^l)\}_{i=1}^{N^l}$, where m is the size of the window and N^l is the number of windowed samples in $\mathbf{X}^l$. In the 'n-gram' method, let s be the step in which we slide the window (i.e., the number of events that separate two adjacent windows) and r^l be the total number of events in $\mathbf{X}^l$, then $N^l = [\frac{r^l - m}{s}]$, which is the floor round number of $\frac{r^l-m}{s}$. It is easy to note that $X_{i,j}^l$ is the $((i-1)*s+j)$-th event in $\mathbf{X}^l$. After the windowed sampling, we have a set of event sequences with fixed length of m. To simplify the notation, we will omit the unit index l in remainder of this paper and simply denote the available event sequence data as $\{\mathbf{X}_i = \{X_{i,j}\}_{j=1}^{m}\}_{i=1}^{N}$, where N is the total number of available event sequences, which are obtained using windowed sampling based on the original event sequences from multiple units.

Knowing an event sequence $\mathbf{X}_i$, we want to predict the occurrence of a critical event of interest, such as a failure event. In other words, we want to predict the time T_i to the next critical event from the last event in $\mathbf{X}_i$ or a boolean value F_i whether the critical event will occur in a preset interval I. If the failure happens in interval I, $F_i = 1$, otherwise, $F_i = 0$. Combining the observed sequence $\mathbf{X}_i$ with the two kinds of responses, we can have the dataset for training deep neural network as $\{\mathbf{X}_i, T_i\}_{i=1}^N$ and $\{\mathbf{X}_i, F_i\}_{i=1}^N$.

To train a neural network, we need to transform the event sequences $\mathbf{X}_i$ into a numerical tensor form. Here we adopt the one-hot encoding method as shown in Figure 2.5, where $t_{i,j}$ denotes the time instance of the occurrence of the jth event in $\mathbf{X}_i$. The original event sequence is shown in Table 2.1. For a time-to-event sequence $X_i = \{Y_i, H_i\}$, where the event type sequence $Y_i = (A, B, A, C, C, \ldots, B)$, and the event time interval sequence $H_i = (0, t_{i,2} - t_{i,1}, t_{i,3} - t_{i,2}, \ldots, t_{i,m} - t_{i,m-1})$. Also, event-wise, $X_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,m})$, where $X_{i,j} = \{Y_{i,j}, H_{i,j}\}$. Specifically, $X_{i,2} = \{Y_{i,2}, H_{i,2}\} = \{(0, 1, 0, \ldots, 0), t_{i,2} - t_{i,1}\}$. In the first n_t row of $X_{i,2}$, the only nonzero term 1 is corresponding to the row of event type B. The meaning of $X_{i,2}$ is that the second event of the ith time-to-event sequence is of event type B and occurs after a time interval of $t_{i,2} - t_{i,1}$, after the previous event occurs.

$$
\mathrm{X}_i = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 1 & 0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & 1 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & t_{i,2} - t_{i,1} & t_{i,3} - t_{i,2} & t_{i,4} - t_{i,3} & t_{i,5} - t_{i,4} & \cdots & t_{i,m} - t_{i,m-1}
\end{bmatrix}
$$

Figure 2.4: Matrix Form of $\mathbf{X}_i$

Event No.	1	2	3	4	5	...	$m-1$	m
Event Type	A	B	A	C	C	...	D	B
Event Time	$t_{i,1}$	$t_{i,2}$	$t_{i,3}$	$t_{i,4}$	$t_{i,4}$	...	$t_{i,m-1}$	$t_{i,m}$

Table 2.1: Original Data Sequence $\mathbf{X}_i$

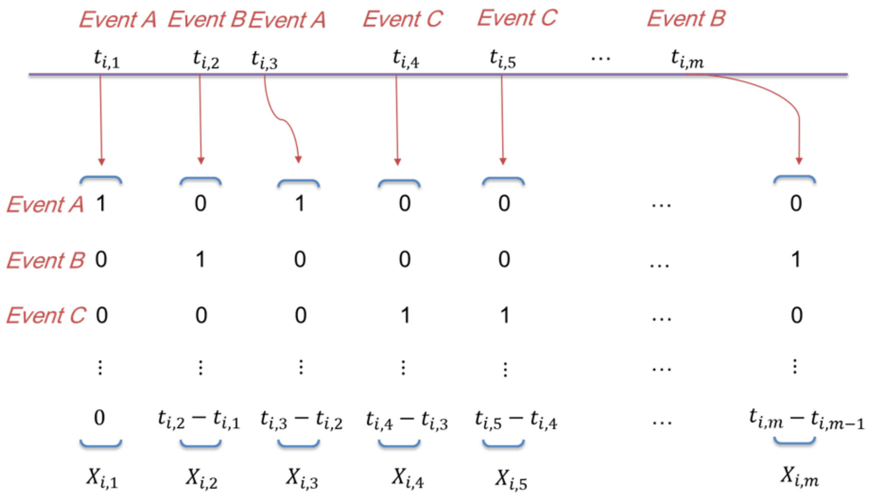


Figure 2.5: Illustration of Columns of $\mathbf{X}_i$

A similar encoding method is also used by Choi *et al* [22] in Electronic Healthcare Record datasets. This method encodes an event sequence with time-to event and event type information to a matrix, where the time-to-event is in the last row and $0, 1$ in the other rows indicate the event type. The total number of rows is the total number of event types plus 1.

2.2.2 Deep Neural Network Structure and Its Training

The basic structure of the proposed deep neural network for event prediction is shown in Figure 2.6. For the ith event sequence $\mathbf{X}_i$, we input it to the RNN layer first, and then the output of the RNN layers would go through a series of dense layers. RNN has been widely used in sequential data analysis in recent years. Different from other networks, RNN has a hidden state for each timestep which contains the information of all the data from before the timestep. In this way, RNN outperforms other networks dealing with sequential data. [55] Event logs data is naturally sequenced data and the temporal relationship of the events is what we are interested in. The parameters of RNN could study the underlying relationships between events and the structure of RNN can fit the event logs better than other deep learning networks. In this way, we could expect a more accurate prediction on the failure event time from a given Event Log data sequence. Thus, utilizing the RNN model on Event logs data is reasonable. As the RNN model attracts more attentions in recent years, various RNN structures have been invented, each with specific advantages. Gated Recurrent Unit (GRU) introduced by Cho *et al*[21], is known for its ability to deal with the vanishing gradient problem and has a good long term memory. Besides GRU, which improves the model performance by modify the internal structures of neurons, external structures for neurons have been developed. Bi-directional RNN (BRNN) [115] can include the information from both the previous and the next timsetep to the network output. This is appropriate in the event logs setting, where an event is correlated with both the previous events and the following events. With the property that a Recurrent Neural layer can return the whole sequence of hidden states, stacking Recurrent layers[46] can also improve the model performance with a more complicated network with the additional depth of network. These network structures are helpful for temporal event prediction and fit the characteristics of the data. In our work, we used different structures of RNNs and compared their performances in event-log based prediction.

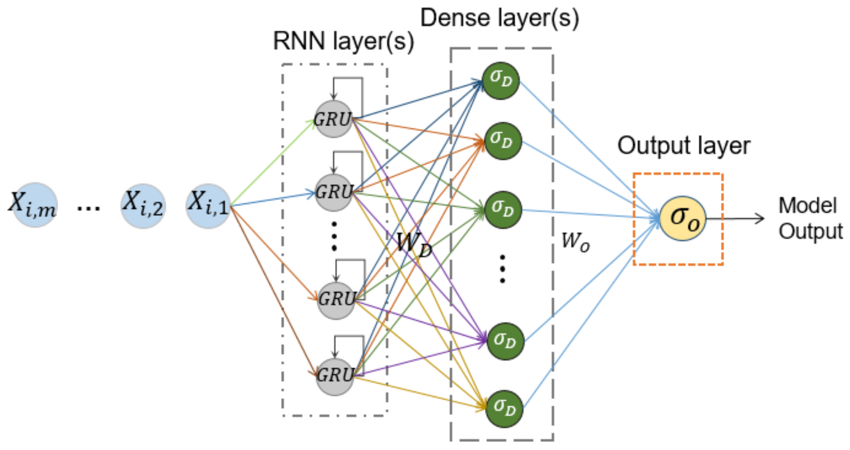


Figure 2.6: Basic Structure of the Proposed Deep Neural Network

In the RNN layer, hidden states are calculated iteratively and passed down to the next states. Equation (2.1) and Figure 2.7 show the formulations in a GRU layer [21]. $X_{i,j}$ is the input vector of the jth timestep which represents the jth event's information of $\mathbf{X}_i$ and $h_{i,j}$ is the output state of the jth timestep. $z_{i,j}$ and $r_{i,j}$ are the update gate vector and the reset gate state respectively. $\hat{h}_{i,j}$ is the candidate activation vector. In this encoding formulation, the hidden state is forced to ignore the previous hidden state and reset with current input only when the reset state $z_{i,j}$ is close to 0. On the other hand, the update gate controls how much information from the previous hidden state will contribute to the current hidden state.

$$\begin{aligned}
z_{i,j} &= \sigma_g(\mathbf{W}_z[h_{i,j-1}, X_{i,j}]) \\
r_{i,j} &= \sigma_g(\mathbf{W}_r[h_{i,j-1}, X_{i,j}]) \\
\hat{h}_{i,j} &= \sigma_h(\mathbf{W}[r_{i,j} * h_{i,j-1}, X_{i,j}]) \\
h_{i,j} &= (1 - z_{i,j})h_{i,j-1} + z_{i,j}\hat{h}_{i,j}, \quad 1 < j \leq m
\end{aligned} \tag{2.1}$$

In the setting of event logs, when the event sequence is very long, an event occurs long time before the failure event probably has little effect on the time to the failure event. To avoid the accumulation of the information from event far from the failure event, we utilize this structure to make sure the network keep the information of new input event that is near to the failure event. The update gate and reset gate can control how much information we gain from the previous event in each timestep. If in one timestep, the reset gate is close to 0, then the previous event can barely influence the output of the current state. The controlled information gain in each timestep can contribute to a better fit of the training model. In this way, the gated structure add flexibility to drop unnecessary information to make a more accurate prediction.

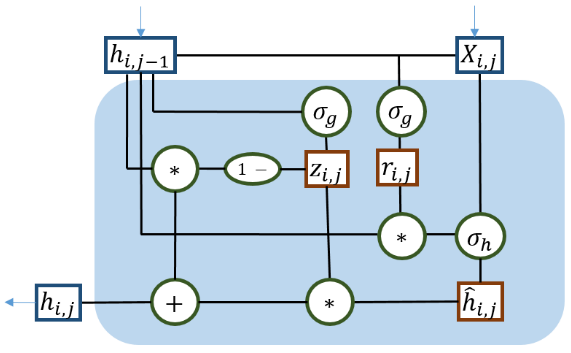


Figure 2.7: Internal Structure of GRU

The output of the GRU layer R_i is the last timestep's output $h_{i,m}$. $\mathbf{W}_z$, $\mathbf{W}_r$, $\mathbf{W}$ and b are the parameter matrices and vector, which will be determined through the training process. σ_g and σ_h are the activation functions, which can be set as different nonlinear functions. By default, σ_g is set as the Sigmoid function and σ_h is set as the hyperbolic tangent function:

$$\sigma_g(x) = \frac{e^x}{e^x+1}, \quad \sigma_h(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{2.2}$$

Besides the inside neuron structure we consider, Bi-directional RNNs can add relationship between neurons in the network. Taking Bi-directional GRU as an example, there are both forward iterated hidden states $\overrightarrow{h}_{i,j}, j = 1, \ldots, m$ and backward states $\overleftarrow{h}_{i,j}, j = 1, \ldots, m$. The architecture of iteration in one layer is shown in Figure 2.8.

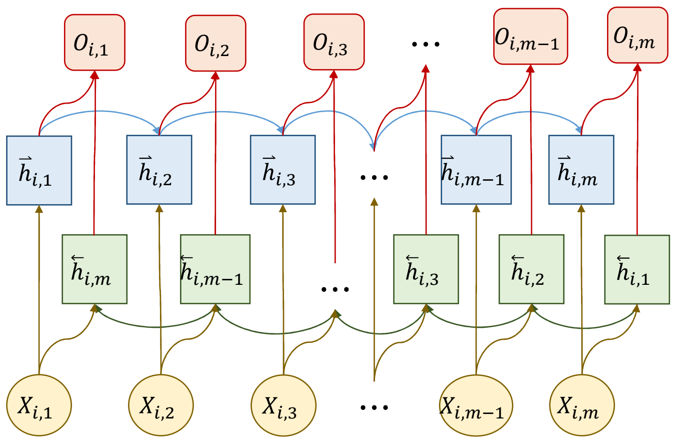

$O_{i,1}$
$O_{i,2}$
$O_{i,3}$
…
$O_{i,m-1}$
$O_{i,m}$
$\overrightarrow{h}_{i,1}$
$\overrightarrow{h}_{i,2}$
$\overrightarrow{h}_{i,3}$
…
$\overrightarrow{h}_{i,m-1}$
$\overrightarrow{h}_{i,m}$
$\overleftarrow{h}_{i,m}$
$\overleftarrow{h}_{i,m-1}$
…
$\overleftarrow{h}_{i,3}$
$\overleftarrow{h}_{i,2}$
$\overleftarrow{h}_{i,1}$
$X_{i,1}$
$X_{i,2}$
$X_{i,3}$
…
$X_{i,m-1}$
$X_{i,m}$

Figure 2.8: Architecture of BGRU

We have:

$$
\begin{aligned}
\overrightarrow{h}_{i,j} &= (1-\overrightarrow{z}_{i,j})\,\overrightarrow{h}_{i,j-1} + \overrightarrow{z}_{i,j}\,\hat{\overrightarrow{h}}_{i,j}, \quad 1 < j \le m \\
\overleftarrow{h}_{i,j} &= (1-\overleftarrow{z}_{i,j})\,\overleftarrow{h}_{i,j-1} + \overleftarrow{z}_{i,j}\,\hat{\overleftarrow{h}}_{i,j}, \quad 1 < j \le m
\end{aligned}
\tag{2.3}
$$

, where $\overrightarrow{z}_{i,j}$, $\overleftarrow{z}_{i,j}$ and $\hat{\overrightarrow{h}}_{i,j}$, $\hat{\overleftarrow{h}}_{i,j}$ are calculated in the similar process as Equation (2.1). Then, hidden states $\hat{\overrightarrow{h}}_{i,j}$ and $\hat{\overleftarrow{h}}_{i,m-j+1}$ are concatenated as $H_j, j = 1, \ldots, m$. Next the output of the BGRU layer R_i is the output of the last hidden state $O_m = \mathbf{W_B} H_m + b_B$, where $\mathbf{W_B}$ and b_B are the parameters to be estimated in training. Another architecture we utilize is the stacked GRU, where we can add layer to the GRU layer. As shown in Figure 2.9, the input of the stacked GRU layer is the corresponding hidden states of the previous GRU layer.

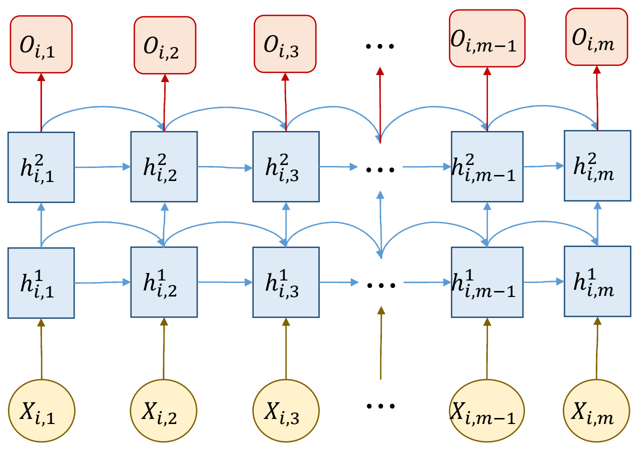


Figure 2.9: Architecture of Stacked GRU

The calculation of the first layer is the same as Equation (2.1). For the second layer, we calculate the states by Equation (2.4). Similar as the mechanism of single layer GRU, $z^2_{i,j}$ and $r^2_{i,j}$ are the update gate vector and reset gate vector that can control the information gains of the neurons. The output of the stacked GRU R_i is the output of the last hidden state in the second layer $O_m = \mathbf{W_S} H_m + b_S$, where $\mathbf{W_S}$ and b_S are the parameters to be estimated in training.

$$\begin{aligned}
z^2_{i,j} &= \sigma_g(\mathbf{W}^2_z[h^2_{i,j-1}, h^1_{i,j}]) \\
r^2_{i,j} &= \sigma_g(\mathbf{W}^2_r[h^2_{i,j-1}, h^1_{i,j}]) \\
\hat{h}^2_{i,j} &= \sigma_h(\mathbf{W}^2[r^2_{i,j} * h^2_{i,j-1}, h^1_{i,j}]) \\
h^2_{i,j} &= (1 - z^2_{i,j})h^2_{i,j-1} + z^2_{i,j}\hat{h}^2_{i,j},
\end{aligned} \tag{2.4}$$

After the RNN layers, we include dense layers, and similar to the formulations in RNN layers, we have activation function σ_D and parameters $\mathbf{W}_D$ and b_D.

$$f_D(R_i) = \sigma_D(\mathbf{W}_D R_i + b_D) \tag{2.5}$$

The activation function σ_o in the output layer depends on the output of the model. We use a Sigmoid function (2.6) to transform the output into a value in $[0, 1]$, when we are predicting F_i.

$$P_i = \frac{1}{1 + exp\{-(\mathbf{W}_o f_D(R_i) + b_o)\}}, \tag{2.6}$$

When the prediction goal is T_i, σ_o becomes an identity function, i.e., $T_i^p = f_D(R_i)$.

To illustrate the number of parameters that need to be trained in a model, an example of a model with 20 event types is shown in Table 2.2:

Table 2.2: An Example of Parameters

Layer Name	# of nodes	# of parameters
GRU	32	5088
Dense_Relu	32	1056
Dense_Tanh	16	528
Output	1	17

To train the network to get estimates of the unknown model parameters, we need to define a loss function according to the model output. Then we can employ the stochastic gradient method to obtain the parameters values through minimizing the loss functions. Specifically, if we want to predict the occurrence of the critical event, F_i, given $\mathbf{X}_i$, we can use the ‘binary cross-entropy’ (2.7) as the loss function.

$$Loss_1 = -\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (F_i \log(P_i) + (1 - F_i)\log(1 - P_i)))w_i \tag{2.7}$$

where P_i is the network output given by (2.6) and N_{train} is the number of training samples. w_i is the sample weight of all the ith data sequence. If we want to predict the time to the critical event, we can use the mean square error (MSE) (2.8) as the loss function,

$$Loss_2 = -\frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (T_i - T_i^p)^2 w_i \tag{2.8}$$

After the model is trained, we need to test and evaluate its prediction performance on a test dataset that has not been used in the training process. If the prediction target is the occurrence of the critical event, then we can use the prediction accuracy defined in (2.9) as the performance measure,

$$ACC = \frac{\#\text{ of } F_i = F_i^p}{N_{test}} \tag{2.9}$$

where N_{test} is the testing sample size and $F_i^p = 1$, if $P_i > 0.5$ or $F_i^p = 1$, otherwise.

If the prediction target is the time to the critical event, we can use the Mean Absolute Error (2.10) as the performance measure

$$MAE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} |T_i - T_i^p| \tag{2.10}$$

The training procedure of the proposed model on a given data set is shown in Algorithm 1. We apply the "Adam" optimizer for the parameter training.[71]

Algorithm 1 Training Procedure of The Proposed Model

```
1: Split training, validation and test data
   *Training*
2: for Models with different network structures and hyper-parameters do
3:     Initialize model parameters: $\theta = \{\mathbf{W}_D, \mathbf{W}_o, b_D, b_o\} \leftarrow \theta_0$
4:     while validation loss is keep reducing do
5:         Update the parameters by Adam optimizer: $\theta = Adam(\theta)$
6:         Calculate the validation loss $Loss_\gamma^v = Loss_\gamma(\theta, \mathbf{X}_{validation}), \gamma = 1, 2$
7:     end while
8:     Save the optimized model parameters $\theta$ and the validation loss $Loss_\gamma^v$
9: end for
10: Compare the validation loss of different models
   *Testing*
11: Apply the best fit model on the testing data
```

2.2.3 Adjustment for Censored Data

In event data, we often have censored observations. Censoring is a condition in which the value of the observation is only partially known. For example, when we observe the occurrence of a failure event but the failure does not occur until the present time, we will know the failure occurs later but will not know the exact occurrence time. This is a censored observation and is often called ‘right’ censored. In some data collection schemes, the percentage of censored data might be high, for example, due to the limited period of observations.

Since we do not have the exact observation of event occurrence time, the censored data cannot be directly used for training or testing the deep learning network. However, if the percentage of censored data is high, the information embedded in the censored data will be lost if they are simply discarded. Most of the existing methods in machine learning replace or modify the censored data based on the uncensored data. However, those generated data are synthetic data which could again lead to inaccurate predictions. Inverse Probability of Censoring Weight (IPCW) is recently proposed to deal with right-censored time-to-event data for machine learning tasks.[135] A promising data pre-processing method is introduced, which gives the weights to uncensored data based on the inverse probability of censoring weights. Here we adopt this algorithm for our deep learning model as follows.

For a given set of units and let i be the unit index. Let T_i represent the time that the event occurs, C_i denote the censoring time. Let d_i^* be the number of units that are censored at time V_i, where $V_i = min\{T_i, C_i\}$, and n_i be the number of units that has not been censored or failed at time V_i. Then, we can define a function $G(t) = P(C_i > t)$ from the Kaplan-Meier estimator of the survival distribution of the censoring time as

$$\hat{G}(t) = \prod_{i:V_i<t} \frac{n_i - d_i^*}{n_i}. \tag{2.11}$$

For each individual i' in the set, an inverse probability of censoring weight is defined as

$$w_{i'} = \frac{1}{\hat{G}(V_{i'})}, T_{i'} < C_{i'} \tag{2.12}$$

For uncensored event sequences, we can apply the corresponding weight obtained in (2.12) into the loss function in (2.7) and (2.8) during the network training process. In this way, the information in the censored observations are utilized indirectly in the model training process. The intuition behind the IPCW method is easy to understand: It gives a higher weight to the uncensored unit with a large time-to-event value, since there is a higher probability for a unit similar to this unit to be censored. In other words, the uncensored unit with a long life should have more similar censored units, and thus it needs to represent more units and be given a higher weight. The rigorous proof of IPCW can be found. [130] We would like to point out that in some cases, $\hat{G}(V_{i'})$ could be very small due to limited number of observations, which will lead to a very large weight. For computational stability, we often need to put an upper limit (e.g., a value below 10) on the weight in the model training.

We want to mention that there are some assumptions of applying IPCW [130, 131]. Essentially, to satisfy the IPCW condition, the censoring should be independent with failure event occurrence and the subject. Compared to the time-to-event data in clinical studies, the conditions of IPCW methods are easier to be satisfied in industrial systems, where censoring are often caused by outside random factors that are not related to the features of the machines or the failure event. For example, the censoring of an event log from a machine in industrial setting is often caused by the communication error (e.g., the internet is down), which is not related with the machine itself. There can also be instances where the censoring is informative, for example, we stopped the machine because it is near the failure. In such situations, censoring is dependent, hence the assumptions of IPCW may not remain valid. To handle such situations, we can use methods like Copula based approaches[31] and Tree-based methods[96], and we leave that as future work.

2.2.4 Model Interpretation

In practice, in addition to making predictions on the critical event, people often want to find out which event is a good preceding indicator for the critical event. This requires us to be able to interpret the trained deep learning neural network. Unfortunately, RNNs typically lack good interpretability. [45] Here we propose an easy-to-implement event deleting method to identify the important indicator event for the critical event through the trained deep neural network.

For an one-hot encoded event sequence $\mathbf{X}_i$, we can replace the $'1'$s that represents event type y by $'0'$ to generate a new event sequence that does not include event type y. By comparing the difference of predicted probability that the critical event would happen, we can quantify the influence of event type y on the occurrence of the critical event. This idea is illustrated in Figure 2.10.

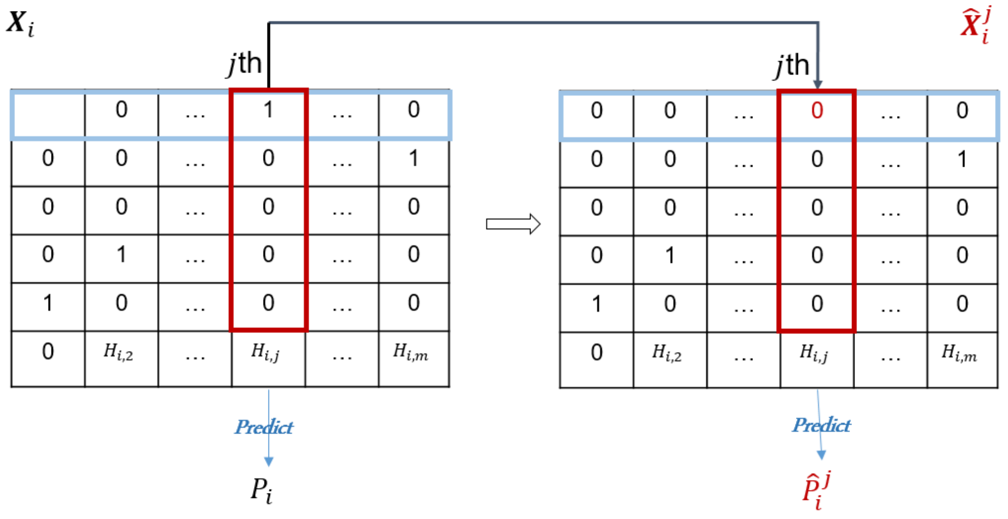


Figure 2.10: Deleting Event y Information

The detailed algorithm is as follows. Assume we have N data sequences. For the jth event occurrence in the ith data sample, we input the data after replacing the $'1'$ in the jth column with $'0'$ and get a new predicted probability $\hat{P_i^j}$. We denote the origin probability of the ith sample as P_i and calculate the difference:

$$D_i^j = \hat{P_i^j} - P_i \tag{2.13}$$

Suppose event y happens N_y times in the N sample event sequence we picked. The average difference of event y is

$$D_y = \frac{\sum_{Y_{i,j}=y} D_i^j}{N_y} \tag{2.14}$$

By ranking D_y, which is called influence index of all the event types, we can find the most influential events. The strength of this algorithm is that we do not need to generate simulated data. Instead, we can directly utilize the existing data and retain the real-time information from the original data.

Here is the pseudocode for influence index D_y calculation:

Algorithm 2 Calculate D_y for Each Event Type y

```
 1: $sum(D_y) \leftarrow 0, N_y \leftarrow 0$
 2: for $1 \le i \le N$ do
 3:     for $1 \le j \le m$ do
 4:         Let $P_i$ be the prediction result of the trained model with input $\mathbf{X}_i$
 5:         $\hat{\mathbf{X}}_i^j \leftarrow \mathbf{X}_i$,
 6:         Flip the '1' in the $j$th column of $\hat{\mathbf{X}}_i^j$ into '0'
 7:         Let $\hat{P_i^j}$ be the prediction result of the trained model with input $\hat{\mathbf{X}}_i^j$,
 8:         $D_i^j \leftarrow abs(\hat{P_i^j} - P_i)$
 9:         if the $j$th event of the $i$th individual is event type $y$ then
10:             $sum(D_y) \leftarrow sum(D_y) + D_i^j, N_y \leftarrow N_y + 1$
11:         end if
12:     end for
13: end for
14: $D_y \leftarrow \frac{sum(D_y)}{N_y}$
```

2.3 Numerical Study

This section we describe the numerical experiments we conducted, and compare the prediction performance of several aforementioned models. First, we show two settings of data generation. Then, we describe several variants of models considered for prediction. Finally, we report and compare the results obtained from this numerical study.

2.3.1 Data generation

As per our motivation that some events have strong association with the occurrence of the event of interest, denoted as the Kth event, also called target event, we generate event sequences consisting of several events. Some of these events have no bearing on the target event's occurrence, while some have strong influence on the occurrence of the target event. We can those events as the *key events*. In the two settings, we generate the target event through simple pattern rules. In total, we use 20 different types of events in the experiments.

For setting I, we first generate independent $K-1$ event variables following a specified distribution. Thereafter, we specify certain rules for the Kth event generation. This setting can be interpreted as if certain past pattern have triggered the onset of the target event, and the target event's time occurs after a specified duration. Four sub-cases are considered for this setting which is tabulated in Table 2.3.

Algorithm 3 Data Generation for Setting (I)

```
   Event time generation:
1: Initialize number of units $N = 5000$, number of event-types except the target event, i.e.
   $K - 1 = 19$.
2: Specify baseline parameters of each event-type, and generate $N$ set of observations.
3: Generate target event times using the rule specified in Table 2.3.
   Censoring:
4: For each $i \in N$, generate censoring/follow-up times $(\tau_i)$ of each unit. Let $\tau_i \sim Weib(\cdot)$.
   We use different values of censoring times for different cases.
5: For each $i \in N, \delta_{ki} = 0$ if $T_{ki} > \tau_i,$ else $\delta_{ki} = 1, \forall k \in \mathbf{K}$.
6: Update $T_{ki} = \min(\tau_i, T_{ki})$.
```

Table 2.3: Setting I Rules Considered

Sub-case	Pattern/Rule
1	$T_K = T_{10} + 1.5$
2	**if** $T_1 < T_2, then, T_K = T_2 + 1,$ **else**, $T_K = T_1 + 5$
3	$T_K = \mathbf{Max}(T_1, T_2, T_3) + 10$
4	$T_K = \mathbf{Max}(T_1, T_2, T_3, T_4) + 10$

Table 2.4: Setting II Rules Considered

Sub-case	Pattern/Rule
1	$T_K = T_1 + \mathcal{N}(5, 0.5)$
2	**if** $T_1 < T_2$, then, $T_K = T_2 + \mathcal{N}(1, 0.5)$, **else**, $T_K = T_1 + \mathcal{N}(5, 0.5)$
3	$T_K = \mathbf{Max}(T_1, T_2, T_3) + \mathcal{N}(5, 0.5)$
4	$T_K = \mathbf{Max}(T_1, T_2, T_3, T_4) + \mathcal{N}(5, 0.5)$

In setting II, we use similar rules for the Kth event generation. However, the Kth event in itself follows a random distribution. Under this setting, the target event can occur before the complete pattern is observed. The sub-cases are tabulated in Appendix 2.6.1. The algorithm to generate data under this setting is the same as Algorithm 3, except the rules are now generated as per Table 2.4.

2.3.2 Cases Considered

We present three cases for comparing the prediction performance. These cases are briefly discussed below:

- ***Case A: IPCW vs Without IPCW***: The first comparison we implement is to show the effectiveness of the IPCW method for censored data. Different censored rates are taken into consideration and the parameter used to generate datasets is shown in Appendix 2.6.2.

- ***Case B: RNN vs Flatten vs BGRU***: In this case, we compare the performance of different network structures. We considered stacked GRU, GRU and Bidirectional GRU, which provides us with a higher accuracy and lower mean absolute error in most of the cases. Also, to illustrate the improvement of RNNs, we performed the results with a network with flattern layers which has only dense layers.

- ***Case C: Cox PH vs RNN***: In this case, the traditional statistical approach Cox [25], called Cox PH regression, is compared against the best fit RNN based deep learning method we decide in *Case B*. a brief review of the Cox PH method is provided in the Appendix 2.6.3.

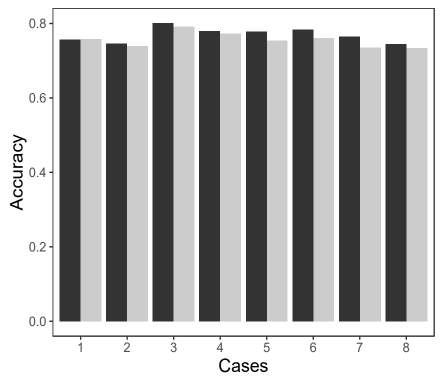
For model evaluation and validation, we generate 5000 sample event sequences for each model and use a training ratio of 0.8. For deep learning models, we also separate 20% of training data as validation data. In other words, we treat 80% of available data as the historical dataset, and use it to obtain offline-stage parameter estimates. Thereafter, we use the remaining data for testing the performance.

We use two criteria to compare the prediction results: Accuracy (2.9) and Mean Absolute Error (2.10) .

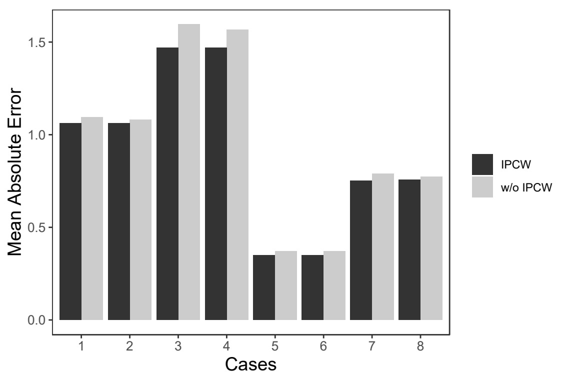
2.3.3 Results and Discussion

2.3.3.1 Effectiveness of IPCW for Censored Data

As shown in Figure 2.11, the IPCW method improved the Accuracy and the Mean Absolute Error. Though we generated the dataset by setting the parameters of the distribution of trigger events, we considered datasets with censored rates varying from 10% to 50%. As shown in Figure 2.11, the IPCW method improved the Accuracy and the Mean Absolute Error in all the different censored rate situations.



(a) Accuracy



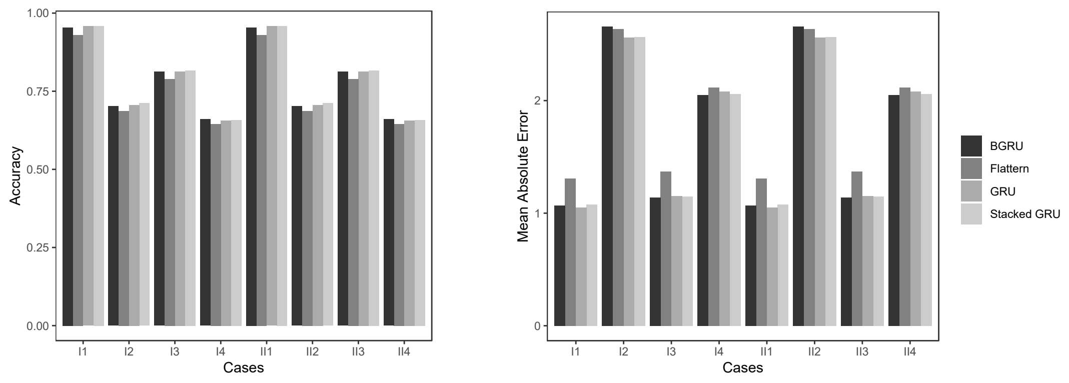
(b) MAE

Figure 2.11: Effectiveness of IPCW for Censored Data

Moreover, in the results of IPCW, we can see that the MAE of different censored data from the original data set were very close. This further show that IPCW can compensate for the influence of censoring.

2.3.3.2 Deep Learning Results with Different Model Structures

In this section, we compare the performances of different neuron structures. We present simulations on simple GRU, stacked GRU, Bidirectional GRU and simple flatten layer (DNN model) respectively with the same data input in Appendix 2.6.1. From Figure 2.12, the RNN structure improves the prediction results significantly. Furthermore, the Bidirectional GRU increases the accuracy and decreases the Mean Absolute Error from the traditional GRU layer in most of the cases. This result is not surprising, since the Bidirectional GRU should fit the data better as it includes a hidden state from the information of future events.

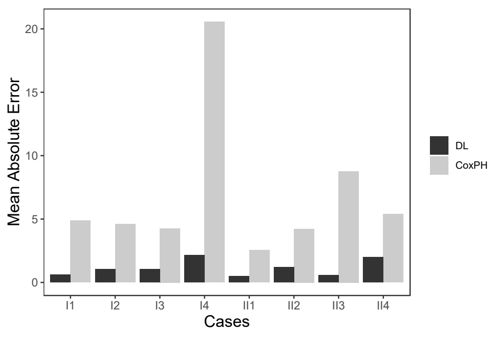


(a) Accuracy (b) MAE

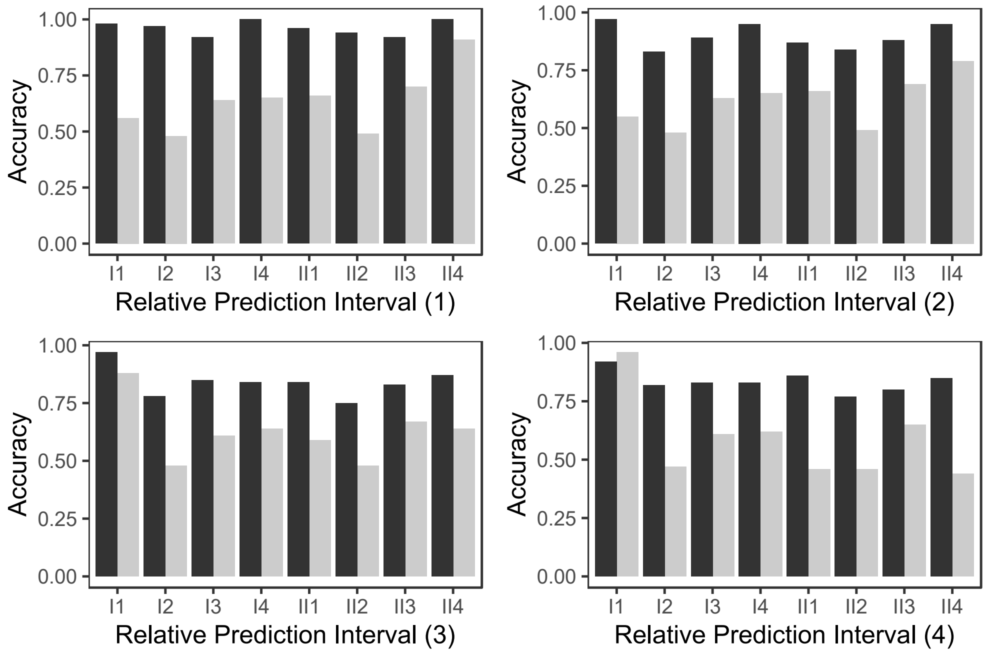
Figure 2.12: Comparison of Different Network Structures

2.3.3.3 RNN vs Cox PH

After finding the best neuron structure, we compare the selected network structure and the CoxPH model on the eight datasets in Appendix 2.6.1. As shown in Figure 2.13a, the model we proposed has a lower MAE than the CoxPH model. For the accuracy evaluation results, considering in the actual prediction, we consider different normalized relatively prediction interval length. As shown in Figure 2.13, our proposed model obtain higher accuracy than the Cox PH model in almost all cases.



(a) Mean Absolute Error



(b) Accuracy under Different Relative Prediction Interval

Figure 2.13: Comparison between Cox PH model based method and deep learning based method

2.3.3.4 Results for Identifying Influential Events

As shown in the generation of simulation data in Tables 2.3 and 2.4, there are close relationships between the target event and several of other events. Using the method we proposed to predict the most influential events, we obtain the influence index on the generated datasets. We can conclude from Figure 2.14 that the key events that are closely associated with the target event are well distinguished from other unrelated events.

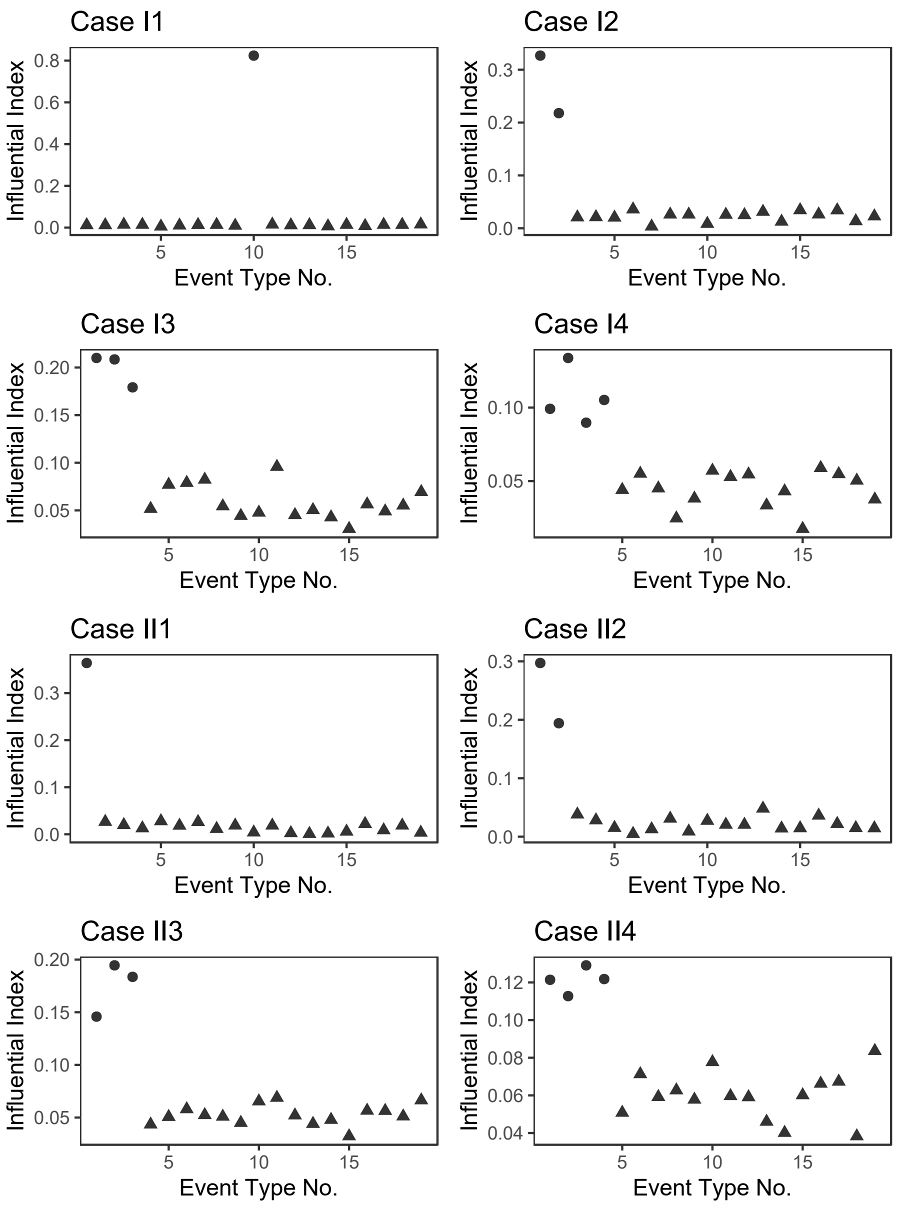


Figure 2.14: Simulation Results of Model Interpretation

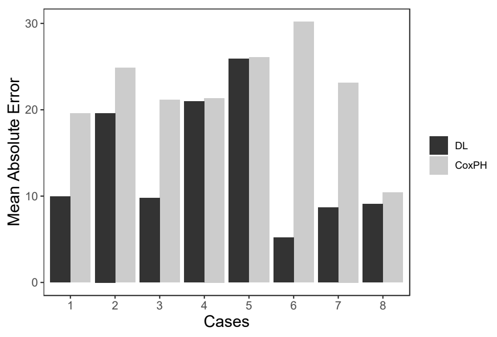
2.4 Case Study Using Real World Data

In this section, we present a comparison of the proposed deep learning event prediction method against the traditional Cox PH model based method on a real-world dataset collected from industrial equipment. Data is available from different units of this industrial equipment. These units are equipped with multiple sensors which transmit signals pertaining to warning or failure of the system or any of its sub-systems. A summary of data is presented in Table 2.5. Please note that the actual calendar times are adjusted for each unit, i.e., the starting time is made zero for all the units. To protect confidentiality, descriptions of event-types is not provided, and a small random noise has been added to the event-times. The noise does not affect the results or interpretations of our analysis.

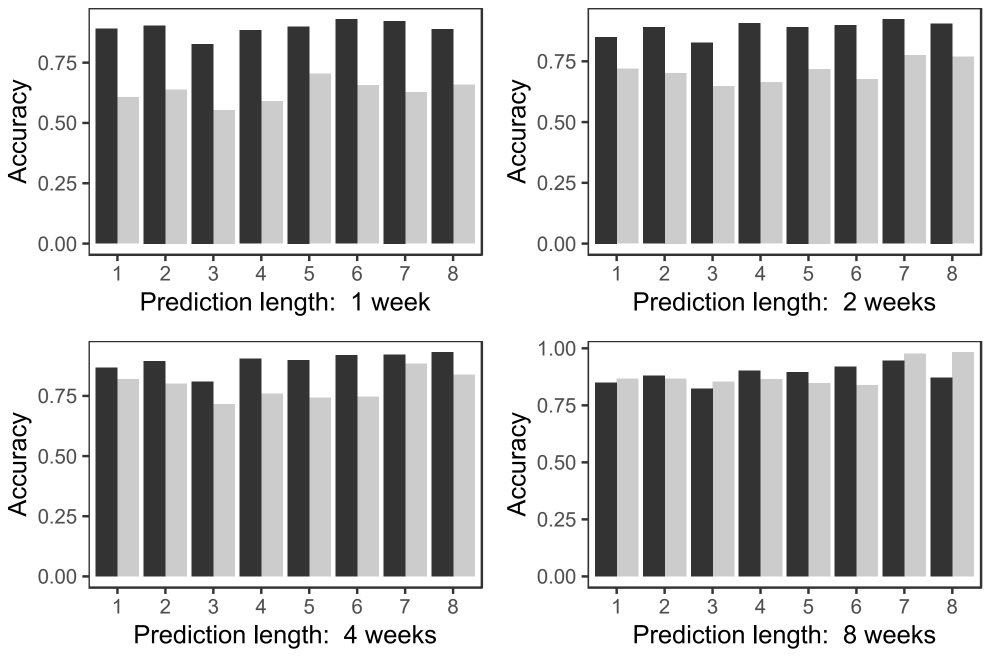
Table 2.5: Summary of The Real World Dataset

Event Duration	**2 years**
Total # of Units	**669**
Total Event Types	**198**
Total Events	**4,976,512**
Maximum Total Events in A Sequence	**99,312**

For the real-world data analysis, we perform 13 cases using a total of 87 important event-types selected from the whole dataset. We augment the data with a sliding window of size 10. Sampling using sliding window may lead to correlation among samples, which may affect the training results. However, knowing deep neural networks always need large volumes of data to train the model, the windowed sampling method, in a way, ensures that the model training could have sufficient data samples when there is no sufficient independent training samples. This strategy has been commonly used in deep neural network training. For example, artificial images from the transformation of a real image has been used as a data augment method in the training of a CNN (Convolution Neural Network) [105]. Similar to the simulation in Section 2.3, 20% of the data is kept aside to be test data, and 20% of the rest data is set to be the validation data in deep learning model. These event-types represent error or warning information arising from different components in the equipment. The sensors onboard collect and transmit information when an event occurs. We employ the Cox PH method and Bidirectional-GRU on the dataset for different units of the equipment. Figure 2.15 presents the comparison of accuracy and MAE for the two approaches. For the accuracy prediction, different lengths of prediction interval are presented.



(a) Mean Absolute Error



(b) Accuracy under Different Prediction Length

Figure 2.15: Prediction results for real world data

The results obtained indicate that the deep learning based approach outperforms the classical Cox PH method in most cases, particularly when the prediction interval is relatively short (e.g., 1, 2 and 3 week). To apply in practice, several variants of the deep learning approach should be tested and applied.

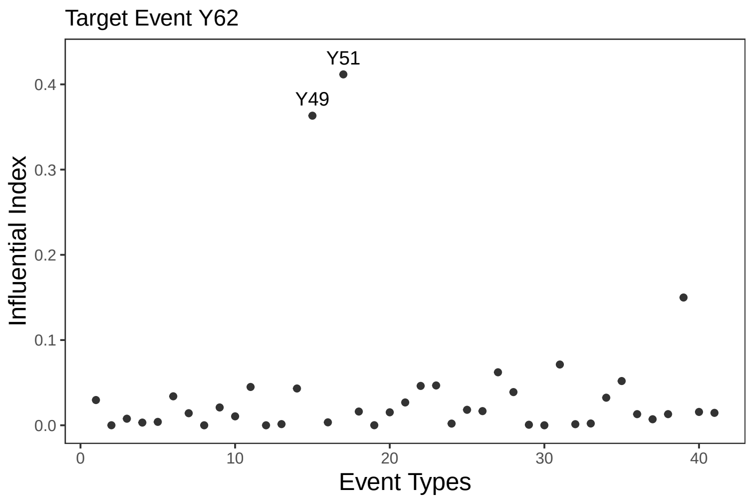


Figure 2.16: An Example of The Influential Event Results for Real World Dataset

We conduct the influential event identification to the selected event sequences, and a typical case of the results are presented in Figure 2.16. We can actually interpret the relationship between the target event and the identified key trigger events based on engineering knowledge of the system. For example, we identify that for the target event-type [Y62: *The BSM power supply error*], as shown in Figure 2.16, [Y51: *Incorrect Lift Power Amplifier Installed*] and [Y49: *Incorrect TPA Module Installed*] are detected as the most influential trigger events. Apparently, the missing power event may have been caused by an incorrect power supply module installation. With the interpretation of our trained model, we could easily detect similar relationships and set up potential corrective action. This further confirms the usefulness of our proposed model interpretation method.

2.5 Conclusion

In this paper, we introduce a model for predicting critical events and interpreting the most influential trigger events using event logs. One-hot encoding, window techniques are utilized to transform and augment the original datasets. Moreover, inverse probability of censored data weights are used to preprocess the data, which allocate weights to uncensored data samples based on the censored information. When training the model, different neuron structures including bidirectional layer and stacked layer are considered to find the best fit model. After the model is well-trained, an event interpretation algorithm is proposed and the Influence Index is defined to find the most influential trigger events. From both simulated datasets and real-world data, our model outperforms the traditional Cox PH model in mean absolute prediction error and also the prediction accuracy in most of the cases. The interpretation results in the real world data further shows the potential for future development of an automatic root cause identification model.

The main contributions of the work are in three folds: (1) We established a detailed procedure of applying deep learning to critical event prediction using event logs, including processing and encoding event logs to the form that fits deep learning tasks, comparing various deep learning structures to identify the most effective one for event prediction, and comparing with conventional statistical methods to provide insights to the event prediction problem. (2) We adapted IPCW method to deep learning so that we can handle the censored observations for event prediction purposes. Censoring is common in event data and the adapted IPCW method can make the censored data fully utilized in the deep learning framework. (3) We proposed an interpretation technique to overcome the weak interpretability of deep learning method. We can identify the important events that are associated with the critical event to be predicted. Deep learning, as a powerful machining learning technique, will find a broad application in event log modeling and analysis. Future works include tuning algorithm to rigorously decide the best hyper-parameters for a proposed deep learning model. Also, the proposed interpretation algorithm in this work can identify the association between a single event with the critical event. This work can be extended to identify a combination of multiple events that are associated with the critical event. We will investigate these problems and report our findings in the future.

2.6 Appendix

2.6.1 Baseline and Censoring Parameters Used

Event	Setting I								Setting II							
	Case I1		Case I2		Case I3		Case I4		Case III1		Case II2		Case II3		Case II4	
	shape	scale	shape	scale	shape	scale	shape	scale	shape	scale	shape	scale	shape	scale	shape	scale
1	1	2	1	2	2.5	10	2.5	10	1	1	1	2	2.5	10	2.5	10
2	1	2	1	2	2	10	2	10	1	2	1	2	2	10	2	10
3	1	2	1	2	2.1	10	2.1	10	1	3	1	2	2.1	10	2.1	10
4	1	2	1	2	2.3	10	2.3	10	1	4	1	2	2.3	10	2.3	10
5	1	2	1	2	2.2	10	2.2	10	1	5	1	2	2.2	10	2.2	10
6	1	2	1	2	2.1	10	2.1	10	1	6	1	2	2.1	10	2.1	10
7	1	2	1	2	2.5	10	2.5	10	1	7	1	2	2.5	10	2.5	10
8	1	2	1	2	2.5	10	2.5	10	1	8	1	2	2.5	10	2.5	10
9	1	2	1	2	2.5	10	2.5	10	1	9	1	2	2.5	10	2.5	10
10	1	2	1	2	2.1	10	2.1	10	1	10	1	2	2.1	10	2.1	10
11	1	2	1	2	2.2	10	2.2	10	2	1	1	2	2.2	10	2.2	10
12	1	2	1	2	2.3	10	2.3	10	2	2	1	2	2.3	10	2.3	10
13	1	2	1	2	2.4	10	2.4	10	2	3	1	2	2.4	10	2.4	10
14	1	2	1	2	2.6	10	2.6	10	2	4	1	2	2.6	10	2.6	10
15	1	2	1	2	2.9	10	2.9	10	2	5	1	2	2.9	10	2.9	10
16	1	2	1	2	2.1	10	2.1	10	2	6	1	2	2.1	10	2.1	10
17	1	2	1	2	2.1	10	2.1	10	2	7	1	2	2.1	10	2.1	10
18	1	2	1	2	2.3	10	2.3	10	2	8	1	2	2.3	10	2.3	10
19	1	2	1	2	2.2	10	2.2	10	2	9	1	2	2.2	10	2.2	10
Cens distb	10	10	10	10	33	23	33	23	33	33	10	10	10	22	33	22

Table 2.6: Baseline and Censoring Parameters

2.6.2 Different Parameters of Censored Dataset

Event	Original Data 1		Original Data 2		Original Data 3		Original Data 4	
	shape	scale	shape	scale	shape	scale	shape	scale
1	2.5	3	2.5	4	2.5	1	2	2
2	2	3	2	4	2	1	2	2
3	2.1	1	2.1	1	2.1	1	2	2
4	2.3	1	2.3	1	2.3	1	2	2
5	2.2	1	2.2	1	2.2	1	2.2	1
6	2.1	1	2.1	1	2.1	1	2.1	1
7	2.5	1	2.5	1	2.5	1	2.5	1
8	2.5	1	2.5	1	2.5	1	2.5	1
9	2.5	1	2.5	1	2.5	1	2.5	1
10	2.1	1	2.1	1	2.1	1	2.1	1
11	2.2	1	2.2	1	2.2	1	2.2	1
12	2.3	1	2.3	1	2.3	1	2.3	1
13	2.4	1	2.4	1	2.4	1	2.4	1
14	2.6	1	2.6	1	2.6	1	2.6	1
15	2.9	1	2.9	1	2.9	1	2.9	1
16	2.1	1	2.1	1	2.1	1	2.1	1
17	2.1	1	2.1	1	2.1	1	2.1	1
18	2.3	1	2.3	1	2.3	1	2.3	1
19	2.2	1	2.2	1	2.2	1	2.2	1
Censored Rate	0.33	0.22	0.55	0.45	0.4	0.33	0.21	0.12
Case No.	1	2	3	4	5	6	7	8

2.6.3 Review of Cox PH Model Based Event Prediction

If we treat the critical event we want to predict as the failure event and other events as time-varying covariates, we can apply Cox PH model to predict the occurrence of the critical event. A Cox PH model is a semi-parametric model which assumes that the hazard function is comprised of two components – (i) a non-parametric baseline, $h_0(t)$, and (ii) external covariates (or features), X which are linked through a link function g. In common practice, the exponential function is used as the link function. The Cox PH model is as follows:

$$h(t|X) = h_0(t)\exp(\boldsymbol{\beta}X) \tag{2.15}$$

where, $\boldsymbol{\beta}$ is the vector of coefficients associated with the covariates. For the event prediction at hand, we can incorporate preceding events as time-varying covariates in the Cox PH model as

$$h(t|\mathbf{Z}(t)) = h_0(t)\exp(\boldsymbol{\beta}\mathbf{Z}(t)) \tag{2.16}$$

where $\mathbf{Z}(t)$ is the vector of time-varying predictors corresponding to all the event-types. A predictor event-type, say A, is encoded as follows

$$Z_A(t) = \begin{cases} 0, & 0 < t < T_A \\ 1, & t > T_A \end{cases} \tag{2.17}$$

where T_A is occurrence time instance of event A.

Through the Cox PH model, we can obtain the hazard function $h(t)$ and then we can obtain the survival function as $S(t) = \exp\Big(-\int_0^t h(s)ds\Big)$. The prediction of failure time at time t^* can be computed using the survival function. [72] Knowing that the unit has not failed until t^*, we can compute and integrate the conditional survival function as follows to get the expected time to failure

$$\mathbf{E}(t|t > t^*) = \int_{t^*}^{\infty} S(s|t^*)ds = \frac{\int_{t^*}^{\infty} S(s)ds}{S(t^*)} \tag{2.18}$$

Please check the regarding detailed parameter estimation for Cox PH model. [72] Also, more details on Cox PH model based event prediction can be found.[79]

Chapter 3

Mixed-input Bayesian Optimization Method for Structural Damage Diagnosis

Abstract

Structural Health Monitoring (SHM) is of significant importance in the operation of engineering systems to ensure the durability and reliability. In this paper, we introduce a Bayesian Optimization method using a multi-output Gaussian process to solve the structural fault diagnosis problem. This method utilizes a high fidelity finite element model (FE) of the structure and the impedance/admittance measurements from the structure to identify the location and severity of the damage. The method improves the accuracy of the damage diagnosis by adopting a multi-output Gaussian process as the surrogate model for the full FE model and Thompson sampling approach is used to guide the search for the structural damage in the Bayesian optimization. The detailed algorithms are presented, and convergence analysis of the method is conducted. We apply our proposed method on simulated synthetic functions and it achieves better performance and higher convergence speed than traditional mixed input optimization methods. We then apply our method on a real world structural damage identification problem using measured piezoelectric admittance data and

This chapter is based on the paper: **Huang, C.**, Lee, J., Zhang, Y., Zhou, S. & Tang, J. (2022). Mixed-input Bayesian Optimization Method for Structural Damage Diagnosis. IEEE Transactions on Reliability, vol. 72, no. 2, pp. 678-691.

illustrate the effectiveness of the proposed method.

3.1 Introduction

Structural health monitoring (SHM) is of vital importance in ensuring the durability and reliability of engineering systems. Structural damage in the systems can cause performance degradation and even lead to catastrophic consequences [48, 50, 53, 84]. Early detection and identification of the location and severity of structural damage is the key for mitigation such risk. To enhance the reliability of the systems, a number of structural damage diagnosis techniques have been developed in recent years. Structural health monitoring is mostly facilitated through measuring and comparing the dynamic responses of structures [17, 67, 75]. Methods of SHM utilize vibration measurements from which the natural frequencies and mode shapes are extracted to infer damage occurrence. These methods are easy to implement, as they employ off-the-shelf sensors and the vibration responses can be modeled in a straightforward manner. However, typically only the lower order natural frequencies and mode shapes can be extracted experimentally [18, 68], because it is generally hard and even impossible to excite and measure high-frequency vibration responses using simple experimental setup. As such, the wave lengths involved is large, leading to relatively low sensitivity in damage detection and identification.

Alternatively, a different class of methods utilize the wave propagation information. As waves pass through damage sites, their propagation pattern may exhibit changes [91]. Using transducers such as piezoelectric actuators/sensors with high bandwidth, high frequency waves can be excited and senses, leading to high detection sensitivity. Nevertheless, the interaction between transient wave and local damage which may have arbitrary profile could be extremely complicated. The identification of damage severity using transient wave propagation becomes difficult [128].

In recent years, a new class of structural damage detection methods, called piezoelectric impedance or admittance based methods [93, 118], has been suggested. The approach works in such a way that a piezoelectric transducer is bonded locally to the host structure to be monitored. The transducer has a two-way electro-mechanical coupling effect and can be used as both the actuator and sensor simultaneously. When a harmonic voltage excitation is applied, the piezoelectric transducer can excite the structure and measure the electro-mechanical signatures. When there is damage in the structure, it will result in impedance change around the structural resonances. These impedance changes can be used to monitor structural conditions. These methods preserve the high-bandwidth nature of piezoelectric transducers and thus lead to high detection sensitivity. When a high-fidelity finite element (FE) model in healthy state is available, the damage identification can be carried out to locate and quantify the damage inversely with the electro-mechanical impedance changes as input. For example, Shuai *et al.*[118] applied the FE modeling, and formulated a linearization of the impedance response through sensitivity matrix computation to represent the relationship of the impedance changes and possible damage. In practice, however, the number of unknowns to be identified, i.e., the location and severity of damage, is large while the data points of impedance response measurements are limited. In another words, the sensitivity matrix is rank-deficient [68]. If the conventional least squares method is adopted to solve the under-determined problem, it may yield untrue solutions.

To address this limitation, the under-determined identification problem can be formulated as a global optimization problem [104, 116], aiming at minimizing the difference of admittance change between experimental measurements and model prediction. Since the sensitivity matrix is rank-deficient, the under-determined problem has many or infinite solutions that may not be the true damage location and severity. To address this issue, some sparsity constraints are added to the model to remove the undesired solutions [59, 138]. In [18], a multi-objective DIRECT [62] algorithm to find the damage location and severity that is closest to the observed admittance under the assumption of sparsity is proposed.

Nevertheless, there still exists a research gap. The model remains the linear formulation, which is accurate only when the damage severity is small enough. In addition, even with sparsity constraint, these methods still provides multiple solutions and requires manually select the most plausible solution from them. A better algorithm is expected to be developed to find a more accurate solution of the problem and provide deterministic solution.

In this work, we propose a mixed-input global optimization approach for structure damage diagnosis. In this approach, we treat the structure FE model as a black-box function and then we search the input to the function (i.e., damage location and severity) that fits the obtained observations the best. Using the full FE model, we can avoid the accuracy loss in the linearization step. Black-box optimization has been studied extensively. Bayesian optimization (BO) is a popular black-box function optimization approach [94, 117, 120, 121], particularly for functions that are expensive to evaluate. It has been applied to a number of scientific domains in recent years [16, 76]. BO uses statistical surrogate model fitted to the data and utilizes the surrogate model to find the next point to evaluate, so that the optimization process can efficiently converge to the optimal solution. Benefiting from the properties inherited from the normal distribution, Gaussian process (GP) has been widely applied in data modeling, simulation optimization, and prediction problems [61, 69, 137]. The majority of applications of BO also utilizes GP as a surrogate model to be fitted to the data. It has been shown that BO that uses GP as the surrogate model will converge to the global optimal under some non-restrictive constraints [15].

Most of the existing BO methods and applications focus on the cases where there are only continuous variables in the input domain of the function [11, 117, 120, 121]. However, for the structure damage diagnosis problem at hand, the input variables are of mixed types: the location of the damage is a discrete variable while the severity of the damage is a continuous variable. Optimization with both discrete and continuous inputs are interesting topics in the optimization field. Unlike the continuous input variables, discrete variables contain the information that is not easy to be explored. Recently, Nguyen et al. used multi-armed bandit model to frame the BO problems (MAB-BO) with discrete and continuous input variables [99].

Although MAB-BO is able to handle both discrete and continuous variables, it does not consider the correlation of the responses corresponding to different discrete inputs in each optimization iteration. In other words, the responses corresponding to different discrete inputs are modeled independently by separate GP models in MAB-BO method [99]. However, in many real world cases, the responses corresponding to different discrete inputs often show similar patterns. Take the structure damage diagnosis problem as an example. Intuitively, damages occurring on close locations may show very similar patterns in the structure dynamic response and the information from adjacent locations may contribute to the prediction for each other. Thus, if we have a model to simultaneously describe multiple responses corresponding to multiple potential damage locations, then we will have more accurate model to describe the underlying function, which will lead to better performance of the BO algorithm.

In this work, we adopt a recently developed multi-output Gaussian process (MGP) model with non-separable covariance functions as the surrogate model in the BO algorithm. Similar to MAB-BO in [99], a Thompson sampling approach is used to guide the sequential sample of the underlying function, which can provide a good trade-off between exploitation and exploration of the search. A rigorous convergence analysis is conducted to show the converging property of the proposed method. Numerical studies based on both simulated data and real world structure damage diagnostics problem compare the proposed method and several other methods and demonstrate the effectiveness of our method.

The main contributions of this work can be summarized into three folds: (i) Apply BO, a black-box function optimization method, to the structure damage diagnosis problem, which can directly work on the full FE model without linearization. (ii) Extend the existing BO method by adopting a flexible non-separable MGP model as the statistical surrogate model in the BO framework. And (iii) Rigorous convergence results are obtained for the proposed BO method.

The rest of the paper is organized as follows. In Section 3.2, we introduce the related background of Bayesian Optimization (BO), including a brief introduction of BO and the Gaussian process used in BO. In Section 3.3, the Multi-armed bandit Multi-output Gaussian process Bayesian Optimization (MAB-MGP-BO) is presented and the convergence of the method is analyzed. In Section 3.4, numerical simulation on two synthetic functions is conducted. The results show that our method can achieve a higher optimal value and a faster convergence than several other methods including MAB-BO method in [99]. In Section 3.5, case study of structural damage identification based on both simulated data and real world data are conducted. It shows that our method can successfully detect the location and severity of the damage. Finally, we conclude our work and have a discussion on future work in Section 3.6.

3.2 Review of Bayesian Optimization

In this section, we give a brief introduction of related background knowledge on the Bayesian optimization (BO) we will utilize in the proposed method. Generally, BO methods consist of two parts. First, a machine learning method is utilized to build a surrogate model of the input and the objective function. Then, an acquisition function is designed to decide where to take the next observation (also called "evaluation") from the objective function. Repeating the two steps sequentially, we can build a surrogate model close to the underlying black-box function and find a solution near the global optimal point. Figure 3.1 is an illustration of BO method using GP as the surrogate model and upper confidence bound [122] of prediction as the acquisition function under noiseless condition, where t is the number of iterations, i.e. the number of evaluations we made on the underlying black-box function f. The x-axis is the input of the function and the y-axis is the value of function f. The values of the acquisition function, the upper confidence bounds with one standard deviation, are shown in dash-dotted line on the bottom of the box at each t. The true underlying functions are presented in solid lines above them. The observations we already have are presented in by round points and the square point is the new observation point we made in the current iteration, which is chosen based on the acquisition function of the former iteration. The dotted and short dotted line are the posterior predictions and the confidence intervals. As we can see from the figure, the prediction is closer to the underlying function and the uncertainty is lower through the iterations.

Formally, let $f(\mathbf{x})$ be the black-box function with global optimizer $\mathbf{x}^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, where $\mathbf{x}$ is the vector of variables and $\mathcal{X}$ is the domain of $\mathbf{x}$. Assume that till the t-th iteration, the set of observation points $D_t = \{(\mathbf{x}_i, y_i)|i = 1, \dots, t\}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ If we fit the surrogate model to the data in each iteration and we obtain the prediction of $f(\mathbf{x})$ with mean $\mu_t(\mathbf{x})$ and variance $\sigma_t^2(\mathbf{x})$. Taking the upper confidence bound $U_t(\mathbf{x}) = \mu_t(\mathbf{x}) + \beta\sigma_t(\mathbf{x})$, (In Figure 3.1, $\beta = 1$) as the acquisition function, we take $\mathbf{x}_{t+1} = \text{argmax}_{\mathbf{x}\in\mathcal{X}} U_t(\mathbf{x})$ as the next evaluation point. Then, we evaluate $\mathbf{x}_{t+1}$ with the underlying model and get the response y_{t+1}. Finally, the tth iteration is finished and the dataset of the $(t+1)$th iteration is $D^{t+1} = D^t \cup \{(\mathbf{x}_{t+1}, y_{t+1})\}$. After a large enough number

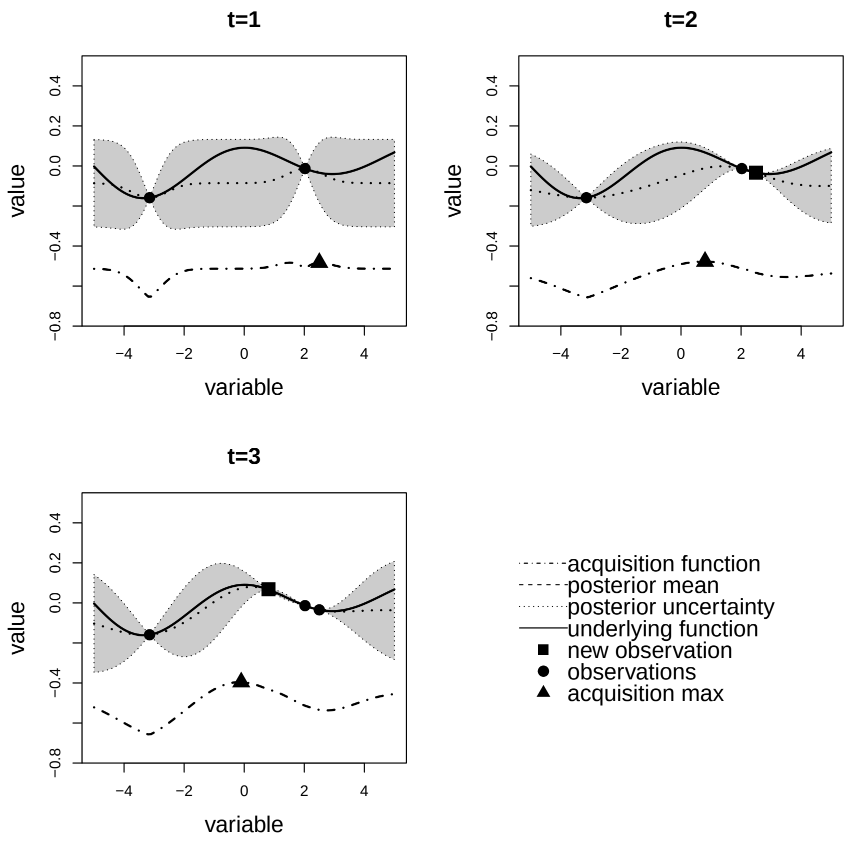


Figure 3.1: An example of Bayesian optimization method

of iterations, we can achieve the global extreme point x^*.

As we mentioned above, the most frequently used surrogate model in BO is the GP model. A GP is a generalization of the Gaussian probability distribution and can be considered as a Gaussian distribution prior over functional data. It has been well established that GP model is a very flexible and expressive model that can be used to describe a wide range of functions [141, 143]. It is a stochastic process, which is a collection of random variables and any finite subcollection of which follows a multivariate normal distribution. Thus, the distribution of a GP is the joint distribution of all those (infinitely many) random variables.

A GP $f(\mathbf{x})$ is specified by its mean function $\mu(\mathbf{x})$, $\mu : \mathcal{X} \to \mathbb{R}$ and its covariance function $c(\mathbf{x}, \mathbf{x}')$, $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $c(\mathbf{x}_i, \mathbf{x}_j)$ is the covariance between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$. Please note that without causing confusion, we use the same symbol f to represent the underlying function and a GP. We denote the GP as $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), c(\mathbf{x}, \mathbf{x}'))$. Specifically, for a set of inputs $\mathbf{X}_n = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$, the vector of output $f(\mathbf{X}_n)$ is Gaussian distributed with mean $\mu(\mathbf{X}_n) = [\mu(\mathbf{x}_1), \mu(\mathbf{x}_2), \ldots, \mu(\mathbf{x}_n)]^T$ and $n \times n$ covariance matrix $C(\mathbf{X}_n, \mathbf{X}_n)$:

$$C(\mathbf{X}_n, \mathbf{X}_n) = \begin{bmatrix} c(\mathbf{x}_1, \mathbf{x}_1) & c(\mathbf{x}_1, \mathbf{x}_2) & \cdots & c(\mathbf{x}_1, \mathbf{x}_n) \\ c(\mathbf{x}_2, \mathbf{x}_1) & c(\mathbf{x}_2, \mathbf{x}_2) & \cdots & c(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ c(\mathbf{x}_n, \mathbf{x}_1) & c(\mathbf{x}_n, \mathbf{x}_2) & \cdots & c(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{3.1}$$

A common covariance function is the squared exponential kernel, defined as $c(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp(-||\mathbf{x}_i - \mathbf{x}_j||_2^2)/(2l)^2$, where l is a length scale parameter and σ^2 is the parameter dictating the uncertainty in $f(\mathbf{x})$.

When we utilize a GP in BO as used in Figure 3.1, the posterior GP with a newly observed $\mathbf{y}$ at iteration t can be expressed as

$$\begin{aligned} &f(\mathbf{x})|D_t \sim \mathcal{N}(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x})) \\ &\quad \mu_t(\mathbf{x}) = C(\mathbf{x}, \mathbf{X}_t)[C(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I}]^{-1}\mathbf{y} \\ &\quad \sigma_t^2(\mathbf{x}) = C(\mathbf{x}, \mathbf{x}) - C(\mathbf{x}, \mathbf{X}_t)[C(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I}]^{-1} C(\mathbf{X}_t, \mathbf{x}). \end{aligned} \tag{3.2}$$

Another essential part of BO is the acquisition function, which determines how the search space are explored during the iterations. Acquisition functions are based on the posterior distributions of the GP fitted in each iteration. There is a trade-off between two directions of search: exploitation and exploration. Exploitation improves the region near the current best result, while exploration develops more unfamiliar regions that have higher uncertainties. Both directions are necessary to get to the optimal point in the long run of the algorithm. Otherwise the algorithm may stick in a local optimum or jump around with no sense. Most common acquisition functions includes upper confidence bound, expected improvement and Thompson sampling[36, 136]. We will provide more detailed discussion on these acquisition functions in the next section.

3.3 Multi-armed Bandit Bayesian Optimization with Multi-output Gaussian Process

In this section, we first introduce the detailed way to construct covariance function of Multi-output Gaussian process (MGP) in Section 3.3.1. Then, we introduce the proposed method MAB-MGP-BO (Multi-armed bandit Multi-output Gaussian Process Bayesian Optimization) in Section 3.3.2. Finally, the convergence analysis of the proposed method is conducted in Section 3.3.3.

3.3.1 Convolved Process and Multi-output Gaussian Process

As we mentioned before, in many cases, not only the observations under the same categorical input value but also the observations from different categorical input values contribute information to the fitting of the model. A model that can consider both the information within the same category and the information across different categories is preferred. In Figure 3.2, we show three functions that correspond to the objective function under three different categorical input values. In the Multi-armed Bandit formulation, the objective function under a categorical input value is also regarded as an arm.

For an MGP model, we should not only specify the covariance function for a single output (as that in the conventional univariate output GP model), but also the cross covariance among different outputs. Thus, the covariance function of MGP is in the form of $c(i, j, \mathbf{x}, \mathbf{x}')$, where $c(i, j, \mathbf{x}, \mathbf{x}') = \mathrm{cov}(f_i(\mathbf{x}), f_j(\mathbf{x}'))$ and i, j are output indices. The parameterization of the covariance function $c(i, j, \mathbf{x}, \mathbf{x}')$ plays a critical role in MGP model because it characterizes the relationship between any two outputs. A popular approach is to use a separable covariance structure, i.e., letting $c(i, j, \mathbf{x}, \mathbf{x}') = \tau(i, j) \times \mathrm{cov}(\mathbf{x}, \mathbf{x}')$ [24, 107, 147]. Other formats combining separable covariance functions are also designed in recent research [101, 111], but they still calculate the covariance of continuous and categorical variables seperately. In the current state-of-the-art open source implementation of GP model such as GPyTorch [38], the separable covariance function is used. The separable structure is appealing due to the simplified covariance structure, however it restricts all outputs to share the same set

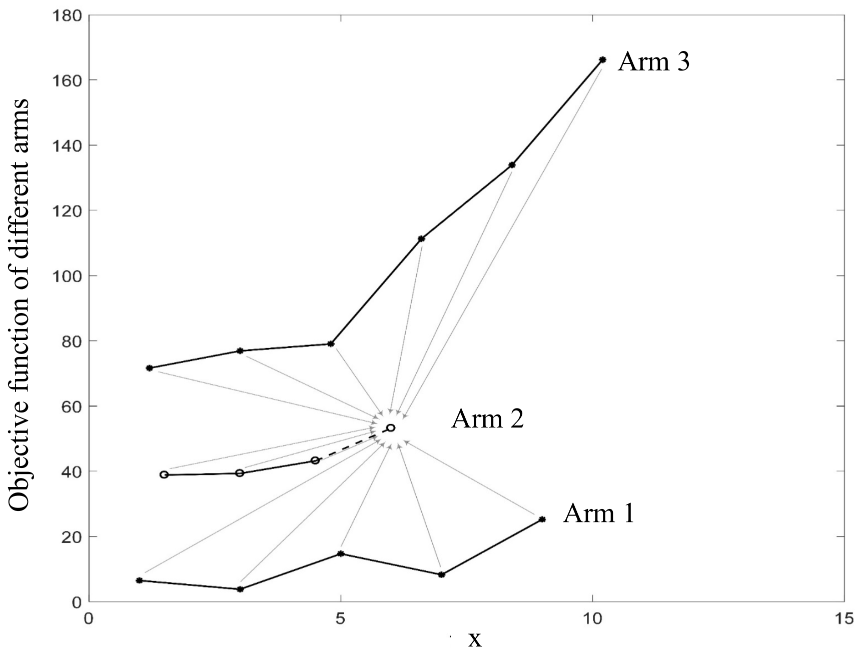


Figure 3.2: Illustration of MGP. Three functions are shown, which correspond to three different categorical input values.

of covariance parameters, i.e., the part $\mathrm{cov}(\mathbf{x}, \mathbf{x}')$ is the same for all the outputs. However, different outputs may have different characteristics and thus it is too restrictive to use the same covariance function for the continuous variables across different outputs.

To overcome the limitation of separable covariance function, we adopt a nonseparable covariance structure that are based on convolution processes (CP) [37, 86, 90]. The CP-based nonseparable covariance function is based on that a GP can be constructed by convolving a latent Gaussian white noise process with a smoothing kernel [133]. In more details, we can construct a GP $f(\mathbf{x})$ by convolving a Gaussian white noise process $W(\mathbf{x})$ with a smoothing kernel $k(\mathbf{x}) = \exp(-\mathbf{x}^2/(2l^2))$, where $\mathrm{cov}(W(\mathbf{x}), W(\mathbf{x}')) = \delta(x - x')$, l is the length scale parameter. δ is the Dirac delta function, defined in [7] as :

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \quad , \delta(x) = \begin{cases} 0, & x \neq 0 \\ \infty, & x = 0 \end{cases} \tag{3.3}$$

such that

$$f(\mathbf{x}) = \int_{-\infty}^{\infty} k(\mathbf{x} - \mathbf{u})W(\mathbf{u})d\mathbf{u}. \tag{3.4}$$

Then, the corresponding covariance function can be expressed as:

$$\mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \int_{-\infty}^{\infty} k(\mathbf{x} - \mathbf{u}) k(\mathbf{x}' - \mathbf{u}) d\mathbf{u}. \tag{3.5}$$

Thus, the GP is parameterized by the parameters in the smoothing kernel k, which is required to be square or absolutely integrable, i.e. $\int_{-\infty}^{\infty} |k(\mathbf{x})|^2 d\mathbf{x} < \infty$.

For a MGP setting, if each output is constructed in this way, and if we share these latent Gaussian white noise process across multiple outputs, then multiple outputs can be expressed as a jointly distributed GP [6, 13]. For example, a nonseparable MGP construction is shown in Figure 3.3. This construction is first proposed by [74] and is adopted in this work. In Figure 3.3, we try to construct a surrogate MGP model for the function corresponding to the rth arm, denoted as f_r, while f_i, $i \in \{1, ..., m\} \backslash r$ correspond to the functions from other arms. Then, according to this construction, $f_r(\mathbf{x})$ can be written as:

$$f_r(\mathbf{x}) = k_{r,r}(\mathbf{x}) \star W_r(\mathbf{x}) + \sum_{i=1}^{m} k_{i,r}(\mathbf{x}) \star W_i(\mathbf{x}), \tag{3.6}$$

where $k_{i,j}(\mathbf{x}) \star W_i(\mathbf{x}) = \int_{-\infty}^{\infty} k_{i,j}(\mathbf{x} - \mathbf{u}) W_i(\mathbf{u}) d\mathbf{u}$ and $k_{i,j}$ is the kernel used in the convolution of the ith latent function and the jth output, $i, j \in \{1, 2, \ldots, m\}$. For the functions from other arms, it is constructed as $f_i(\mathbf{x}) = k_{i,i}(\mathbf{x}) \star W_i(\mathbf{x})$. Clearly, W_i is shared across f_i and f_r in the construction and thus, the cross correlation between f_i and f_r can be modeled.

With this construction, we can obtain the covariance matrix of the MGP model for f_r as [74]

$$C_{N \times N} = \begin{bmatrix} C^{1,1}_{n_1 \times n_1} & \mathbf{0}_{n_1 \times n_2} & \cdots & \mathbf{0}_{n_1 \times n_m} & C^{1,r}_{n_1 \times n_1} \\ \mathbf{0}_{n_2 \times n_1} & C^{2,2}_{n_2 \times n_2} & \cdots & \mathbf{0}_{n_2 \times n_m} & C^{2,r}_{n_2 \times n_r} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{n_m \times n_1} & \mathbf{0}_{n_m \times n_2} & \cdots & C^{m,m}_{n_m \times n_m} & C^{m,r}_{n_m \times n_r} \\ C^{r,1}_{n_r \times n_1} & C^{r,2}_{n_r \times n_2} & \cdots & C^{r,m}_{n_r \times n_m} & C^{r,r}_{n_r \times n_r} \end{bmatrix}, \tag{3.7}$$

where n_i is the number of observed data points from function f_i, $N = \sum_{i=1}^{m} n_i$, $C^{i,i}_{n_i \times n_i}$ is the covariance within function f_i, $C^{r,i}_{n_r \times n_i}$ is the cross covariance between f_r and f_i, and $\mathbf{0}_{n_i \times n_j}$ is the zero matrix with dimension $n_i \times n_j$. Please note the covariance matrix in (3.7) is a

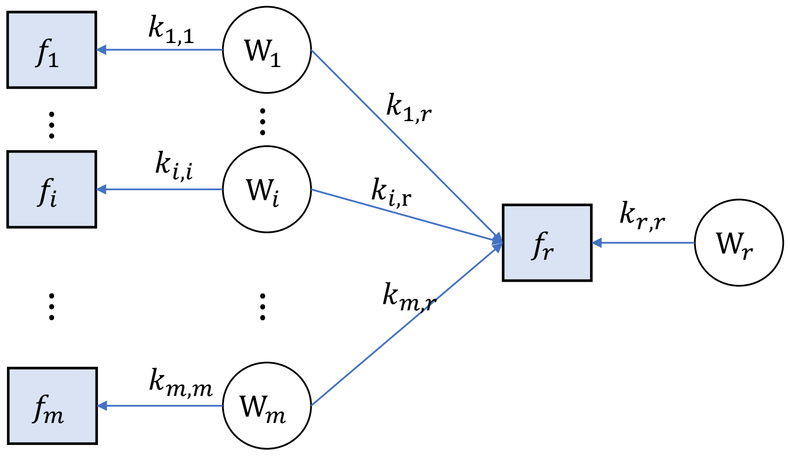


Figure 3.3: Illustration of the construction of MGP with shared white noise Gaussian Processes

function of all the parameters of kernel functions used in the construction, including $k_{i,i}$, $k_{i,r}$, and $k_{r,r}$, $i = 1, ..., m$. Once the covariance matrix is constructed, we can actually view the MGP as a conventional GP model and use the maximum likelihood estimation method to estimate the kernel function parameters based on the observed data from the functions of all the arms. With the fitted MGP model, we can also make probabilistic predictions of f_r at other input locations using the formula in (3.2). More technical details of the model construction and estimation can be found in [74].

One point we want to emphasize here is that the MGP model in Figure 3.3 is to model the function f_r only. If we want to model the function from a different arm, then we need to put that function in the place of f_r in Figure 3.3 and build a different MGP model. In other words, in our proposed MAB-MGP-BO method (details are shown in Section 3.3.2), for each arm, we have a different MGP surrogate model. It is possible to establish a complex CP based MGP model to model all the function simultaneously. However, such a model may involve a very large number of parameters and difficult to estimate and use. The arrangement in Figure 3.3 allows us to use multiple simpler models to describe the functions and each MGP is relatively easy to estimate.

3.3.2 MAB-MGP-BO Algorithm

Suppose we have a collection of black-box function $\{f_a(\mathbf{x}_a)\}_{a=1}^m$ with both categorical variable $a \in \{1, 2, \dots, m\}$ and continuous variable $\mathbf{x}_a \in \mathcal{X} \subset \mathbb{R}^d$. The goal of the algorithm is to find the optimum point $[a^*, \mathbf{x}^*_{a^*}] = \text{argmax}_{[a,\mathbf{x}]\in\{1,2,\dots,m\}\times\mathcal{X}_a} f_a(\mathbf{x})$, which can also be written as $a^* = \text{argmax}_{a\in\{1,2,\dots,m\}} f(\mathbf{x}^*_a)$ and $\mathbf{x}^*_a = \text{argmax}_{\mathbf{x}\in\mathcal{X}_a} f(\mathbf{x}_a)$.

To solve the problem, we can formulate it into a Multi-armed bandit(MAB) problem [14]. For each arm a, we use BO to find the optimal point $\mathbf{x}^*_a = \text{argmax}_{\mathbf{x}\in\mathcal{X}_a} f(\mathbf{x}_a)$. Then, to find the optimal arm $a^* = \text{argmax}_{a\in\{1,2,\dots,m\}} f(\mathbf{x}^*_a)$. In the proposed MAB-MGP-BO, we use MGP model in each arm as the surrogate model for the black-box function corresponding to the arm. The MGP model can borrow information from the black-box functions of other arms, which could lead to more accurate surrogate model and in turn lead to better optimization performance. The detailed algorithm is presented in Algorithm 4.

Algorithm 4 MAB-MGP-BO

```
Input: $m$: number of arms/outputs, $t$: number of iterations
Output: $\mathbf{x}^*$: the optimal solution of $f$ and $f^* = f(\mathbf{x}^*)$
 1: for $t = 1, 2, \dots$ do
 2:     for $a = 1, 2, \dots, m$ do
 3:         Fit the Multi-output Gaussian process model to $D_t = \{(a_i, \mathbf{x}_i, y_i)|i = 1, \dots, t\}$
 4:         Draw a sample $\tilde{f}_a(\mathbf{x})$ from the fitted model $\hat{f}_a(\mathbf{x})$: $\tilde{f}_a(\mathbf{x}) \sim p(\hat{f}_a(\mathbf{x})|D_t)$
 5:         Find $\tilde{\mathbf{x}}^*_a = \text{argmax}_{\mathbf{x}\in\mathcal{X}_a} \tilde{f}_a(\mathbf{x})$
 6:         Let $\tilde{f}^*_a = \tilde{f}_a(\tilde{\mathbf{x}}^*_a)$
 7:     end for
 8:     $a_{t+1} = \text{argmax}_{1\le a\le m} \tilde{f}^*_a$
 9:     $\mathbf{x}_{t+1} = \tilde{\mathbf{x}}^*_{a_{t+1}}$
10:     Evaluate $y_{t+1} = f(\mathbf{x}_{t+1}) + \epsilon_{t+1}$
11:     if $y_{t+1} > f^*$ then
12:         $f^* = y_{t+1}, \mathbf{x}^* = \mathbf{x}_{t+1}$
13:     end if
14:     $D_{t+1} = \{(a_{t+1}, \mathbf{x}_{t+1}, y_{t+1})\} \bigcup D_t$
15: end for
```

Take m as the number of arms (also called number of outputs) and t as the number of iterations or evaluations.In each iteration t, we fit the MGP to D_t, the total data set at iteration t. The fitted model is denoted by $\hat{f}_a(\mathbf{x})$. Then, we draw a sample function $\tilde{f}_a(\mathbf{x})$ from the posterior Gaussian distribution of the MGP: $\tilde{f}_a(\mathbf{x}) \sim p(\hat{f}_a(\mathbf{x})|D_t)$. We locate $\tilde{\mathbf{x}}^*_a$, the optimal point at arm a, by maximizing the sampled function $\tilde{f}$. The value of $\tilde{f}$ at $\tilde{\mathbf{x}}_a^*$ is denoted by $\tilde{f}_a^*$. After finding the optimal solution for the every arm, find the best arm and data point a_{t+1} and $\mathbf{x}_{t+1}$ for evaluation at the next iteration. Then, evaluate the underlying true function f and get the new observation value $y_{t+1} = f(\mathbf{x}_{t+1}) + \epsilon_{t+1}$. Finally, the data set is updated as D_{t+1} by adding the new observed data point $(a_{t+1}, \mathbf{x}_{t+1}, y_{t+1})$.

We would like to mention a couple of points regarding the above algorithm:

- Line 5 to Line 13 of Algorithm 1 is the Thompson sampling (TS) approach to determine the next input point to evaluate. As mentioned in the introduction section, there are many different method to sequentially select the next point to evaluate, including the popular methods such as the expected improvement (EI), the upper confidence bound (UCB), and TS. EI is a greedy improvement-based strategy that suggests the point that could improve the expectation of the function the most over the current evaluated point. Although EI is effective and provides reasonable performance in the practice, it is often too greedily focusing on the existing optimum point and collecting little information about the unfamiliar regions in the domain [113]. UCB, as we mentioned earlier, chooses the point that maximum the upper confidence bound $U_t(\mathbf{x}) = \mu_t(\mathbf{x}) + \beta\sigma_t(\mathbf{x})$ of the posterior prediction of the underlying function for the next iteration. This algorithm can balance the exploration and exploitation of the optimization. However, the performance of the optimization may depend on the confidence parameter β and it holds a lower regret bounds than the TS method as an acquisition function of BO [112]. TS proposed by [127] is a sequential sampling heuristic that can handle the exploration-exploitation dilemma. Instead of setting a closed form expression, this method samples a function from the fitted Gaussian process and suggests the next iteration point by the maximum the value of the sampled function. TS has desirable theoretical properties [112] with a pretty tight regret bound and provides basis to the convergence analysis for MAB-MGP-BO as discussed in Section 3.3.3.

- On Line 8 of Algorithm 1, we need to find the maximum value from a sample (i.e., a realization) of GP from the fitted MGP model. A straightforward grid search is often used to solve the optimization problem on Line 8: we just treat the MGP as a multivariate Gaussian distribution and sample it at a large number of regularly distributed input points (also called grid points) and then find the largest value among the sampled values. This method is effective and viable only when the dimension of input is low. If the dimension of input is high, the number of grids points to fill the input space will be overwhelming. For example, for a problem with 10 dimensional input and 1000 grid points for each dimension, we will have 10^{30} grid points. It is infeasible to sample such a large number of points simultaneously. Instead, we need to adopt a sequential sampling strategy when we solve the problem in Line 8: We just sample one or a small batch of points from the MGP at an iteration and let a searching algorithm (such as a gradient descent searching method) to determine what next points to sample for the next iteration One critical point for the sequential sampling method is that the samples from different iterations should not be *independent* samples from the MGP. Rather, *the later samples depend on the previous samples so that the samples obtained from the sequential procedure should be the same as if they are sampled simultaneously.* We have established a sequential sampling method as shown in Algorithm 2.

The sequential sampling algorithm from MGP is shown in Algorithm 5.

Algorithm 5 Sequential Sampling from MGP

```
Initialize: Global $D_t^s = \emptyset$: the set of sampled points from the fitted MGP, denoted by $\hat{f}_a(\mathbf{x})$.
1: function SAMP($\mathbf{x}$)
2:     Calculate the posterior Gaussian distribution of given both the evaluated and sampled data $\hat{f}_t^a | D_t, D_t^s$.
3:     Sample a new point from the posterior distribution $\tilde{f} \sim p(\hat{f}_a(\mathbf{x}) | D_t, D_t^s)$
4:     Save the new sampled point: $D_t^s = D_t^s \bigcup \{(\mathbf{x}, \tilde{f})\}$
5:     return $\tilde{f}$
6: end function
7: Optimize the sampling function: $\tilde{\mathbf{x}}_a^* = \text{argmax}_{\mathbf{x} \in \mathcal{X}_a} \text{SAMP}(\mathbf{x})$
```

At each iteration t and for each arm a of the Bayesian optimization model, we wrap the sampling process as a function of the new sample point $\mathbf{x}$ given the set of previously sampled points D_t^s. The sampled points D_t^s are regarded as evaluated points and the new sample value $\tilde{f}$ is based on the posterior distribution of not only the evaluated points D_t, but also the previously sampled points. Given the fitted MGP $\hat{f}_a(\mathbf{x})|D_t$ and a sampled data set D_t^s, we can use the following lemma to obtain the posterior MGP sample $\tilde{f} \sim \hat{f}_a(\mathbf{x})|D_t, D_t^s$. This result is used in Line 4 of Algorithm 2.

Lemma 1. *$D_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$ is set of the observed data points, where $y_i \sim f(\mathbf{x}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Suppose $D_t^s = \{(\mathbf{x}_j^s, \tilde{f}_j)\}_{j=1}^{n_s}$ is the set of sampled functions from the posterior distribution of the fitted MGP $\hat{f}(\mathbf{x})$ in the tth iteration. Let $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_t}]^T$, $\mathbf{X}_s = [\mathbf{x}_1^s, \ldots, \mathbf{x}_{n_s}^s]^T$, $\mathbf{y} = [y_1, \ldots, y_{n_t}, \tilde{f}_1, \ldots, \tilde{f}_{n_s}]^T$. Then, we have $f_a(\mathbf{x})|D_t, D_t^s \sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$, where*

$$\begin{aligned}
\boldsymbol{\mu}_s &= \begin{bmatrix} C(\mathbf{x}, \mathbf{X}_t) \\ C(\mathbf{x}, \mathbf{X}_s) \end{bmatrix} \begin{bmatrix} C(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I} & C(\mathbf{X}_t, \mathbf{X}_s) \\ C(\mathbf{X}_s, \mathbf{X}_t) & C(\mathbf{X}_s, \mathbf{X}_s) \end{bmatrix}^{-1} \mathbf{y} \\
\boldsymbol{\Sigma}_s &= c(\mathbf{x}, \mathbf{x}) - \begin{bmatrix} C(\mathbf{x}, \mathbf{X}_t) \\ C(\mathbf{x}, \mathbf{X}_s) \end{bmatrix} \\
&\begin{bmatrix} C(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I} & C(\mathbf{X}_t, \mathbf{X}_s) \\ C(\mathbf{X}_s, \mathbf{X}_t) & C(\mathbf{X}_s, \mathbf{X}_s) \end{bmatrix}^{-1} [C(\mathbf{x}, \mathbf{X}_t), C(\mathbf{x}, \mathbf{X}_s)].
\end{aligned} \tag{3.8}$$

Following Algorithm 2, we can sequentially sample a MGP while guarantee the sequentially sampled data points have the same property as if they are sampled simultaneously from the MGP. The proof of this property utilizes the basic properties of conditional multivariate normal distribution and is omitted here. With the sequential sample approach, gradient based optimization algorithms can be used in Line 8 of Algorithm 1 to find the optimal point of the sampled GP from the fitted posterior MGP directly.

3.3.3 Convergence Analysis

In this section, we present the convergence analysis of the proposed MAB-MGP-BO method. Bayesian regret has been used for a performance measure for the Bayesian optimization with Thompson sampling due to its sampling scheme [112]. The Bayesian regret is defined as follows. Assume the underlying function is f and $\mathbf{x}^*$ is the optimal solution of f, $\mathbf{x}^* = \text{argmax} f(\mathbf{x})$. In iteration t, the algorithm suggests $\mathbf{x}_t$ as the optimal solution, then $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ is the instantaneous regret of the algorithm. The Bayesian regret is the expectation of the summation of the instantaneous regret by the Tth iteration: $\mathit{BayesRegret}(T) = E[\sum_{t=1}^{T} r_t]$. With the Bayesian regret, we can assess the convergence rate of the algorithm by finding the bound of the marginal increment of the Bayesian regret $\mathit{BayesRegret}(T)/T$.

For the observed data in the tth iteration of the algorithm, suppose we choose the a_tth arm and the optimal point is $\mathbf{x}_t$, then we have:

$$y_t = f_{a_t}(\mathbf{x}_t) + \epsilon_t, \quad t = 1, \ldots, T \tag{3.9}$$

We define $f_a(\mathbf{x})$ is the MGP for the ath arm with mean zero and covariance function $\sigma^2 c(\mathbf{x}, \mathbf{x}')$, and $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_a \subset \mathbb{R}^d$. We denote the observed data of the ath output as $\mathcal{D}_t^a = \{(a_t, x_t, y_t) | a_t = a\}_{t=1}^T$ and the whole dataset by $\mathcal{D}_t = \cup_{a=1}^m \mathcal{D}_t^a$, where m is the number of arms.

According to our formulation, Bayesian regret of the MAB-MGP-BO model by the T iteration is defined as the expected difference between the underlying optimal value and the optimal value we find by the Bayesian optimization:

$$\mathit{BayesRegret}(T) = \mathbb{E}\left[\sum_{t=1}^{T} \{f_{a^*}(x^*) - f_{a_t}(x_t)\}\right] \tag{3.10}$$

To find the Bayesian regret of the MAB-MGP-BO method, we need two following assumptions. These assumptions are not restrictive and have been used in other convergence analysis works [122] and [99].

Assumption 1. *For all a and for any sample f from $\mathcal{GP}$, there exist constants $r, s > 0$ such that its partial derivatives satisfy the following condition*

$$\begin{aligned} &P(|\partial f / \partial x_i| < L) \geq 1 - dr \exp(-L^2/s^2), \\ &\forall L > 0, \forall i \in \{1, \ldots, d\}, \end{aligned} \tag{3.11}$$

where d is the dimension of the model input $\mathbf{x}$.

With this assumption, the partial derivatives of f is bounded in probability. Another assumption is on the maximum information gain of the model. The information gain is the mutual information shared by f and the observations $y_o = f(\mathbf{x}_o) + \epsilon_o$ on $\mathbf{x}_o \in O \subset \mathcal{X}$, where

$\epsilon_o \sim \mathcal{N}(0, \sigma^2)$. Then the information gain is defined as:

$$I(y_o; f) = H(y_o) - H(y_o|f) \tag{3.12}$$

$H(y_o)$ is the marginal entropy of the observation y_o and $H(y_o|f)$ is the conditional entropy of the observation y_o given the corresponding function value $f(\mathbf{x}_o)$. For a GP $f \sim \mathcal{N}(\mu, \Sigma)$, $H(\mathcal{N}(\mu, \Sigma)) = \log|2\pi e \Sigma|/2$. The information gain is a measure of the reduction in uncertainty of function f after knowing the observations y_o. The maximum information gain after T iterations is then defined as :

$$\gamma_T = \max_{O \subset D : |O| = T} I(y_o; f) \tag{3.13}$$

With the following Assumption 2, we can guarantee the maximum information gain of the model within a sub-linear bound.

Assumption 2. *The maximum information gain γ_{T_a} about f_a in Bayesian Optimization with Gaussian process is sub-linear in T_a which is the number of observations in the ath arm; there exists an α such that $\gamma_{T_a} \sim \mathcal{O}(T_a^{\alpha})$ where $0 \leq \alpha < 1$.*

Both assumptions 1 and 2 hold for our Multi-output Gaussian process. The kernel used for the GP in this work reduced to the form of Gaussian kernel with paired scale parameters corresponding to each category. Assumption 1 holds for any four-times differentiable covariance functions as stated in [122]; thus, it holds for the Gaussian kernel function. Assumption 2 holds for the Gaussian kernel function [99]. In other words, our Multi-output Gaussian process satisfies both the Assumptions 1 and 2.

Under the assumptions, the Bayesian regret of the MAB-MGP-BO method in Algorithm 4 can be bounded as follows. This result stated in Theorem 1 shows that the Bayesian regret bound of the algorithm is growing sub-linearly in T with factor $\sqrt{m}$. A sub-linear Bayesian regret implies the solution of MAB-MGP-BO will converge to the global optimal point.

Theorem 1.

$$\begin{aligned} BayesRegret(T) &= \mathbb{E}\left[\sum_{t=1}^{T}\{f_{a^*}(x^*) - f_{a_t}(x_t)\}\right] \\ &\leq \mathcal{O}\left(\sqrt{mT^{\alpha+1}\log T}\right), \end{aligned} \tag{3.14}$$

where $0 \leq \alpha < 1$.

We provide an outline of the proofs of the theorem. We use $f_a(x)|\mathcal{D}_t$ for the posterior distribution of the MGP instead of the posterior distribution of $f_a(x)|\mathcal{D}_t^a$, which is a conventional univariate output GP as they used. In other words, instead of using the dataset for specific a, we used whole dataset $\mathcal{D}_t$ leveraging all the information across a by using MGP. Theorem 1 can be proved after the following decomposition of the Bayesian regret.

$$\begin{aligned} BayesRegret(T) &= \mathbb{E}\left[\sum_{t=1}^{T}\{f_{a^*}(x^*) - f_{a_t}(x_t)\}\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a^*}(x^*) - f_{a_t}(x^*_{a_t})\right\}\right] \\ &+ \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a_t}(x^*_{a_t}) - f_{a_t}(x_t)\right\}\right] \qquad (3.15) \\ &= R_T^{MAB} + R_T^{BO}. \qquad (3.16) \end{aligned}$$

The boundary of Bayesian regret will be obtained by having the boundaries of R_T^{MAB} and R_T^{BO} in Lemma 2 and 3. For the proofs of Lemma 2 and 3, we state them in Appendix 3.7. In the MAB-MGP-BO model, the Bayesian regret consists of two parts. One is the Bayesian regret of the Multi-armed bandit model $R_T^{MAB} = \mathbb{E}[\sum_{t=1}^{T}\left\{f_{a^*}(x^*) - f_{a_t}(x^*_{at})\right\}]$, which is the expected cumulative difference between the underlying optimal value of all the arms and the optimal value of the selected optimal arm a_t from the MAB model. We define the Gaussian process model used for the function at the arm a as $\mathcal{GP}_a$. The boundary of R_T^{MAB} is presented in Lemma 2. Given the observations by the Tth iteration, the Bayesian regret generated from Multi-armed bandit model can be bounded by an sub-linear upper bound.

Lemma 2. *The Bayesian regret from Multi-armed bandit* R_T^{MAB} *in the MAB-MGP-BO* *model has an upper bound of* $\mathcal{O}(\sqrt{mT^{\alpha+1}\log T})$.

$$\begin{aligned} R_T^{MAB} &= \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a^*}(x^*) - f_{a_t}(x^*_{a_t})\right\}\middle|\mathcal{D}_T, \mathcal{GP}_1, \ldots, \mathcal{GP}_m\right] \\ &\leq \mathcal{O}(\sqrt{mT^{\alpha+1}\log T}), \qquad (3.17) \end{aligned}$$

where $0 \leq \alpha < 1$.

Another part of the Bayesian regret (3.16) is the regret of the Bayesian optimization $R_T^{BO} = \mathbb{E}[\sum_{t=1}^{T}\{f_{a_t}(x^*_{a_t}) - f_{a_t}(x_t)\}]$, which is the expected cumulative difference between the underlying optimal value within arm a_t and the optimal value obtained by the Bayesian optimization algorithm with arm a_t. Similarly, the boundary of R_T^{BO} is presented in Lemma 3. Given the observations by the Tth iteration, the Bayesian regret generated from Bayesian optimization can be bounded by an sub-linear upper bound.

Lemma 3. *The regret from Bayesian optimization* R_T^{BO} *in the MAB-MGP-BO model has an upper bound of* $b\sqrt{mT^{\alpha+1}\log T}$.

$$\begin{aligned} R_T^{BO} &= \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a_t}(x^*_{a_t}) - f_{a_t}(x_t)\right\}\middle|\mathcal{D}_t, \mathcal{GP}_1, \ldots, \mathcal{GP}_m\right] \\ &= \sum_{a=1}^{m}\sum_{t_a=1}^{T_a}\mathbb{E}\left[f_a(x^*_a) - f_a(x_{t_a})|\mathcal{D}_t, \mathcal{GP}_a\right] \qquad (3.18) \\ &\leq b\sqrt{mT^{\alpha+1}\log T}, \qquad (3.19) \end{aligned}$$

where b *is an arbitrary constant,* $0 \leq \alpha < 1$.

Combining Lemma 2 and 3, we have the Bayesian regret bound in Theorem 1. Then we have, $\lim_{T\to\infty} \mathit{BayesRegret}(T)/T = 0$. The expected cumulative regret is sub-linear and the average expected regret of the proposed model is convergent.

3.4 Illustration through Benchmark Example

In this section, we present two benchmark examples to compare the performance of the proposed MAB-MGP-BO method with other three Bayesian Optimization methods.

3.4.1 Methods Considered

To show the advantage of our methods, we compared the following 4 methods on the simulated functions:

- **Onehot-BO**[43]: BO method using one-hot encoding method to transform the categorical variable into additional continuous variables.
- **MAB-BO**[99]: Multi-armed bandit Bayesian Optimization method using traditional univariate output Gaussian process in each arm.
- **MAB-SMGP-BO**: This method is the same as our proposed MAB-MGP-BO method except that we use a separable covariance functions for the MGP. We adopt the separable covariance functions developed in [147]. In this approach, one positive definite matrix with unit diagonal elements (PDUDE) is constructed to measure the correlation among the functions from different arms. A spherical coordinate parameterization method is used to simplify the fitting algorithm for the hyperparameters of the PDUDE matrix.
- **MAB-MGP-BO**(Our proposed model in Section 3.3.2).

3.4.2 Simulated Function

In this simulation, we take the same synthetic functions used in [99]. One of the function we try to maximize is a two-dimensional function containing one continuous variable x and a categorical variable a. The expression of the function is shown in (3.20) and Figure 3.4.

$$\begin{aligned} f([a,x]) = \exp(-(z_1-2)^2) + \exp\left(\frac{(-z_1-6)^2}{10}\right) \\ + \frac{1}{z_2^2+1} + \frac{a}{2}, \end{aligned} \tag{3.20}$$

where $z_i = x + 0.05a(-1)^i, x \in [-2,10], a = 0,1,\ldots,5$.

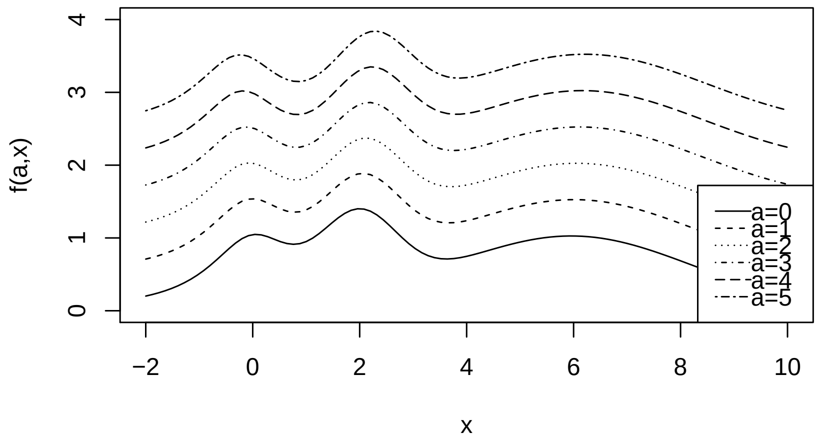


Figure 3.4: 2d Synthetic function used in simulation

We can see the set of functions follow a similar pattern and the maximum value are obtained with similar value of x. There are positive correlations between different functions and the proposed algorithms can take advantage of information from other functions to predict each function value. There are also fluctuations in the functions which are common properties of real world functions.

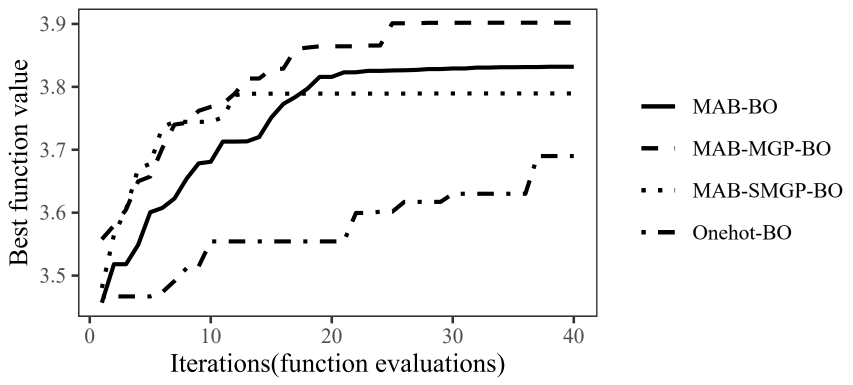
The second simulated function contains four continuous variables $\mathbf{x}$ and one categorical variable a. The expression of the function is shown in (3.21).

$$f([a,\mathbf{x}]) = \prod_{i=1}^{4} \sqrt{z_i} sin(z_i) + 2a, \tag{3.21}$$

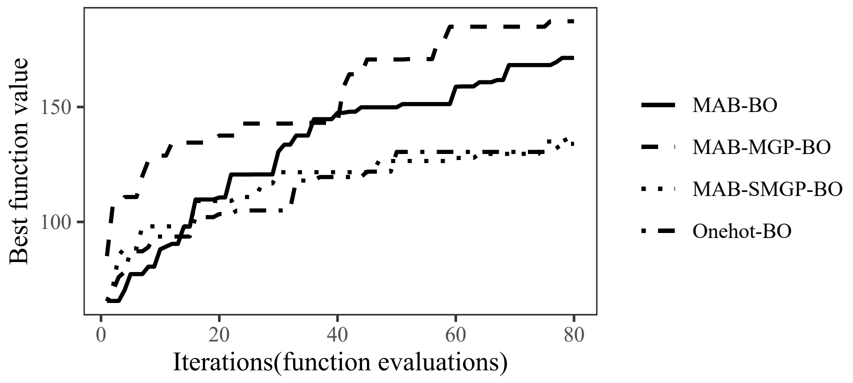
where $z_i = x_i + 2a, \mathbf{x} \in [1, 10]^4, a = 0, 1, \ldots, 5$.

Though there are 5 dimensions of continuous variables and we can not present a figure to show the similarities of the functions, we can see from the expression they share similar patterns and have positive correlations. The simulations are both validated 10 times and the mean of the best function values are shown in Figure 3.5.

We can see from both function optimizations that our proposed method can not only achieve a higher maximum value but also converge faster than the other three methods. Among the other three methods, MAB-BO model produces a higher optimal value, while



(a) 2d function



(b) 5d function

Figure 3.5: The comparison of our proposed method and other methods

the MAB-SMGP-BO model has a faster convergence speed in the earlier iterations. This result may be because the MAB-SMGP-BO model has more data information than the MAB-BO model in the earlier iterations and the fitting and prediction of the Gaussian process is faster at the beginning. However, as the model accumulates enough information in the later iterations, the separable structure of the MAB-SMGP-BO method restricts the fitting and can not achieve a higher optimal value than the MAB-BO method does. The proposed MAB-MGP-BO method, on the other hand, is not restricted by the separable covariance structure and can fit a better Gaussian process of the underlying function and obtain the best optimum in a faster rate.

3.5 Application to Structural Damage Diagnosis

In this section, we apply the proposed MAB-MGP-BO method for a real world structural damage identification using piezoelectric admittance measurement [18]. We will first give the problem description in Section 3.5.1. Then, a simulation study is conducted in Section 3.5.2. Lastly, the structural damage identification using real experimental data is shown in Section 3.5.3.

3.5.1 Problem Description

As shown in Figure 3.6, a piezoelectric transducer is attached to the host structure and the dynamic response of the structure in terms of structural admittance at different frequencies can be measured and observed. To build a finite element analysis (FE) model, the host structure is divided into 11250 elements. It is then divided into 25 segments, which are regarded as 25 locations for damage identification. With the FE model, we can link the structural damage, which is often modeled as a property change such as the stiffness loss at a specific element, with the observed dynamic response. Now the structural damage identification problem can also be considered as a black-box function optimization problem with categorical and continuous inputs, where the FE model is the black-box function, the location (i.e., segment) of the damage is the categorical input, and the severity of the damage is the continuous input. In structural damage identification, we try to find the location and severity of the damage by minimizing the difference between the FE model prediction and the experimental measurements. As presented in Figure 3.6, the PZT segments are located consecutively in a line and it is intuitive to assume there are correlations between adjacent segments. If the damage locations are near to each other, the admittance change should follow similar patterns.

We can consider the difference of the observed and the computed admittance change $||\Delta I_{\text{exp}} - \Delta I_{\text{FE}}||$ as the objective function we need to optimize. The location and severity of the damage $[l, s]$ is the input of the model, where $l \in \{1, \ldots, m\}$ is the categorical variable

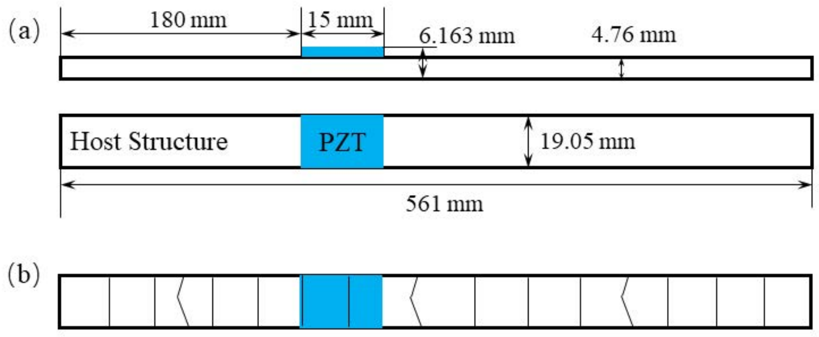


Figure 3.6: (a) Dimensions of the structure and PZT patch; (b)Diagram of division of segments

and $s \in \mathcal{S} \subset \mathbb{R}$ is the continuous variable. To identify the damage, we need to find:

$$[L^*, s^*] = \text{argmin}_{[l,s]} ||\Delta I_{\text{exp}} - \Delta I_{\text{FE}}(l, s)||. \tag{3.22}$$

In this case study, the FE model of the host structure contains $m = 25$ segments. To make comparison, we consider the same cases of damages as that considered in [18]. In admittance-based damage detection, the damage occurrence causes admittance changes around resonant peaks. Without loss of generality, we pick two frequencies, 14th (1893.58 Hz) and 21st (3704.05 Hz) natural frequencies, to conduct frequency sweeping. Two frequency ranges around the two natural frequencies (from 1891.69 to 1895.47 Hz and from 3700.35 to 3707.75 Hz) are used in the inverse analysis. The experimental setup is shown in Figure 3.7. The data points for the experimental measurements contained in each frequency range are detailed in Table I. The voltage drop is measured across a small resistor $R = 100\Omega$ which is connected in serial to the transducer. And the current in the circuit can be obtained which then yields the admittance information. A Dynamic signal analyzer (Agilent 35670 A) with a source channel and the sweep sine capability is utilized. The source channel is used to generate the sinusoidal voltage V_{in} sent to the piezoelectric transducer, and the output voltage V_{out} across the resistor is recorded.

A small mass block is introduced to emulate the damage, as shown in the Figure 3.7. The damage is introduced under assumption that the mass of the whole system now is unchanged,thus resulting in equivalent stiffness reduction. In Case I, the damage locates on

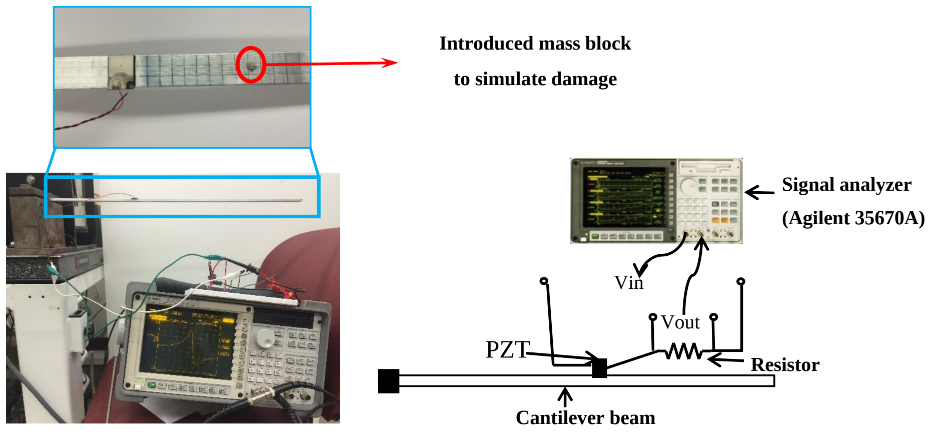

Introduced mass block to simulate damage
Signal analyzer (Agilent 35670A)
Vin
Vout
PZT
Resistor
Cantilever beam

Figure 3.7: Experimental setup

Table 3.1: Experimental cases considered

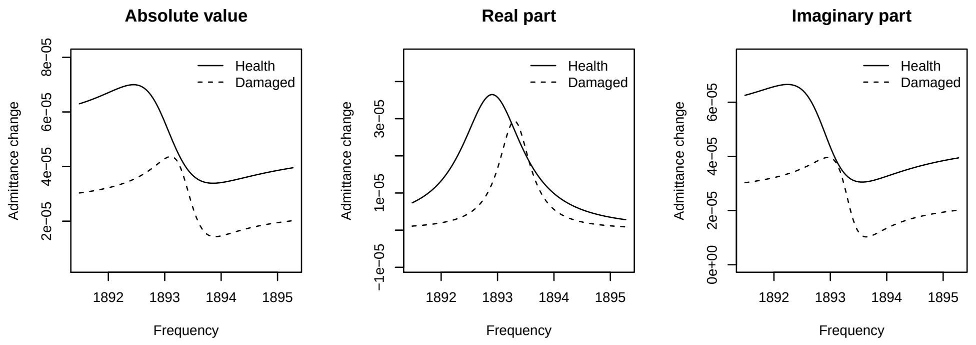
Experiment Case	Segment	Severity	Frequency Range (Hz)	# of Frequency Points
Case I	12	0.0016	[1891.69,1895.47]	100
Case II	14	0.0028	[3700.35,3707.75]	85

the 12th segment and the severity is equivalent to a stiffness loss of 0.16%. In Case II, the real damage locates on the 14th segment and the severity is equivalent to a stiffness loss of 0.28%.

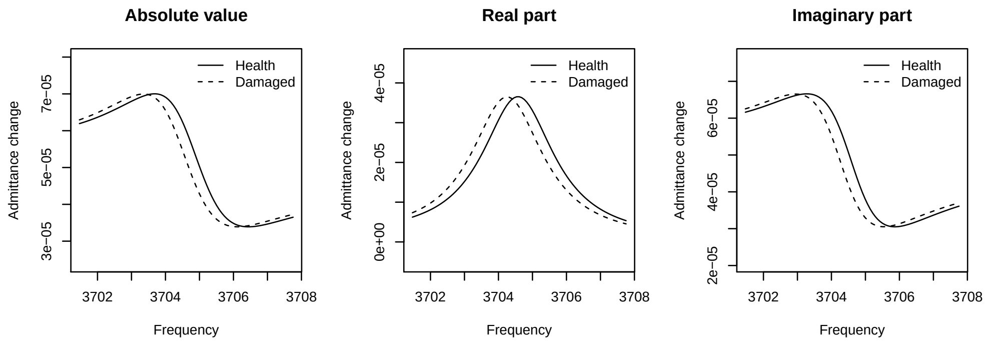
3.5.2 Structural Damage Identification Using Simulated Observations

First, we take the simulated admittance as shown in Figure 3.8 from the FE model as if they were the true observed admittance. The figures of each case are the absolute value, the real part and the imaginary part of the complex admittance respectively. We can see there is an overlay of admittance with and without damage in the figures. We attempt to locate the damage location and severity of the structure based on the admittance change.

Since there is no noise and other uncertainties in the simulated observations, we expect the proposed MAB-MGP-BO method can identify the damage quickly and accurately. Indeed, as shown in Figure 3.9, MAB-MGP-BO method identifies the correct damage location and has a very close damage severity estimation to the underlying damage severity. The comparison



(a) Case I



(b) Case II

Figure 3.8: The admittance change of the health and damaged structure

with the Multi-DIRECT method proposed in [18] is shown in Table 3.2. Our method can achieve more accurate estimation of the damage severity, which confirms the significance of non-linearity in the model.

Other three models are also applied on this simulation study, but there are some limitations. Because all the three other methods presented in the numerical study showed too bad performance, we did not include the results. For the MAB-SMGP-BO model, when the number of arms becomes large (25 in this case), the number of hyperparameters in the correlation matrix becomes $25^2 = 625$ in each iteration of Gaussian process fitting. This model takes too long time to complete, and it is not feasible on the simulation case. For the MAB-BO model, the convergence is relatively slow because the Bayesian optimization only considers the information of the current arm. For the Onehot-BO method, similar problem happens to the model convergence. The transformed data is too sparse for the Gaussian process to fit. Neither of the 2 methods can locate the correct damaged location within 200

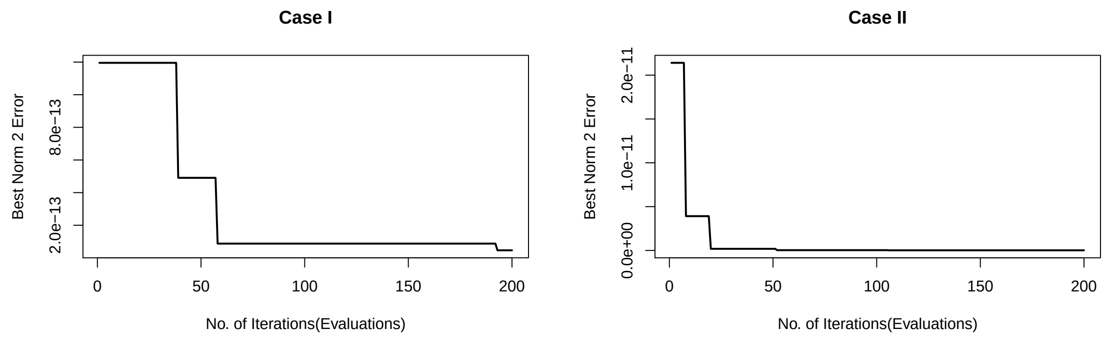


Figure 3.9: The performance on the simulated structural damage identification

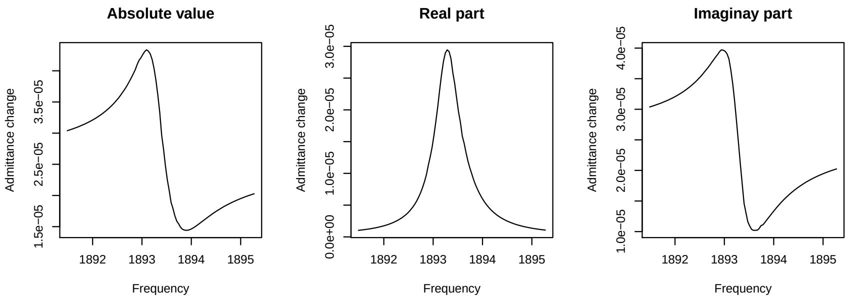
Table 3.2: Results comparison of two models

True damage [location, severity]	Model	Prediction	Estimation error
[12, 0.0016]	MAB-MGP-BO	[12, 0.00159]	**0.625%**
	Multi-DIRECT	[12, 0.0017]	6.25%
[14, 0.0028]	MAB-MGP-BO	[14, 0.00277]	**1.074%**
	Multi-DIRECT	[14, 0.003]	7.14%

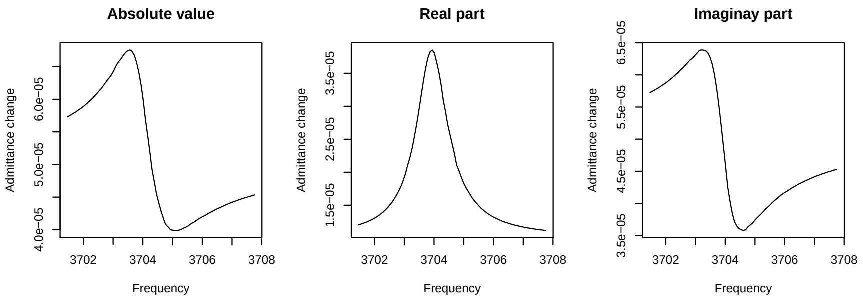
of iterations.

3.5.3 Structural Damage Identification based on Real Observations

In this section, we conduct the structural damage identification based on real observations as shown in Figure 3.10. Structural damage identification based on real observed responses is more challenging. First, there are always noises in the real observations. The measurement noise will not only make the damage severity estimation less accurate, but also possibly cause alias in the damage location. In other words, it is possible that other damage locations have closer simulated admittance change to the real observations. These noises may lead to the incorrect damage identification results. Second, the FE model itself is not perfect. There will always be modeling errors which will lead to bias in the damage identification. As a result, we generally cannot guarantee that the model based structural damage identification using Bayesian Optimization can always find the unique optimal solution. To avoid these difficulties in structural damage identification using real observations, we provide an extended algorithm that can give a set of possible solutions instead of a single solution. The algorithm we used to generate the group of optimal results are shown in Algorithm 6. The basic idea



(a) Case I



(b) Case II

Figure 3.10: The experimental observations of Admittance change for Case I and Case II

is simple: once we identify the current best solution, we remove it from the solution space and then find the best solution in the rest solution space. The result shows that the correct damage locations are included in the first several solutions.

The first several best results we found using MAB-MGP-BO for the two cases are shown in Table 3.3 and Table 3.4, respectively.

Table 3.3: Optimal results: experimental case I

Optimal results	Residuals	Damage [location,severity]
1	5.99337×10^{-8}	[23, 0.00109]
2	6.30363×10^{-8}	[12, 0.00181]
3	6.17052×10^{-8}	[9, 0.00140]

Algorithm 6 Optimal results generation

```
1: $t$: number of iterations, $tol$: largest number of iterations of one optimal result.
2: $\alpha_t^*$: optimal arm in iteration $t$. $s_t^*$: optimal severity in iteration $t$.
3: $\mathcal{G}$: Group of optimal results.
4: for $t = 1, 2, \ldots$ do
5:     if $r == tol$ then
6:         Save the optimal result. $\mathcal{G} = \mathcal{G} \bigcup \{[\alpha_t^*, s_t^*]\}$
7:         Remove all the points from the temporal optimal arm $\alpha_{t-1}^*$.
8:         $r = 1$. Restart the count of number of iterations with the same optimal result.
9:     end if
10:    Run the MAB-MGP-BO model and find the optimal arm $\alpha_t^*$.
11:    if $\alpha_t^* == \alpha_{t-1}^*$ then
12:        $r = r + 1$.
13:    end if
14: end for
```

Table 3.4: Optimal results: experimental case II

Optimal results	Residuals	Damage [location,severity]
1	6.57408×10^{-8}	[23, 0.00209]
2	6.593299×10^{-8}	[24, 0.00209]
3	6.96688×10^{-8}	[25, 0.00217]
4	6.593457×10^{-8}	[9, 0.00299]
5	6.595832×10^{-8}	[14, 0.00298]

For both cases, the underlying true damage location are successfully detected in the first several solutions. Considering the noise of measurement and the bias of the FE model, this result is satisfactory and we can expect to utilize the proposed model in practical applications.

3.6 Conclusion

In this article, we propose a Multi-armed bandit method using Bayesian Optimization with Multi-output Gaussian process to solve Structural Health Monitoring problems, especially in complex systems. This method can be also applied to other fields of engineering applications, such as design optimization with complicated structures and hyperparameter tuning of surrogate models.

The proposed method utilizes the information of the data collected under all the categorical input values to make prediction of the black-box function under a specific categorical input value. We also presented the convergence analysis of the proposed method and provide a Bayesian regret bound of the algorithm. Numerical benchmark examples are conducted to show the proposed model has a better performance than several other existing Bayesian Optimization methods. We also apply our model to a real world case study on structural damage identification. The results show the proposed method can identify the location and the severity of the damage with a better performance than the existing fault diagnostics Multi-DIRECT algorithm.

In addition to the diagnosis problem, the mixed input Bayesian optimization strategy proposed in this article can also be applied to many design problems, where both categorical inputs (e.g., material selection, geometry configuration selection) and continuous inputs influence the design performance. The proposed strategy can utilize the information under different categorical inputs and achieve the optimal design within the design space spanned by both categorical and continuous variables.

A limitation of MAB-MGP-BO method is that the computation load is heavy when the number of categories is very large. A MGP is fitted in each category of the data and it is hard to find a way to simplify the computation because of the complexity of the mathematical formulation. Though the data we fit the MGP to for different categories are the same, the assumption and the category of interest and the calculated covariance matrix of fitted categories can not be reused. Other kinds of constructions of MGP can be explored to make the algorithm more scalable. We will extend the algorithm in this direction and report the findings in the near future.

3.7 Appendix: Proof of Convergence

3.7.1 Lemmas used

Following Lemma 4 and 5 will be used to prove Lemma 2 and Lemma 3.

Lemma 4. *The Bayesian regret of the T_ath iterations of the ath arm has an upper bound of*

$\mathcal{O}\left(\sqrt{T_a^{\alpha+1}\log T_a}\right)$.

$$\begin{aligned}
BayesRegret(T_a) &= \mathbb{E}\left[\sum_{t_a=1}^{T_a}\left\{f_{a^*}(x^*) - f_{a_{t_a}}(x_t)\right\}\middle|\mathcal{D}_{T_a}\right] \\
&\le \mathcal{O}\left(\sqrt{\gamma_{T_a}T_a\log T_a}\right) && (3.23) \\
&\le \mathcal{O}\left(\sqrt{T_a^{\alpha+1}\log T_a}\right) && (3.24)
\end{aligned}$$

where γ_{T_a} is the maximum information gain about $f_a(x)$ after T_a iterations.

The boundary in (3.23) is proved by [112, 122]. Using the assumption 2 (i.e., $\gamma_{T_a} \sim \mathcal{O}(T_a^{\alpha})$), the boundary of (3.24) can be obtained.

Bayesian simple regret is defined for the Bayesian regret of individual function rather than summation over time. Then, the boundary of the Bayesian simple regret can be obtained as follows.

Lemma 5. *The Bayesian regret of individual function by the T_ath iteration has an upper bound of $\mathcal{O}\left(\sqrt{\frac{\log T_a}{T_a^{1-\alpha}}}\right)$*

$$\begin{aligned}
BayesSimpleRegret(T_a) &= \mathbb{E}\left[f_{a^*}(x^*) - \max_{t\le T_a} f_{a_t}(x_t)\middle|\mathcal{D}_{T_a}\right] \\
&\le \mathcal{O}\left(\sqrt{\frac{\gamma_{T_a}\log T_a}{T_a}}\right) \\
&\le \mathcal{O}\left(\sqrt{\frac{\log T_a}{T_a^{1-\alpha}}}\right) && (3.25)
\end{aligned}$$

3.7.2 Proof of Lemma 2

Proof.

$$\begin{aligned}
R_T^{MAB} &= \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a^*}(x^*) - f_{a_t}(x^*_{a_t})\right\}\middle|\mathcal{D}_T, \mathcal{GP}_1, \ldots, \mathcal{GP}_m\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T}\left\{f_{a^*}(x^*) - U_t(a^*)\right\}\middle|\mathcal{D}_T, \mathcal{GP}_1, \ldots, \mathcal{GP}_m\right]
\end{aligned}$$

$$+\mathbb{E}\left[\sum_{t=1}^{T}\left\{U_t(a_t)-f_{a_t}(x^*_{a_t})\right\}\middle|\mathcal{D}_T,\mathcal{GP}_1,\ldots,\mathcal{GP}_m\right], \tag{3.26}$$

where $U_t(a)$ is an upper confidence bound that is a determined function of a. a_t is a random variable selected from the posterior sampling. Here, our function $f_a(x)$ is a random variable, and we do not know the true function $f_a(x)$. Therefore, we select the a_t based on the posterior samples of functions from each arm. In particular, we select a_t by $\max_{a,x}\tilde{f}_a(x)$, where $\tilde{f}_a(x)$ is posterior samples given $\mathcal{D}$. Because posterior sampling $\tilde{f}_a(x)|\mathcal{D},\mathcal{GP}_a$ is precise, we can claim that the distributions $p(a^*)$ and $p(a_t|\mathcal{D},\mathcal{GP}_1,...,\mathcal{GP}_m)$ are identically distributed. This argument is fundamental in the Bayesian regret proof of Thompson sampling used in [112]; since the Thompson sampling at each arm precisely uses the posterior distribution to propose a_t at iteration t, both a_t and a^* are identically distributed conditioned on $\mathcal{D}_T$ (Eq. (3.26)).

We select $U_t(a)$ as follows.

$$U_t(a)=\mathbb{E}\left[\max_{t'\le t_a}f_a(x^a_{t'})+a\sqrt{\frac{\log t_a}{t_a^{1-\alpha}}}\middle|\mathcal{D}_t,\mathcal{GP}_a\right] \tag{3.27}$$

where t_a is the number of times that a is selected during t times of total iterations, and a is an arbitrary constant. Then, the first term in (3.26) is non-positive by the definition of (3.27), that is,

$$\mathbb{E}\left[\sum_{t=1}^{T}\{f_{a^*}(x^*)-U_t(a^*)\}\middle|\mathcal{D}_T,\mathcal{GP}_1,\ldots,\mathcal{GP}_m\right]\le 0$$

.

The second term in (3.26)

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}\left\{U_t(a_t)-f_{a_t}(x^*_{a_t})\right\}\middle|\mathcal{D}_T,\mathcal{GP}_1,...,\mathcal{GP}_m\right]\\
&\le\mathbb{E}\left[\sum_{t=1}^{T}\{U_t(a_t)-L_t(a_t)\}\middle|\mathcal{D}_T,\mathcal{GP}_1,...,\mathcal{GP}_m\right]\\
&=\sum_{a=1}^{m}\mathbb{E}\left[\sum_{t_a=1}^{T_a}\{U_{t_a}(a)-L_{t_a}(a)\}\middle|\mathcal{D}_T,\mathcal{GP}_a\right]
\end{aligned}$$

$$
\begin{aligned}
&= a \sum_{a=1}^{m} \sum_{t_a=1}^{T_a} \sqrt{\frac{\log t_a}{t_a^{1-\alpha}}} \\
&\leq a\sqrt{\log T} \sum_{a=1}^{m} \sum_{t_a=1}^{T_a} \frac{1}{\sqrt{t_a^{1-\alpha}}} \\
&\leq 2a\sqrt{mT^{\alpha+1}\log T} \qquad (3.28)
\end{aligned}
$$

where $L_t(a_t)$ is defined as $L_t(a_t) = \mathbb{E}\left[\max_{t' \leq t_a} f_a(x_{t'}^a)|\mathcal{D}_t\right]$ because $L_t(a_t) \leq \mathbb{E}\left[f_{a^*}(x^*)|\mathcal{D}_t\right]$. Here, again, we sum the Bayesian regret bound obtained for each arm. Therefore, the Bayesian regret bound of R_T^{MAB} is Eq. (3.17)

■

3.7.3 Proof of Lemma 3

Proof.

$$
\begin{aligned}
R_T^{BO} &= \mathbb{E}\left[\sum_{t=1}^{T} \left\{f_{a_t}(x_{a_t}^*) - f_{a_t}(x_t)\right\} \middle| \mathcal{D}_t\right] \\
&= \mathbb{E}\left[\sum_{a=1}^{m} \sum_{t_a=1}^{T_a} \{f_a(x_a^*) - f_a(x_{t_a})\} \middle| \mathcal{D}_t\right] \\
&\leq \mathbb{E}\left[\sum_{a=1}^{m} b\sqrt{T_a^{\alpha+1}\log T_a} \middle| \mathcal{D}_t\right] \\
&\leq b\sqrt{mT^{\alpha+1}\log T} \qquad (3.29)
\end{aligned}
$$

where b is an arbitrary constant. ■

Chapter 4

Allocating Robots/Cobots to Production Systems for Productivity and Ergonomics Optimization

Abstract

Collaboration between humans and robots has great promise in manufacturing systems. Utilization of cobots in a manufacturing system can improve both productivity and ergonomics. In this paper, we study the problem of how to allocate limited cobot/robots to manufacturing systems with multiple workstations so that an integrated performance measure, considering both productivity and ergonomics is optimized. Previous work on cobot/robot allocation in manufacturing systems focus on the decomposition of tasks for a single workstation into multiple work elements, and then split them between human and robots, rather than studying multi-machine systems. To bridge this gap, we consider the allocation of cobots/robots to a multi-stage manufacturing system. Specifically, we establish an integrated performance measure and formulate cobot/robot allocation into a constraint integer programming problem. With this formulation, we obtain the optimal allocation of one available cobot/robot in

This chapter is based on the paper: **Huang, C.**, Zhou, S., Li, J. & Radwin, R. (2023). Allocating Robots/Cobots to Production Systems for Productivity and Ergonomics Optimization, IEEE Transactions on Automation Science and Engineering. DOI:10.1109/TASE.2023.3270207.

simulated production systems, based on the integrated performance measure of productivity and ergonomics. Furthermore, allocation problem of production systems with multiple cobots/robots are considered and solved with a scalable algorithm.

4.1 Introduction

Robots have been used in manufacturing systems for several decades and have led to significant productivity improvement [100]. Traditional robots need to work in an enclosed space, separate from human workers so that they can accomplish their jobs safely. In recent years, a new type of robot, collaborative robots (also called cobots), has been emerged and gained significant growth in manufacturing applications [30]. Cobots are designed to work alongside human operators, rather than in their own enclosed space, to assist the human workers on various tasks, including assembly, screwing, material handling, and pick-and-place. By sharing the work space with human operators, not only can floor space be saved, they can work collaboratively with the worker to finish complex and delicate operations, otherwise difficult for a traditional robot. Another significant benefit of using cobots is that stress and strain on human worker can often be substantially reduced [47, 82]. According to the Collaborative Robots Market Report in 2021 [1], the collaborative robot market share is predicted to grow substantially in the coming years. With the far-reaching benefit and the rapid development of robot technology, human-robot collaborative manufacturing will contribute to the future smart manufacturing and factory automation.

Due to the significant growth of cobot applications, recent attention has been paid to allocating robots to workstations to optimize system productivity. For example, Maganha *et al.*created a model that designs schedules for tasks to minimize the number of robots needed [85]. In 2020, Weckenborg *et al.*formulated a mixed-integer programming problem to group the tasks in different stations with respect to cycle time [140]. The makespan of the system has been taken as an optimization objective in many cobot scheduling studies to improve productivity performance. A constraint programming based algorithm was proposed to minimize makespan in the print circuited boards industry [95]. Faccio *et al.*introduced a model to allocate different work proportions for humans and robots based on the characteristic of the task and makespan of the system [32]. Chen *et al.*applied a logic mathematical method to calculate the integrated cost of assembly time and worker payment and utilized a genetic revolutionary model to allocate subtasks regarding the cost-effectiveness requirement [19].

In addition to the productivity performance of the system, ergonomics factors for human workers in production systems are also an important concern. Ergonomics, which is one of the motivations for involving robots in manufacturing operations, includes high forces, repetitive motions, awkward postures, and exposure to vibration from equipment and tools. Numerous studies have been implemented to assess and improve ergonomics in manual operations [4, 132, 134].

Only a few existing efforts focus on robot/cobot allocation problems considering both productivity and ergonomics factors. In [103], an optimization framework is presented to allocate work between a human operator and a cobot to minimize production makespan and ergonomics, which provides insights to how to integrate a single cobot into a manufacturing process at one workstation. In [97], a mixed integer programming model is proposed to find optimal job rotation schedules in human-based production systems, with the goal to maximize production rate while simultaneously reducing and balancing human workload and ergonomics. However, the applications of robot/cobot are not considered in their work. In [26], a genetic approach for solving assembly line balancing problem both human and cobots is introduced, where the goal is to minimize the assembly line cost, the number of skilled workers on the line, and the energy load variance among workers, based on their energy expenditure. In [123], a similar assembly balancing problem is considered under a mixed linear integer programming framework. The energy expenditure is used as a measure of human operator strain. But the impact on processing time for collaborative operations is not considered in the model and the formulation is restricted to linear optimization. Recently, a unified measurement, referred to as throughput rate per unit of work effort time, to evaluate both productivity and ergonomics is proposed [145]. The trade-offs between the two metrics (productivity vs ergonomics) is characterized. However, only the evaluation of performance for a single workstation is considered in terms of throughput and ergonomics under a given configuration of human-cobot collaboration. The allocation strategies for human workers and robots/cobots are not investigated.

In this paper, we study the problem of how to allocate a limited number of robots and cobots in a manufacturing system so that both system throughput and ergonomics for human workers are optimized. Note that although there exist a number of recent research addressing on design and programming of robots to make them collaborate with human workers effectively [35, 41, 70, 106, 119], we focus on the methodology to determine which workstations in a production system should adopt robots/cobots to improve productivity and reduce ergonomics, rather than the design of a collaborative task.

Specifically, we propose a mixed nonlinear integer programming framework to solve the following allocation problem: Given a human-based manufacturing production system with multiple workstations and a limited number of robots/cobots, what is the best allocation of these robots/cobots to the selected workstations so that an integrated measure considering both system throughput and ergonomics is optimized? Such a problem is different from an assembly line balancing problem dealing with a new production line. Instead, the proposed method is more useful when introducing robots/cobots to existing systems. The main contributions of the proposed work are the following:

- The problem of robot/cobot allocation to a manufacturing system is formulated into a nonlinear optimization problem, which integrates both productivity and ergonomics considerations.
- Allocation of simulated production systems with one available cobot/robot is conducted and insights to the features of optimal robot/cobot allocation are obtained.
- Production systems with multiple cobots/robots are considered and to reduce the computation load, the original problem is re-formulated into a mixed integer nonlinear programming problem and solved through an effective and scalable procedure.

The remaining article is organized as follow. In Section 4.2, the problem of allocating robots/cobots to a production system is described and the framework of the optimization problem is introduced. In Section 4.3, we consider the scenarios when there is only one cobot or robot available in the production systems and obtain the optimal allocation solution with the proposed formulation. In Section 4.4, more general cases with multiple cobots/robots are simulated and solved by a scalable optimization solver to address the challenge of combinatorial explosion. The explanations of the allocation strategies are presented in both sections to verify the optimization results. In Section 4.5, we discuss the results from the generated cases and provide insights for future allocation problems. In Section 4.6, conclusions are summarized and future works are discussed.

4.2 Problem formulation

In this section, we first define a generic human-robot collaborative assembly system and introduce the assumptions in Section 4.2.1. Then, in Section 4.2.2, we present a strain index model for measurement of ergonomics performance. Finally, we combine the productivity performance and the strain index model and propose an integrated objective function to be used in the optimization model and the optimization framework of the robot/cobot allocation problem formulated in Section 4.2.3.

4.2.1 System description

In this study, we consider a collaborative assembly system with three serial production lines as shown in Figure 4.1. The Moore and Garg Strain Index (SI) [124] is used to quantify physical stress exposure. The stochastic model to describe the operations of such sysatems is defined by the following assumptions.

- Each serial line of the assembly system consists of multiple workstations and the work pieces flow through the stations from left to right. There are n_i workstations, where the jth station is denoted as W_j^i. The set of all workstations is represented by $\mathcal{W}$, where $\mathcal{W} = \{W_1^m, \ldots, W_{n_m}^m\}, m = 1, 2, 3\}$.
- There is infinite buffer capacity between each pair of stations in the system.
- Each workstation can be operated by a human worker (h), a robot (r), or through a collaborative operation (c). The job processing time at workstation w is assumed to follow an exponential distribution with parameter λ_w^p. , $p \in \mathcal{P}$, where $\mathcal{P}= \{h, r, c\}$ is

the set of operation alternatives. If the job at workstation w cannot be carried out by an alternative p, λ_w^p is set as $\epsilon 1 \to 0$.

- We assume all workstations in the production system are operated by human workers initially. Then q_r robots and q_c cobots will be allocated to replace $q_r + q_c$ human operated workstations in the system. When a robot is allocated to a workstation, the SI for that station will become zero as no human operator is involved. When a cobot is allocated to a workstation, the SI will be reduced for the workstation.

- Two serial lines are operated independently and then merged, followed by another serial line.

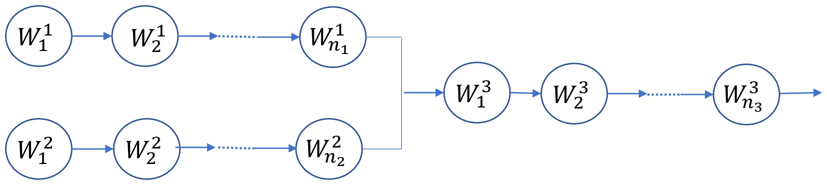


Figure 4.1: Collaborative assembly system model

Note that as illustrate in Section 4.2.3, a faster station needs to wait for the slower station to finish the work, resulting in no blockage in the system. This implies that all stations work independently and all workers and robots stay in the same workstation. Such an assumption can simplify system throughput calculation. However, it will not alter the essential structure of the optimization problem for robot/cobot allocation. The possible ways to relax this assumption are discussed in Section 4.6.

To formulate the robot/cobot allocation problem, we introduce $x_w^p \in \{0, 1\}$ as the indicator for operation alternative p assigned to workstation w. When a job at workstation w is assigned as p, $p \in \{h, r, c\}$, then $x_w^p = 1$, otherwise $x_w^p = 0$. In addition, $x_w^h + x_w^c + x_w^r = 1$.

4.2.2 Strain index model

To measure the ergonomic performance of the system, the strain index model in [145] is used. This strain index was first proposed in [124] and has been verified to be an reliable and effective way to evaluate the risk level related to hand workload over time [27, 110]. Based on assumption M4 in [145], the strain index of a manual operation can be evaluated by six risk factors of the overall process, denoted as set $\mathcal{S}= \{IE, DE, DM, HWP, SW, DD\}$. The definition of the six risk factors are following.

- Intensity of exertion (IE): An estimate of the strength required to perform the task on time.

- Duration of exertion (DE): The physiological and biomechanical stresses related to how long an exertion maintained.

- Efforts per minute (EM): The number of exertions per minute, which is synonymous with the frequency.

- Hand/wrist posture (HWP): An estimate of the position of the hand or wrist relative to a neutral position.

- Speed of work (SW): An estimate of how fast the worker is working.

- Duration per day (DD): A measure obtained from plant personnel.

For each task and each hand, all risk factors are evaluated and assigned to one of five categories, each corresponding to a rating based on a scale of $1-5$ as well as a multiplier value. The SI for each workstation is the product of the six ratings of the workstation. Then, the score is compared to the multiplier to identify the the level of task risk. At workstation w, with operation alternative p, the standard or data range in each category is characterized by $r^p_{w,j}$, for factor $j \in \mathcal{S}$ and the corresponding multiplier value of risk factor j in each category is denoted as $S^p_{w,j}$. The detailed category values are shown in Table 4.1. The ratings of risk factors IE and HWP have five levels from light to hard, since they cannot be evaluated numerically. Other risk factors can be measured numerically and are further categorized into different levels.

The strain index of workstation w using alternative p is defined as SI^p_w. In mathematical terms, the SI of a manual operation at workstation w with operation alternative p is the

Category rating	IE		DE		EM		HWP		SW		DD	
	r_{IE}	S_{IE}	$r_{DE}(\%)$	S_{DE}	r_{EM}	S_{EM}	r_{HWP}	S_{HWP}	r_{SW}	S_{SW}	r_{DD}	S_{DD}
1	Light	1	<10	0.5	<4	0.5	Light	1			$0-1$	0.25
2	Somewhat hard	3	$10-29$	1	$4-8$	1	Somewhat hard	1			$1-2$	0.5
3	Hard	6	$30-49$	1.5	$9-14$	1.5	Hard	1.5	<0.5	1	$2-4$	0.75
4	Very hard	9	$50-79$	2	$15-19$	2	Very hard	2	$0.5-1$	1.5	$4-8$	1
5	Near maximal	13	≥ 80	3	≥ 20	3	Near maximal	3	>1	2	>8	1.5

Table 4.1: Category rating of strain index

product of values for six factors, i.e.,

$$SI_w^p = \prod_{j\in\mathcal{S}} S_{w,j}^p. \tag{4.1}$$

In previous work, the physical stress exposure of workers in different workstations were not combined together, while we expect to have an overall ergonomics measure for the entire system in this work. Consider systems with the same SI in the most stressful workstation, we definitely prefer the system with smaller SI in the other workstations. Intuitively, the physical stress exposure for the entire system is defined as the sum of the SI of all workstations, i.e.,

$$SI_{all} = \sum_{w:x_w^h=1 \ or \ x_w^c=1} SI_w^p = \sum_{w:x_w^r=0} \prod_{j\in\mathcal{S}} S_{w,j}^p. \tag{4.2}$$

There are two advantages applying summation to characterize the stress exposure for the entire system. First, the derivative of the sum operation can be easily calculated, and the following optimization problem can be simplified. Second, the definition of the physical stress exposure for the entire system can capture the change of SI in all the individual workstations. If SI in any of the workstations varies, the integrated measure will change monotonously.

Besides optimizing the measure of the overall ergonomics in the entire system, we also expect less SI values in the most stressful workstation particularly. A threshold K is set to avoid high strain index at each workstation. Thus, the following constraint exists:

$$SI_w^p < K, \quad \forall w \in \mathcal{W}, \forall p \in \mathcal{P}. \tag{4.3}$$

The concept of a threshold K has been previously used for evaluating the SI for a given job. An $SI <= 3$ has been considered safe, while an $SI > 13.5$ has been considered hazardous [40]. Here, K is a parameter that can be adjusted based on the real world scenario, which provides some flexibility in the allocation problem. When the variance of SI plays a more important role in the requirement, a smaller K value can be selected. On the other hand, when the overall ergonomics is more critical in the allocation, a larger K value can be taken.

Since the work split of human and cobot can influences the strain index and the processing time of the workstations with human-cobot collaborative operations, without loss of generality, we define $\theta_w \in (0.1, 0.9)$ as a parameter to determine the proportion of work that a cobot handles in the workstation.

4.2.3 Integrated performance measure and optimization framework

The integrated performance measure takes both productivity and ergonomic performance of the system into consideration. We use the throughput TP of the system as the productivity measure and the strain index quantifies the ergonomics behavior. Since the workstations are operating independently and the assembly system has infinite buffer capacity and is connected serially and then merged, the throughput of the system is the processing rate of the slowest workstation. Thus we have

$$TP = \min\{\lambda_w^p | x_w^p = 1, w \in \mathcal{W}\}. \tag{4.4}$$

Then, the integrated performance measure of the system, denoted as E, is defined in (4.5).

$$\begin{aligned} E &= TP - \eta SI_{all} \\ &= \min\{\lambda_w^p | x_w^p = 1, w \in \mathcal{W}\} - \eta \sum_{w: x_w^r = 0} \prod_{j \in \mathcal{S}} S_{w,j}^p \end{aligned} \tag{4.5}$$

As shown above, there is a tradeoff between the productivity measure and the ergonomics performance, and $\eta > 0$ is the tradeoff coefficient.

To allocate robots/cobots to a production system, the integrated performance measure E of the system will be maximized. First, a production system with human workers only is treated as a baseline system, whose parameters are denoted with superscript o. For example, the baseline processing rate of workstation w is λ_w^o and the baseline ergonomic risk factors of workstation w is $S_{w,j}^o, j \in \mathcal{S}$. Also, TP^o= $\min\{\lambda_w^o | w \in \mathcal{W}\}$ is the throughput of the baseline human only system. Then, based on the following assumptions, the change of processing rate and strain index from the baseline system can be defined if a cobot or robot is allocated to workstation w:

- The processing rate of workstation w is changed by coefficient ρ_w^p.
- Risk factor DD of workstation w remains the same as in the baseline system so that $S_{w,DD}^p = S_{w,DD}^o,\ p \in \{h, c\}$. This is because the human worker in the collaborative operation still follows the general working schedule and the work durations per day are the same under manual and collaborative alternative.
- Risk factors IE and HWP of workstation w are not dependent on processing time and their changes are defined by coefficients $\beta_{w,IE}$ and $\beta_{w,HWP}$, thus $S_{w,j}^c = S_{w,j}^o * \beta_{w,j}, j \in \{IE, HWP\}$. Since the original rating of these factors are not numeric, their changes are made on the strain index directly and the calculated strain indices are continuous.
- The system throughput TP is constrained by the slowest workstation. Thus, faster workstations will be waiting after finishing their jobs. DE, the duration of the exertion is changed based on both TP of the whole system and the processing rate of the workstation. We have $r_{w,DE}^p = r_{w,DE}^o * \frac{\lambda_w^o}{\lambda_w^p} * \frac{TP}{TP^o},\ p \in \{h, c\}$.
- Strain indices EM and SW of a workstation are proportional to the station's processing time. Since the whole system is at the same pace, the effort per minute and the speed of work are only related to the throughput of the system. Thus $r_{w,j}^p = r_{w,j}^o * \frac{TP}{TP^o},\ p \in \{h, c\},\ j \in \{EM, SW\}$.

Then, a mathematical formulation to optimize the integrated performance measure of the system can be expressed as

$$\underset{x_w^p, \theta_w, w \in \mathcal{W}, p \in \mathcal{P}}{\text{Maximize}}\ E = TP - \eta \sum_{w: x_w^r = 0} SI_w$$

$$s.t.\ \sum_{w \in \mathcal{W}} x_w^r + x_w^c \le q_r;\ \sum_{w \in \mathcal{W}} x_w^h \le q_h; \tag{4.6a}$$

$$\sum_{w\in\mathcal{W}} x_w^r + x_w^c + x_w^h = |\mathcal{W}|; \tag{4.6b}$$

$$TP = \min\{\lambda_w^p | x_w^p = 1, w \in \mathcal{W}\}; \tag{4.6c}$$

$$TP^o = \min\{\lambda_w^o | w \in \mathcal{W}\}; \tag{4.6d}$$

$$SI_w = \prod_{j\in\mathcal{S}} S_{w,j}^p < K, \;\; for \;\; x_w^r = 0; \tag{4.6e}$$

$$\lambda_w^p = \frac{\lambda_w^o}{\rho_w^p}, \;\; p \in \{r, c\}; \;\; \lambda_w^h = \lambda_w^o; \tag{4.6f}$$

$$S_{w,DD}^p = S_{w,DD}^o, \;\; p \in \{h, c\}; \tag{4.6g}$$

$$S_{w,j}^c = S_{w,j}^o * \beta_{w,j}, \;\; j \in \{IE, HWP\}; \tag{4.6h}$$

$$r_{w,DE}^p = r_{w,DE}^o * \frac{\lambda_w^o}{\lambda_w^p} * \frac{TP}{TP^o}, \;\; p \in \{h, c\}; \tag{4.6i}$$

$$r_{w,j}^p = r_{w,j}^o * \frac{TP}{TP^o}, \;\; p \in \{h, c\}, \;\; j \in \{EM, SW\}. \tag{4.6j}$$

In eqs. (4.6a) and (4.6b), the constraints on available numbers of robots and cobots are included and the summation of all workers and robots used in the system should be the same as the number of workstations. The formal expression of system throughput is introduced in eqs. (4.6c) and (4.6d). The threshold constraint of strain index in each non-robot workstation is shown in (4.6e). In eqs. (4.6h) to (4.6j), calculations of strain indices are included.

In this formulation, the values of parameters $\beta_{w,IE}, \beta_{w,HWP}, \rho_w^c$ need to be determined, which are related to the change in strain index from a manual operation to a human-cobot collaborative operation. Such parameters are dependent on the design of collaborative operation and split of workload between human and cobot. Without loss of generality, these parameters can be expressed by a function of cobot proportion parameter θ_w at workstation w. In this paper, functions in (4.7) determine the values of $\beta_{w,IE}, \beta_{w,HWP}, \rho_w^c$ for a given θ_w.

$$\beta_{w,IE} = \frac{2}{3}(1 - \theta_w) \tag{4.7a}$$

$$\beta_{w,HWP} = \frac{1}{3}(1 - \theta_w) \tag{4.7b}$$

$$\rho_w^c = \rho_{w,o}^c + \theta_w - 0.5 \tag{4.7c}$$

The above three functions are all monotone and are linearly related to a cobot's work split in a collaborative operation. In (4.7c), $\rho^c_{w,o}$ is the baseline parameter of mean processing rate under different work split of cobot, and the related change coefficient ρ^c_w is in the range of $[\rho^c_{w,o} - 0.4, \rho^c_{w,o} + 0.4]$. The smaller the work split of a cobot, the faster the operation speed. In addition, eqs. (4.7a) and (4.7b) are modified versions of strain index change in [145], where $\beta_{w,IE} = \frac{2}{3}$ and $\beta_{w,HWP} = \frac{1}{3}$ to used to characterize the difference in strain index between collaborative and manual operations, respectively. Here, influence of θ_w is included in the expression, and the smaller the work split of cobot, the less difference from the baseline model. Although only three functions are discussed in this work, other customized functions characterizing collaborative operations can be applied as well, and the optimization model can be easily adapted to any continuous customized functions.

4.3 Production systems with one available cobot/robot

In this section, we consider production systems with one available cobot/robot resource and use the formulated model in Section 4.2 to solve the allocation problem. We simulate several numerical cases designed to mimic real world applications to gain some insights to which workstation will be first replaced by cobot or robot in a manufacturing system.. In Section 4.3.1, we work on cases where there is only one available cobot and cobots with different working efficiency are considered. In Section 4.3.2, there is only one available robot in the systems and we also consider robots with different production rates.

4.3.1 Production systems with one available cobot

Assume we have 6 workstations in an assembly production system and each serial line in the system has $n_m = 2$ workstations, $m = 1, 2, 3$. Assuming there is no independent robot, but $q_c = 1$ collaborative robot available. The baseline processing rates λ^o_w are assigned to be $\{\lambda^o_{W^1_1}, \lambda^o_{W^1_2}, \lambda^o_{W^2_1}, \lambda^o_{W^2_2}, \lambda^o_{W^3_1}, \lambda^o_{W^3_2}\} = \{12, 13.6, 15.2, 16.8, 18.4, 20\}$ respectively. The baseline collaborative robot processing change rate coefficients are the same for the 6 workstations, which means $\rho^c_{w,o} = \rho^c_{u,o}, u, w \in \mathcal{W}$. The baseline strain index ratings are identical for all the workstations, i.e. $r^o_{w,j} = r^o_{u,j}, u, w \in \mathcal{W}, j \in \mathcal{S}$, and the detailed values of strain index ratings are shown in Table 4.2. We would like to study how the optimal allocation of cobot

Table 4.2: The detailed values of baseline strain ratings, $w \in \mathcal{W}$

Parameter	$S_{w,IE}$	$S_{w,HWP}$	$r_{w,DE}$	$r_{w,SW}$	$r_{w,EM}$	$r_{w,DD}$
Value	1	2	0.2	0.05	5	5

Table 4.3: The optimal results in Case 4.3.1

$\rho^c_{w,o}$	Cobot station	Bottleneck station	Cobot working split
0.5	W^1_1	W^1_1, W^1_2	0.88
1.1	W^1_2	W^1_2	0.82
1.5	W^2_2	W^2_2	0.76

changes, when the baseline processing rate change parameter $\rho^c_{w,o}$ varies. As we discussed in Section 4.2, when $\rho^c_w = \rho^c_{w,o} - 0.5 + \theta_w$ getting larger, processing rate of workstation w will becomes slower after involving collaborative operation. We takes three values of the baseline processing rate change parameter into consideration: $\rho^c_{w,o} \in \{0.5, 1.1, 1.5\}$. The three cases correspond to three different scenarios, when the collaborative processing rate is higher, similar, and lower than the manual operation. In the simulations, optimization parameter $\eta = 1$, and SI threshold $K = 10$. The results obtained from the optimization of the three cases are shown in Table 4.3 and Figure 4.2.

When $\rho^c_{w,o} = 0.5$, we have $\rho^c_w = \rho^c_{w,o} - 0.5 + \theta_w = \theta_w \in [0.1, 0.9]$ and $\lambda^c_w = \frac{\lambda^o_w}{\rho^c_w} > \lambda^o_w$. This means the collaborative operation is always faster than the human manual operation and involving a cobot will not only improve the productivity of the system but also reduce the ergonomics of workers. Originally in the baseline system, workstation W^1_1 has the lowest processing rate parameter and is the bottleneck station. After one cobot is allocated, the bottleneck station turns out to be station W^1_2 and W^1_1. The cobot is assigned to station W^1_1 by the optimization algorithm since it has the lowest processing rate. When the cobot taking different work split, the collaborative processing rate $\lambda^c_{W^1_1} = \frac{\lambda^c_{W^1_1}}{\rho^c_{W^1_1}} \in [\frac{12}{0.9}, \frac{12}{0.1}] = [13.33, 120]$. When $\lambda^c_{W^1_1} \in [13.33, 13.6]$, the strain index of the system has no change, so the model will choose θ_w so that the collaborative processing rate is the highest, which is 13.6. When $\lambda^c_{W^1_1} > 13.6 = \lambda^o_{W^1_2}$, the bottleneck of the system is W^1_2 and the throughput of the system does not change when the work split of cobot in W^1_1 changes. However, when the work split of cobot decreases, the strain index of human operator will increase. Thus, the model set $\theta_{W^1_1} = 0.88$, resulting in the collaborative processing rate is exactly 13.6, which is the

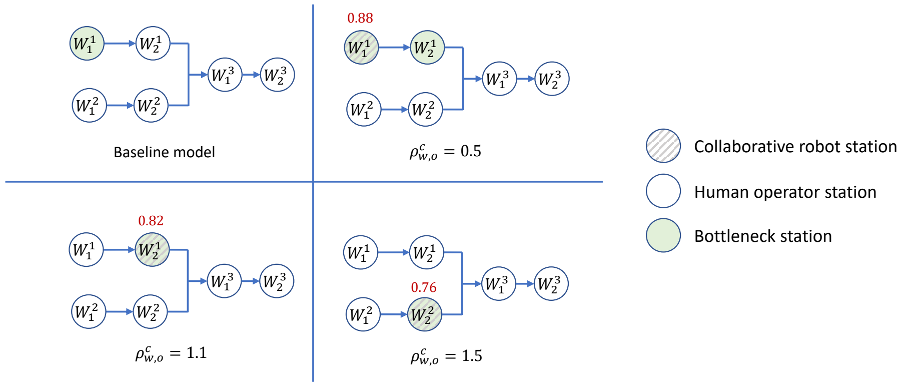


Figure 4.2: Optimal allocations in Case 4.3.1

optimal solution for the system.

When $\rho^c_{w,o} = 1.1$, similarly, we have $0.7 \leq \rho^c_w \leq 1.5$ and the processing rate may be improved or reduced after the collaborative operation is implemented. First, we can drop the allocation that the cobot is assigned to no workstation, since the replacement with collaborative operation can always achieve the same processing rate as the manual operation and keep TP the same as TP^o, meanwhile the strain index is reduced and the objective function is improved. Let assume the station cobot assigned to be $w_c, w_c \in \mathcal{W}$. When the cobot is assigned to station W_1^2, W_2^2, W_1^3, or W_2^3, the bottleneck station will be workstation W_1^1 or w_c, and we have $TP = \min(\lambda^c_{w_c}, \lambda^o_{W_1^1}) \geq \lambda^c_{w_c} = \frac{\lambda^o_{w_c}}{\rho^c_{w_c}} \geq \frac{\lambda^o_{w_c}}{1.5} \geq \frac{15.2}{1.5} = 10.13$. Then, we take a look at one of the strain factor EM. According to (4.8c), $r^c_{w,EM} = r^o_{w,EM} * \frac{TP}{TP^o} = 0.416TP \geq 0.416 * 10.13 = 4.22 \geq a^1_{w,EM} = 4$ and $S_{w,EM} \geq c^2_{w,EM} = 1$, for $w \in \mathcal{W}$. However, when the cobot is assigned to W_1^1 or W_2^1, it is possible for $r_{w,EM}$ to be less than 4 and $S_{w,EM} = c^1_{w,EM} = 0.5$. This difference in strain rating leads the optimization method to choose between W_1^1 and W_2^1 as the cobot station. After we inspect the optimal work split assignment for the two stations, we find out that the model decide the work split so that the processing rate of the station is 9.6 and $r_{w,EM}$ is exactly $a^1_{w,EM} = 4$. Given the throughput of the system and all other strain factors are the same, assigning the cobot to W_2^1 has a lower strain index rating in IE and HWP. Thus, this explains why the model assigns the cobot to W_2^1 with a work split of cobot 0.82.

When $\rho^c_{w,o} = 1.5$, we have $1.1 \leq \rho^c_w \leq 1.9$ and the collaborative operation always reduce the speed of the workstation, compared to the baseline model. Though the processing rate

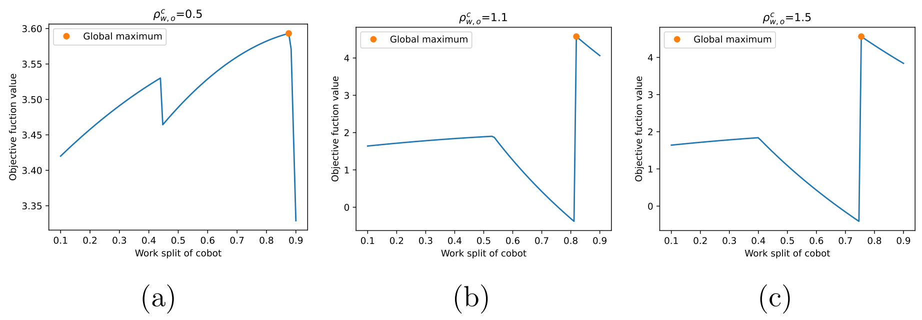


Figure 4.3: The objective function change on different work split of cobot

is reduced a little, but the strain index is reduced more, so the case that the cobot is assigned to no station is also dropped. Actually, larger values of $\rho^c_{w,o}$ are simulated and only when $\rho^c_{w,o} > 7.6$, the cobot is assigned to no station in the optimal solution. Similarly as when $\rho^c_{w,o} = 1$, when the cobot is assigned to station W^3_1 or W^3_2, $r^c_{w,EM} = r^o_{w,EM} * \frac{TP}{TP^o} = 0.416TP = 0.416\min(\lambda^c_{w_c}, \lambda^o_{W^1_1}) \geq 0.416\lambda^c_{w_c} = \frac{0.416\lambda^o_{w_c}}{\rho^c_{w_c}} \geq \frac{0.416\lambda^o_{w_c}}{1.9} \geq 0.22 * 18.6 = 4.07$, and $S_{w,EM} \geq c^2_{w,EM} = 1$, for $w \in \mathcal{W}$. Thus, the model choose the cobot station from the other four stations and the work split of cobot is determined so that the collaborative processing rate is 9.6 and $r_{w,EM}$ is exactly $a^1_{w,EM} = 4$. It also follows that assigning the cobot to W^2_2 has a lower strain index rating in IE and HWP, given the throughput of the system and all other strain factors are the same. The work split of cobot in station W^2_2 is 0.76.

We also calculate the objective function under different work split of cobot in the cobot station and the result is shown in Figure 4.3. The dot points in the figures are the global optimums of the problem. As illustrated in the figure, there are local (but not global) maximums in the objective function because the discontinuous strain index functions, but the model can locate the correct maximum and this further verifies our explanations on the optimal allocation.

4.3.2 Production systems with one available robot

In this section, we consider a different scenario where only but $q_r = 1$ robot is availble in the production system. We also have 6 workstations in total and each serial line in the system has $n_m = 2$ workstations. The baseline processing rate of the stations and the detailed stain rating parameters as case 4.3.1. The independent robot processing change rate coefficients

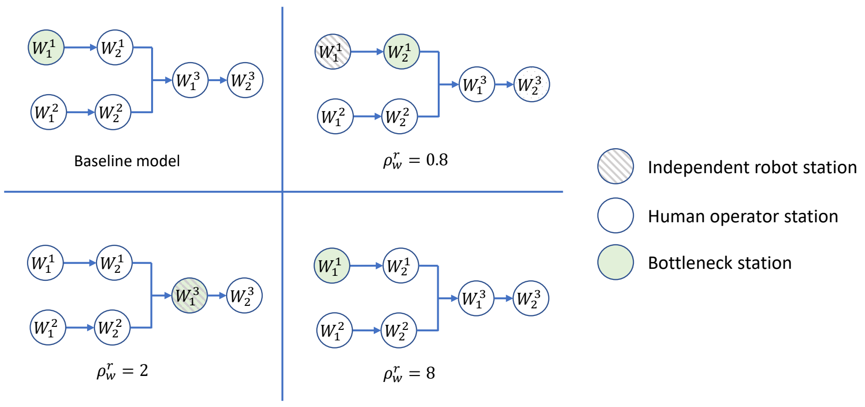


Figure 4.4: Optimal allocations in Case 4.3.2

are the same for the 6 workstations, which means $\rho_w^r = \rho_u^r, u, w \in \mathcal{W}$.

When the processing rate change parameter ρ_w^r varies, how the proposed method assign the robot to the system is discussed in this case. As shown in (4.8a), larger the ρ_w^r leads to slower processing rate of workstation w after implementing robot operation. We takes three values of ρ_w^r into consideration: $\rho_w^r \in \{0.8, 2, 8\}$. The three cases correspond to three different scenarios, when the robot processing rate is higher, lower and much slower than the manual operation. The results obtained from the optimization of the three cases are shown in Table 4.4 and Figure 4.4.

Table 4.4: The results in Case 4.3.2

ρ_w^r	Robot station	Bottleneck station
0.8	W_1^1	W_2^1
2	W_1^3	W_1^3
8	None	W_1^1

When $\rho_w^r = 0.8$, the robot operation is slightly faster than the manual operation. It is obvious that the robot will be added to the system, since it can both improve the throughput and release the strain of human. The strain index of the workers do not change a lot, when the robot is assigned to different stations. Thus, station W_1^1 is chosen because it has the smallest baseline processing rate and is the bottleneck machine in the baseline system. After the replacement, the processing rate of station W_1^1 is improved to be $\lambda_{W_1^1}^r = \frac{\lambda_{W_1^1}^o}{\rho_{W_1^1}^r} = 15 > \lambda_{W_2^1}$. In the optimal solution, W_2^1 becomes the new bottleneck workstation.

When $\rho_w^r = 2$, the robot operation has lower processing rate than the manual operation. Though it reduce the productivity of the system, the strain index may be reduced more and the robot is still assigned to the system. This result is actually surprising, since station W_1^3 is neither the station with the smallest baseline processing rate, nor the largest. We take a look at the specific scenario and we realize that the result is reasonable. Let the station the robot assigned to be $w_r \in \mathcal{W}$ and the bottleneck station be $w_b \in \mathcal{W}$. Since $12 \leq \lambda_w^o \leq 20$ and $\lambda_{w_r}^r = \frac{\lambda_{w_r}^o}{\rho_{w_r}^r} = \frac{\lambda_{w_r}^o}{2} \leq 10 < 12 \leq \lambda_w^o$, the robot station is always the new bottleneck station in the system, $w_r = w_b$. The strain index of stations is influenced by the processing rate of the robot station. For other stations remain the human operation, $\lambda_w^o = \lambda_w^h$. Let the strain index rating and value of workstations given w_b is the bottleneck station be $r_{w,j}^{h,w_r}$ and $S_{w,j}^{h,w_r}, j \in \mathcal{S}$. We take a look at the strain factor DE. Based on the formulation (4.8b), we have the DE strain factor of other workstations given w_b is the bottleneck station becomes $r_{w,DE}^{h,w_b} = r_{w,DE}^o * \frac{\lambda_w^o}{\lambda_w^h} * \frac{TP}{TP^o} = r_{w,DE}^o * \frac{\lambda_{w_b}^r}{TP^o}$, where $r_{w,DE}$ is only determined by the baseline processing rate of the bottleneck workstation. Since the strain index rating function is level function, the model tends to choose the highest processing rate station when the strain index ratings are at the same level. With the parameters specified in this case, there is a separate point between $r_{w,DE}^{h,W_1^3}$ and $r_{w,DE}^{h,W_2^3}$, which means $S_{w,DE}^{h,W_1^1} = S_{w,DE}^{h,W_2^1} = S_{w,DE}^{h,W_1^2} = S_{w,DE}^{h,W_2^2} = S_{w,DE}^{h,W_1^3} < S_{w,DE}^{h,W_2^3}$. The other strain index factors are the same under different selection of robot station. Thus, station W_1^3 is chosen because it has highest potential processing rate among the stations with the same strain index.

When $\rho_w^r = 8$, the processing rate of robot operation is much slower than the human worker operation. The robot is not assigned to any workstation, since the throughput reduction is more influential than the strain index reduction under this scenario. More values of ρ_w^r are taken into consideration in this case and we can determine that when $\rho_w^r > 7.95$, the robot will not be added to the system in the optimal solution.

4.4 Allocation with multiple cobots/robots

In this section, we consider a more general scenario where multiple cobots/robots are available in the system. We present the optimal allocation strategy of with multiple cobots/robots available in one production system and address explanations on the results. Since there is only one cobot/robot available and the number of feasible solution is restricted in 4.3.1 and Section 4.3.2, we can use straightforward simulations to find the optimal allocation of the cobot/robot. However, in this section, due to combinatorial explosion, the computational load of exhaustive search is prohibitively high with multiple cobots/robots available. To deal with the restriction, we reformulate it as a mixed integer nonlinear programming (MINP) problem and use a scalable optimization solver to solve the problem in Section 4.4.1. Later, the simulation results and the explanations are presented in Section 4.4.2

4.4.1 Computational load problem and solutions

In this case, we remain most of the settings in case 4.3.1, but we have both collaborative robot and independent robot. The baseline processing rates λ_w^o are assigned to be $\{\lambda_{W_1^1}^o, \lambda_{W_2^1}^o, \lambda_{W_1^2}^o, \lambda_{W_2^2}^o, \lambda_{W_1^3}^o, \lambda_{W_2^3}^o\} = \{12, 13.6, 15.2, 16.8, 18.4, 20\}$ respectively. The baseline collaborative robot processing change rate coefficients are the same for the 6 workstations. The baseline strain index ratings are identical for all the workstations and the detailed values of strain index ratings are shown in Table 4.2. In principle, exhaustive search can be used to find the optimal robot/cobot allocation. Because the number of possible combinations can be large in this situation and we need to find the optimal work split for cobots in each case, the exhaustive search method may not be applicable. Due to the combinatorial explosion, the computational load of exhaustive search is prohibitively high. Given the available number of cobot q_c, the available number of robot q_r and the total number of stations N, the number of feasible combinations can be estimated as $\sum\limits_{i=1,\ldots,q_r} \sum\limits_{j=N-i,\ldots,q_c} \frac{N!}{i!j!} \approx e^2 N!$, and it will increase super-exponentially. When $N = 10$, there will be around 300 millions of possible combinations. Considering the split of work also need to be determined in each combination including cobot, the computation is considerably expensive. Thus, we re-formulate the problem into a traditional MINP problem and apply a scalable optimization solver to solve it.

Since the strain index rating functions are level functions, we need to recast it as regular continuous functions with discrete inputs. The reason is that most of the traditional mixed integer nonlinear programming solver are using gradient descent method to find the optimum of the functions. Discontinuous feature can lead to inaccurate optimization and local optimal solutions. To avoid discontinuity, most of the existing optimization solver purposely do not allow the IF-THEN statement. However, some mathematical reformulation can be implemented to remove the discontinuity in our problem. The reformulation can be achieved by creating new binary variables $f^l_{w,j} \in \{0, 1\}$, $j \in \{DE, EM, SW\}$, $l = 1, 2, \ldots, n_j$ indicating the level of strain index, where n_j is the number of levels of the strain index rating. Correspondingly, we have $a^l_j, e^l_j, j \in \{DE, EM, SW\}$, $l = 1, 2, \ldots, n_j$, where a^l_j is the separating points of the level functions and e^l_j is the separating function value of the level functions. An illustration of strain index SW is shown in Figure 4.5.

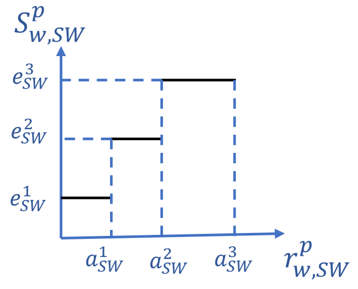


Figure 4.5: An illustration of level function reformulation

With this representation, we have $S^p_{w,j} = \sum_{l=1}^{n_l} f^l_{w,j} e^l_j$, where $\sum_{l=1}^{n_l} f_{w,j} = 1$, $\forall j \in \{DE, EM, SW\}$, $w \in \mathcal{W}$. There are new constraints need to be satisfied corresponding to the new representation. As shown in (4.8d), the latent $r^c_{w,j}$ should be in the range set up by the corresponding $f^l_{w,j}$'s. For example, when $a^1_{SW} \leq r^c_{w,SW} \leq a^2_{SW}$, this constraint is equivalent to $\sum_{l=1}^{n_l-1} f^{l+1}_{w,SW} a^l_{SW} \leq r^c_{w,SW} \leq \sum_{l=1}^{n_l} f^l_{w,SW} a^l_{SW}$, where $f^2_{w,SW} = 1$ and $f^l_{w,SW} = 0, l \neq 2$. Also, we have $S^c_{w,SW} = e^2_{SW}$. Then, we have the new formulation for the problem:

$$\underset{x^p_w, f_{w,j}, \theta_w, w \in \mathcal{W}, p \in \mathcal{P}, j \in \mathcal{S}}{\text{Maximize}} \quad E = TP - \eta \sum_{w: x^r_w = 0} SI_w$$

$$s.t. \sum_{w \in \mathcal{W}} x^r_w + x^c_w \leq q_r; \quad \sum_{w \in \mathcal{W}} x^h_w \leq q_h;$$

$$\sum_{w \in \mathcal{W}} x^r_w + x^c_w + x^h_w = |\mathcal{W}|;$$

$$TP = \min\{\lambda^p_w | x^p_w = 1, w \in \mathcal{W}\};$$

$$TP^o = \min\{\lambda^o_w | w \in \mathcal{W}\};$$

$$\mathcal{J}_{le} = \{DE, EM, SW\}; \;\; \mathcal{J}_{nl} = \{IE, HWP, DD\};$$

$$SI_w = \prod_{j \in \mathcal{J}_{le}} \sum_{l=1}^{n_l} f^l_{w,j} e^l_j \prod_{j \in \mathcal{J}_{nl}} S^p_{w,j} < K, for \;\; x^r_w = 0;$$

$$\lambda^p_w = \frac{\lambda^o_w}{\rho^p_w}, \;\; p \in \{r, c\}; \;\; \lambda^h_w = \lambda^o_w; \tag{4.8a}$$

$$S^p_{w,DD} = S^o_{w,DD}, \;\; p \in \{h, c\};$$

$$S^c_{w,j} = S^o_{w,j} * \beta_{w,j}, \;\; j \in \{IE, HWP\};$$

$$r^p_{w,DE} = r^o_{w,DE} * \frac{\lambda^o_w}{\lambda^p_w} * \frac{TP}{TP^o}, \;\; p \in \{h, c\}; \tag{4.8b}$$

$$r^p_{w,j} = r^o_{w,j} * \frac{TP}{TP^o}, \;\; p \in \{h, c\}, \;\; j \in \{EM, SW\}; \tag{4.8c}$$

$$\sum_{l=1}^{n_l - 1} f^{l+1}_{w,j} a^l_j \le r^p_{w,j} \le \sum_{l=1}^{n_l} f^l_{w,j} a^l_j, \;\; \forall j \in \mathcal{J}_{le}; \tag{4.8d}$$

$$\sum_{l=1}^{n_l} f_{w,j} = 1, \;\; \forall j \in \mathcal{J}_{le}.$$

With the reformulation, the problem is transformed to a traditional MINP (Mixed Integer Nonlinear Programming) problem. In this work, we adopt APOPT algorithm [54] in Gekko Python optimization suite [10] to solve the problem. Gekko is a python package for machine learning and optimization of mixed-integer and differential algebraic equations. APOPT is an active set solver on large-scale MINP problem and has been applied to various engineering problems, including system biology with efficient sensitivities [77], solid oxide fuel cell [60] and production scheduling decisions [9]. The effectiveness and convergence rate of APOPT is analyzed and compared with other benchmark optimization algorithms in 2012 [54]. The APOPT solver can reduce the time expense of the MINP problem in our numerical studies, compared to exhaustive search method, especially when the number of stations becomes large.

4.4.2 Results and explanations

Based on case 4.3.1 and 4.3.2, we take $\rho^r_w = 2$ and $\rho^c_{w,o} = 1.5$, which means the robot operation is slower than the cobot operation and both of them are slower than the manual operation. We select these parameters because such a case is often most difficult to decide the

Table 4.5: The results in Case 4.4

q_c	q_r	Cobot station	Robot station	Cobot working split	Bottleneck station
1	1	W_2^2	W_2^3	0.75	W_2^2
	2	W_2^2	W_1^3, W_2^3	0.83	W_2^2, W_1^3
2	1	W_2^2, W_1^3	W_2^3	$0.75, 0.9$	W_2^2
	2	W_1^2 ,W_2^2	W_1^3, W_2^3	$0.65, 0.83$	W_1^2, W_2^2, W_1^3

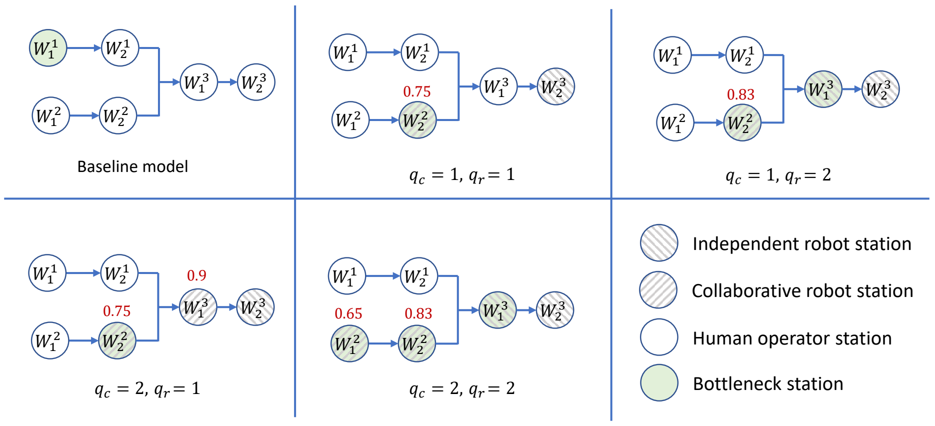


Figure 4.6: Optimal allocations in Case 4.4

allocation decision in practice because a trade-off between productivity and ergonomics needs to be made. When the available number of both kinds of robots change, the workstations the robots assigned to are shown in Table 4.5 and Figure 4.6.

When $q_r = q_c = 1$, the reason that the cobot is assigned to workstation W_2^2 is similar as when $\rho_{w,o}^c = 1.5$ in case 4.3.1. The model tends to set the bottleneck processing rate as 9.6, which is a separate point in the strain index function of factor EM. Also, this explains why the robot is assigned to station W_2^3, which has the highest baseline processing rate. If the robot is assigned to other stations, the throughput of the system would be smaller, but the strain indices of human involved stations would remain at the same level or have small difference.

When $q_r = 2, q_c = 1$, knowing that there is a stain index separate point at system processing rate of 9.6, we would like to find the highest possible throughput of the system and we assign the two robots to the stations with the highest baseline processing rate, which are W_2^3 and W_1^3. However, different from when $q_r = 1$, the bottleneck station becomes one of the robot station, the robot processing rate of workstation W_1^3 is $\lambda^r_{W_1^3} = \frac{\lambda^o_{W_1^3}}{\rho^r_{W_1^3}} = \frac{18.4}{2} = 9.2$, which is smaller than 9.6. This fact in turn changes the optimal work split in workstation W_2^2. The optimal processing rate of W_2^2 turns out to be 9.2 and the work split of cobot is 0.83 accordingly. There is no need to increase the strain rating of IE and HWP, given that the throughput of the system is 9.2 and can not be improved.

When $q_r = 1, q_c = 2$, the robot is assigned to station W_2^3, since it has the highest baseline processing rate and robot operation will slow down the speed by coefficient $\rho^r_w = 2$. Assigning the robot to other stations would reduce the throughput of the whole system. Then, the two cobots are assigned to stations with the highest baseline processing rate among the left stations. Upon the previous discussion on when $q_r = q_c = 1$, we know that processing rate of 9.6 is a separate point for strain factor EM. Thus, the two cobot stations tend to choose a work split that can make the cobot processing rate as close as possible to 9.6. Then, station W_2^2 use a cobot work split $\theta_{W_2^2} = 0.75$. In station W_1^3, the most work split of cobot 0.9 will be applied, since the cobot processing rate of W_1^3 is $\lambda^c_{W_1^3} = \frac{\lambda^o_{W_1^3}}{\rho^c_{W_1^3}} \geq \frac{18.4}{1.9} = 9.68 > 9.6$ and the closest processing rate is achieved when $\theta_{W_1^3} = 0.9$.

When $q_r = q_c = 2$, the two robots are assigned to the stations with the highest baseline processing rates, which are W_1^3 and W_2^3. The bottleneck of the system becomes station W_1^3 with a robot processing rate $\lambda^r_{W_1^3} = 9.2$. Intuitively, the cobots are also assigned to the stations with high baseline processing rate, so that the throughput of the system will not be reduced. In this case, the strain index separating points do not influence the results of assignments and the cobots are assigned to station W_2^2 and W_1^2. The cobot work splits are determined as 0.83 and 0.65 respectively, so that the processing rates of the two cobots station are aligned with the bottleneck processing rate of the new system.

The objective function under different work splits of cobot in the cobot stations are shown in Figure 4.7. Similarly, in the top two plots, there is only one available cobot and the dot point in the top two the figures are the global optimums of the problem. The one available case also has a local (but not global) maximum in the objective function, while the case with two available robots has no local maximum. In the bottom two plots, there are two available cobots and the x-axis and the y-axis are corresponding to the work split of cobot in each station. The color of the dots presents the value of the integrated objective function

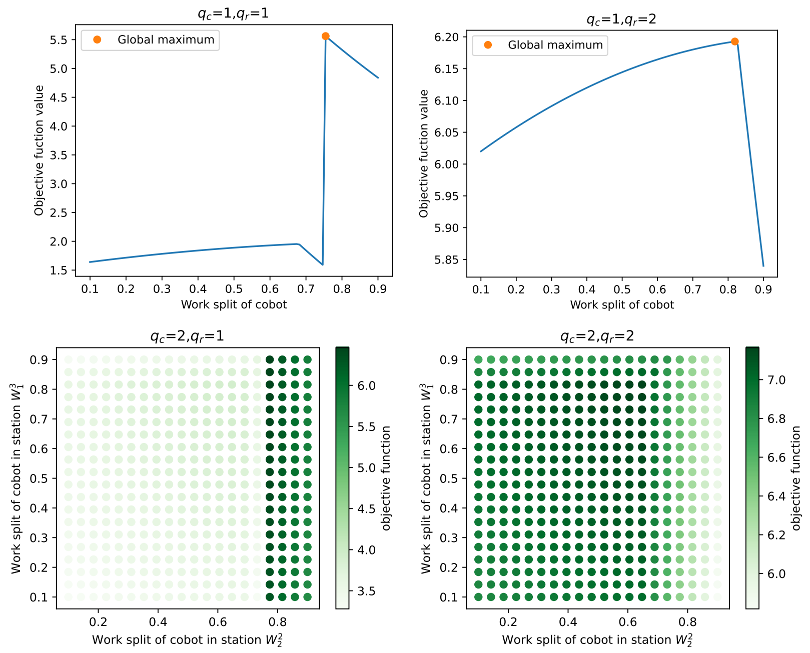


Figure 4.7: The objective function value v.s. the work splits of cobot stations

and the darker the dot, the larger the objective function. In the left plot, the case with one available robot, there is a gap between work split higher and lower than 0.75 in workstation W_2^2, which infers that the work split of the first workstation makes a larger difference in the allocation. However, in the right bottom plot, the case with two available robots, the work split of both cobot stations influence the allocation similarly.

4.5 Discussion

In this section, we discuss some insights we obtain from the results of the numerical problems in Section 4.3 and Section 4.4. Though some parameters are fixed in the study, we can still observe some patterns in the results. We anticipate that these insights can be helpful for allocating cobots/robots.

First, it is not always better to apply all available cobot/robot resources to the system. When the cobot or robot operation is much slower than the human operation, the integrated performance of the whole system will not be improved after the replacement. As mentioned in Section 4.3.1, when the cobot operation is 7.6 times slower than the human operation, involving a cobot is not beneficial for the overall performance. Similar situations happen to the robot allocation case in Section 4.3.2. In the real world, when the tasks require high level human intelligence, the cobot/robot may need much longer time while human workers take barely take any time to finish the task. In these cases, we suggest not to replace the human workers and remain the original operations.

Second, the optimal allocation of cobots/robots is not always the slowest or the fastest station in the system. As discussed in Section 4.3.2, the bottleneck station of the system is always playing an important role in allocating the robot. However, the model does not allocate the robot station to the bottleneck station of the baseline system all the time. As shown in Figure 4.4, $\rho_w^r = 2$, station W_1^1 is the bottleneck station in the baseline system, but another station is replaced by the robot after allocation. We have shown that the result is reasonable and the detailed illustrations are in Section 4.3.2. In a word, one strain factor of the workstations other than the new bottleneck station is determined by the baseline processing rate of the new bottleneck workstation after some simplifications of the formulation. The model tends to choose the highest processing rate station when the strain index ratings are at the same level, and the robot station is chosen because it has highest potential processing rate among the stations with the same strain index. This result suggests us consider the replacement of all the stations, because we can not determine what kind of allocation of the cobot/robot could improve the system performance the most.

Third, under the same operation alternative allocation, the optimal work split between human and cobot in the cobot operation station depends on both the bottleneck station after the allocation and the separation points of the strain index level functions. As shown in Figure 4.8, there are always local (but not global) maximums of the integrated objective

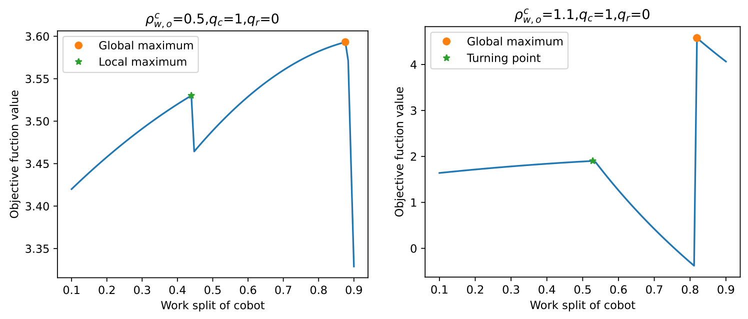


Figure 4.8: Illustration of the influential factors of the objective function

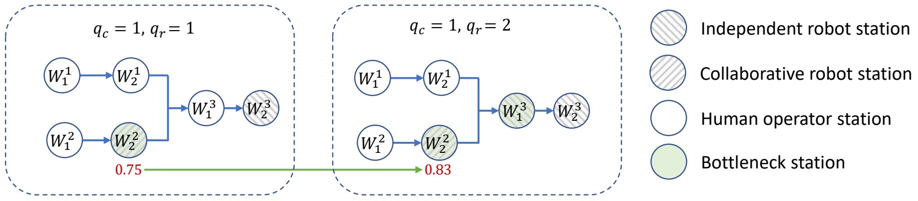


Figure 4.9: Illustration of correlation between cobot and robot allocation

function against the change of different work split of cobot. These local maximums of the objective functions are corresponding to the separation points of the strain index level functions. Once the level of strain index decreases, there is always a sudden jump in the value of the objective function, since there is a sudden fall in the quantified ergonomics performance. Besides the sudden changes in the objective function, there are also some turning points in the objective function. These turning points are caused by the change of the bottleneck station of the system. When the bottleneck station changes, the overall throughput or productivity performance is influenced due to the increment of the work split of cobot. These changes are gradually, so we do not observe sudden changes in the objective function. These observations inform us that we need to pay attention to not only the extreme cases in one specific station, but also the overall balance of all the workstations.

Forth, in the optimally allocated systems, not only is the overall ergonomics measure SI_{all} reduced, the SI in the most stressful workstation also decreases, compared to the baseline system. Take the one available cobot case in Section 4.3.1 as an example. Keeping most of the settings, let $r_{W_1^1,DE} = 0.7$, $r_{W_2^1,DE} - 0.6$, $r_{W_1^1,SW} = 1$, $r_{W_2^1,SW} = 1.5$, $r_{W_1^1,EM} = 16$, $r_{W_2^1,EM} = 15$, and $\rho^c_{w,o} = 1.1$. The overall ergonomics measure for the entire baseline system $SI_{all} = 36$. After the cobot was applied to the system, SI_{all} became 10.83, which was highly reduced from the baseline system. Also, with the improvement of one collaborative robot, the SI in the most stressful workstation changed from 16 to 6.75, which is not hazardous anymore by the definition in [40]. This observation further confirms that the definition of overall ergonomics measure and the design of threshold parameter K is reasonable in the formulation.

Last, the priority of allocating cobots and robots depends on the properties of the available cobots/robots and the baseline parameters of the production systems. We are not sure if the cobots or the robots would make larger differences to the system before we run the model. Also, the allocation of different cobots/robots may influence each other, when there are multiple cobots/robots available in the system. As shown in Figure 4.9, the optimal allocation and work split of cobot is related to the allocation of robots in the system. When there is one robot and one cobot available, the optimal work split of the cobot station 0.75, and it becomes 0.83 when there is one more robot. Later, when there are two cobots and one robot, the cobots are assigned to station W_2^2 and W_1^3, and the cobots stations are changed to W_1^2 and W_2^2 when there is one more robot. This indicates that the allocation of multiple cobots/robots are highly correlated with each other, and we need to consider all the possible scenarios to find the optimal allocation.

4.6 Conclusion

In this paper, we propose a method to optimally allocate cobot/robot to a manufacturing system with multiple workstations. This work fills the research gap of system level job allocation between human operators and collaborative or independent robots. We use a integrated objective function to consider the tradeoff between system throughput of the ergonomics level of the human workers in the system. Based on the objective function, we formulate an optimization problem for the cobot/robot allocation problem. With the problem formulation, we solve the allocation of cobot/robot in production systems with one available cobot/robot and the insights of the allocation are illustrated. Then, to solve the allocation of the production systems with multiple cobots/robots available, we revise the formulation to address the challenge caused by the discontinuous objective function by introducing new binary variables so that the problem can be put into a conventional mixed integer nonlinear optimization problem. The APOPT solver in Gekko optimization suite is utilized to solve the optimization problem in an efficient way. The solutions of the allocations are explained and confirmed by the domain knowledge. Later, we also conclude some insights for the allocation of cobot/robot resources.

In the future, more general cases considering buffer and starvation of the production systems will be studied, which can extend the application of the proposed optimization method. The allocation of buffer size in the workstations can also be taken into consideration and provide the companies with suggestions on setting up a production line. We will pursue these directions and report the findings in the future.

Chapter 5

Collaborative Production System Design with Ergonomics Concerns and Precedence Constraints

Abstract

Robots/Cobots have shown promise for significant improvement in the well-being of human workers and increment in productivity for manufacturing production systems. In this paper, we investigate and study the optimal design of a serial production system with fixed cycle time, considering the ergonomics constraints of human workers. Previous research on resource allocation in collaborative systems focuses on the task level and omits the optimal assignment of tasks. Taking the precedence relationship of tasks into consideration and the physical exposure of human workers as constraints, we formulate and allocate the robot/cobot resources to a production system to maximize productivity and minimize the economic cost of the system. We proposed to solve the combinatorial optimization problem with a reinforcement learning guided evolutionary algorithm. Simulations and real-world scenario based case studies are carried out and useful insights are provided for future prac-

This chapter is based on the paper: **Huang, C.**, Konstant, A., Zhou, S., Li, J. & Radwin, R. Collaborative Production System Design with Ergonomics Concerns and Precedence Constraints, to be submitted to IEEE Transactions on Automation Science and Engineering.

titioners.

5.1 Introduction

Allocation of the collaborative robot (so-called cobot) and independent robot in manufacturing systems have shown significant improvement in productivity and a notable reduction in the physical strain of human workers [100]. Compared to independent robots, collaborative robots have fewer safety requirements and can work together with human workers, rather than in an enclosed working space. With the cobots working alongside, the human workers can save effort on a wide range of tasks and reduce physical exposure during work [30]. Stressful tasks such as assembly, screwing, and heavy lifting can be assigned to or shared with collaborative robots and the strain of human workers can be reduced [47]. In the meantime, with the assistance of collaborative robots, the productive performance of the entire production system can be improved. Based on the Collaborative Robots Market Report in 2021 [1], the collaborative robot market share is predicted to grow substantially in the coming years. It can be foreseen that the human-robot collaborative manufacturing module will continuously contribute to the future advanced manufacturing and factory automation, considering the profound benefit and the rapid development of robot and cobot technology.

Task allocation in a collaborative production system belongs to line balancing problems when there are multiple workstations in the production line. Line balancing is a crucial aspect of production optimization. It ensures that each workstation has a comparable workload, smooths the streamline operations and improves the system productivity. Effective line balancing requires a comprehensive analysis of the production processes and appropriate measurements of factors such as task processing time, physical strain of human workers, and precedence relationship between tasks. As shown in Table 5.1, in recent years, there have been an increasing number of research and implementation studies in line balancing and task allocation problems in production systems [8, 20, 80, 81, 146]. Most of the these take the production cost and productivity as the overall objective of the optimization problem. Some handle the case that there are multiple workstations in the production system [20, 81, 146], and some consider the precedence relationship as a constraint in the allocation [80, 81, 146].

Table 5.1: State of art task allocation works in production system

References	**Robot/cobot**	**Production**	**Ergonomics**	**Precedence**	**Multiple workstations**
Chen [20]		✓			✓
Liu et al. [81]; Zhang et al. [146]		✓		✓	✓
Liu et al. [80]		✓		✓	
Arkat et al. [8]		✓			
Maganha et al. [85]; Sarker et al. [114]; Takata and Hirano [126]; Fechter et al. [34]; Tsarouchi et al. [129]; Ranz et al. [109]; Choi et al. [23]; Jose and Pratihar [63]; Chen et al. [19]	✓	✓			
Mossa et al. [97]		✓	✓		
Ou et al. [102]; Heydaryan et al. [56]; Liu et al. [83]	✓	✓	✓		
Mokhtarzadeh et al. [95]; Huang et al. [58]; Dalle Mura and Dini [26]	✓	✓	✓		✓
Hu and Chen [57]	✓		✓	✓	
Müller et al. [98]; Rahman and Wang [108]; Faccio et al. [33]; Pearce et al. [103]	✓	✓	✓	✓	
Gjeldum et al. [42]	✓	✓			✓
Michalos et al. [92]; Boschetti et al. [12]	✓	✓		✓	
Weckenborg et al. [140]	✓	✓		✓	✓
Proposed framework	✓	✓	✓	✓	✓

Some recent research tried to make the collaboration between human workers and cobot/robot effective at the task level [35, 41, 70, 106, 119], focusing on the design and programming of robots. In the meantime, with the rapid development of the applications of cobots and robots, researchers have put tremendous effort into the allocation of robots in workstations to optimize the system productivity [19, 23, 34, 63, 85, 109, 114, 126, 129]. When implementing the cobot/robots in production systems, they focus more on improving productivity and reducing the production cost than reducing the physical strain of human workers. In addition to the productivity performance of the system, ergonomics factors are also an important concern. Ergonomics concerns, which is one of the motivations for involving robots in manufacturing operations, includes high forces, repetitive motions, awkward postures, and exposure to vibration from equipment and tools. Numerous studies have been implemented to assess and improve ergonomics in manual operations [4, 132, 134, 145]. Some of the existing task allocation research in production systems consider the ergonomics factors and the physical strain of human workers [26], but the influence of processing time from collaborative operations is not included in the model and some of them did not consider the applications of cobot/robot. Some implementations of cobots/robots consider the influence on ergonomics factors of human workers but they omit the precedence relationship between different tasks [26, 56, 58, 83, 95, 102]. Other studies analyze the precedence relationship during task allocation, but they are limited to one workstation [33, 57, 98, 103, 108]. On the other hand, some, including the precedence relationship or multiple workstations, do not consider the ergonomics impact on the production system [12, 42, 92, 140]. Among all the research studies, some of them assign resources to a combination of tasks, while some assign resources to each task that can not be further split up. Due to the limitation of robots' capability, deployment of cobot/robot to an unsplittable task is a more realistic task allocation strategy. In some ergonomics studies, the unsplittable task is also called 'subtask' and the 'task' term refers to a sequence of subtasks operated in the same work cycle [39]. To avoid misunderstanding, the term 'task' is referred to the unsplittable task in the later part of this article. In this work, we comprehensively formulate the task allocation problem in the production systems with cobots/robots by considering the production cost, ergonomics of human workers, precedence relationship between tasks, and the line balancing between different workstations.

With the formulation, the allocation task becomes a combinatorial optimization problem with a complex objective function and a large number of constraints. Using traditional optimization algorithms to solve the problem from the mathematical formulation is almost impossible. When the number of tasks is large, the computational time will also dramatically increase. Instead of traditional optimization methods, we can adopt a heuristic algorithm to solve the combinatorial optimization problem. There are some advantages and benefits of using a heuristic optimization algorithm. First of all, since the solution is generated from an iterative approach and the number of visited solutions keeps increasing, it is guaranteed that the global optimal solution can be found after a large enough number of iterations. Also, the infeasible solutions will not be kept in the possible solution set and this makes sure the derived solutions are feasible considering all the constraints. Furthermore, if there is uncertainty in the production system, or when there are real-time data involved, the data-driven heuristic algorithm can be adapted to the stochastic settings easily.

In this article, we propose an optimization framework to optimally allocate a limited number of robot/cobot resources to a manufacturing system to reduce the cost caused by transformation, considering the ergonomics factors of human workers, and the precedence relationships between tasks. Specifically, we propose a heuristic algorithm to solve the following allocation problem: Given a human-based manufacturing production system with multiple workstations and a limited number of robots/cobots, what is the best allocation of these robots/cobots to the selected workstations and tasks so that an integrated measure considering both system throughput and economic cost is optimized, taking the precedence relationships between tasks and the ergonomics factors as constraints? The main contributions of the proposed work are the following:

- Production systems with multiple workstations are considered and the cumulative ergonomics factors are estimated based on the tasks assigned to the human worker in the same workstation.

- The precedence relationships between different tasks are considered and the optimal allocation can meet the requirement of precedence and ensure the workflow of the entire production system.

- An optimization framework of the allocation problem is established and a reinforcement learning guided evolutionary algorithm is proposed to solve the optimization problem in an effective and scalable manner.

The remaining article is organized as follows. In Section 5.2, the problem of allocating robots/cobots to a production system is described and the framework of the optimization problem is introduced. In Section 5.3, a combinatorial representation of the solution and the proposed reinforcement learning guided evolutionary algorithm are illustrated. In Section 5.4, real-world senario based cases are implemented and useful insights are provided for future allocation. In Section 5.5, conclusions are summarized and future works are discussed.

5.2 Problem formulation

In this section, we introduce the problem settings, the ergonomics evaluation and the optimization formulation of the task allocation problem in a collaborative production system. In Section 5.2.1, the assumptions of the system are provided and the variables used in the optimization problems are defined. In Section 5.2.2, the ergonomic strain of human workers is evaluated from task level to work shift level. Then, in Section 5.2.3, the ergonomic change with different operation alternatives is analyzed, and the overall optimization problem is established.

5.2.1 System description

In this study, we consider a serial production system. The stochastic model to describe the operations of such systems is defined as follows:

- In the system, there is a serial line of workstations and the workpieces flow through the stations from left to right. There are N workstations, where the u th station is denoted as w^u. The set of all workstations is represented by $\mathcal{W}$.
- There is infinite buffer capacity between each pair of stations in the system.
- There are n tasks that need to be finished in the system. Let t^i represent the ith task in the system, and $\mathcal{T} = \{t^i | i = 1, \dots, n\}$ represents the set of tasks, where n is the number of tasks in the system. Let n_u be the number of tasks allocated in workstation w^u and $\mathcal{T}_u = \{t_u^l | l = 1, \dots, n_u\} \subset \mathcal{T}$ be the set of tasks allocated in workstation $w^u, w^u \in \mathcal{W}$.

- Each task can be operated by a human worker (h), a robot (r), or through a collaborative operation between human and cobot(c). The job processing time at workstation w is assumed to follow an exponential distribution with parameter λ_p^i, $p \in \mathcal{P}$, where $\mathcal{P}= \{h, r, c\}$ is the set of operation alternatives. If task t^i cannot be carried out by an alternative p, λ_p^i is set as $\eta \to 0$.

- The cobot and robot will be dedicated to one task. There is at most one human worker in one workstation.

- We assume all tasks in the production system are operated by human workers initially. Then q_r robots and q_c cobots will be assigned to replace $q_r + q_c$ human-operated tasks in the system. When a robot is assigned to a task, the ergonomic strain index for that task will become zero as no human operator is involved. When a cobot is assigned to a task, the ergonomic strain index will be reduced for the task.

- Let $\mathcal{D} = \{d^{i,j} | i, j = 1, \ldots, n, i \neq j\}$ be the set of precedence relationship between tasks, where $d^{i,j}$ is the precedence relation from task t^j to task t^i. If t^i can not be proceeded without the completment of t^j, we have $d^{i,j} = 1$, otherwise $d^{i,j} = 0$.

- The system has a fixed cycle time CT for all the workstations in the system, which has been decided by the real practice of the system and can not be changed.

- For a workstation with human or cobot operation in the system, there is one individual human worker who handles all the human work.

Since the cycle time is fixed in the system, the productivity of the entire system is fixed and we only need to optimize the ergonomics of the system when there are a given number of available robots and cobots. The number of workstations can be changed based on the processing time change of new operation alternatives, but new workstations with human or collaborative operations will need more human workers and increase the overall ergonomics of the entire system. The objective of the optimization problem is to minimize the cost of production operations and the set-up cost of workstations. $C_p, p \in \{h, c, r\}$ is defined as the implementation and operation cost of alternative p and C_w is the set-up cost of one workstation.

To formulate the robot/cobot allocation problem, we introduce $\mathcal{X} = \{x^i_{u,p} \in \{0,1\} | i = 1, \dots, n, w^u \in \mathcal{W}, p \in \mathcal{P}\}$ as the set of allocation indicators $x^i_{u,p}$. If task t^i is allocated to use operation p in workstation w^u, $x^i_{u,p} = 1$, otherwise $x^i_{u,p} = 0$. Then, we have $x^i_u = \sum\limits_{p \in \mathcal{P}} x^i_{u,p} \in \{0,1\}$ is the indicator if task t^i is allocated in workstation w^u. In addition, $\sum\limits_{p \in \mathcal{P}} x^i_{u,p} = 1$, for $w^u \in \mathcal{W}$, $i = 1, \dots, n$.

Besides the task allocation for different workstations and operation alternatives, we also need to consider the time constraints for tasks allocated in the same workstation.

- Let $ST^i_{u,p}$ and $FT^i_{u,p}$, represent the start time and finish time of task t^i under operation alternative p in workstation w^u. Let ET^i_p be the estimated time taken of task t^i under operation p. Since one task can only start after all its precedent tasks have been finished we have $ST^i_{u,p} = \max\limits_{i:d^{i,j}=1} \sum\limits_{p' \in \mathcal{P}} x^j_{u,p'} FT^j_{u,p'}$. With no transition time, we have the finish time of task t^i is the sum of start time and the estimated time taken: $FT^i_{u,p} = ST^i_{u,p} + ET^i_p = ST^i_{u,p} + 1/\lambda^i_p$.
- All the tasks should be finished in the cycle time of the system and we have $FT_u \leq CT, w^u \in \mathcal{W}$, where $FT_u = \max\limits_{i:x^i_{u,p}=1} FT^i_{u,p}$
- Since there is only one human worker in each workstation, the operation time of human-involved tasks in the same workstation should not overlap with each other. We have $ST^i_{u,p} \geq FT^j_{u,p'}$ or $ST^j_{u,p'} \geq FT^i_{u,p}$ for $x^i_{u,p} = 1, x^j_{u,p'} = 1, p, p' \in \{h, c\}, w^u \in \mathcal{W}$.

5.2.2 Ergonomics Evaluation in the Collaborative System

In this section, we illustrate the ergonomics evaluation in the production system with cobots/robots. The physical strain factor of the human workers are characterized at task level in Section 5.2.2.1, the task cycle level in Section 5.2.2.2, and the work shift level in Section 5.2.2.3.

5.2.2.1 Subtask Level

In 2017, Garg *et al.*proposed the Revised Strain Index (RSI) [40], which is a Distal Upper Extremity (DUE) physical exposure assessment model improved from the 1995 Strain Index

[124]. The 1995 strain index model was first proposed in [124] and has been verified to be a reliable and effective way to evaluate the risk level related to hand workload over time [27, 110]. Based on assumption M4 in [145], the strain index of a manual operation can be evaluated by six risk factors of the overall process. The definitions of the six risk factors are following.

- Intensity of exertion: An estimate of the strength required to perform the task on time.
- Duration of exertion: The physiological and biomechanical stresses related to how long an exertion is maintained.
- Efforts per minute: The number of exertions per minute, which is synonymous with the frequency.
- Hand/wrist posture: An estimate of the position of the hand or wrist relative to a neutral position.
- Speed of work: An estimate of how fast the worker is working.
- Duration per day: A measure obtained from plant personnel.

For each task and each hand, all risk factors are evaluated and assigned to one of five categories, each corresponding to a rating based on a scale of $1-5$ as well as a multiplier value. The strain index of each workstation is the product of the six ratings of the workstation. Then, the score is compared to the multiplier to identify the level of task risk. There are two drawbacks of the categorical rating in the 1995 strain index model. First, the categorical rating can not fully characterize the information of the numerical factors. The difference between tasks with the same strain level but different strain ratings can not be captured. Second, the level function in the model is hard to adapt to later optimization formulation and algorithms. Thus, the 1995 strain index model is modified and developed into the Revised Strain Index (RSI) model. RSI uses continuous rather than categorical multipliers and replaces the duty cycle with duration per exertion. Based on a simulation of 13,944 tasks, RSI showed good estimations of the risk levels and is useful for DUE task analysis, intervention, and design. In this work, we use the RSI model to evaluate the ergonomic strain of human workers and the exact definition is illustrated as follows.

The RSI score is defined as the product of five components, expressed in (5.1)

$$RSI = M_I(I) \cdot M_E(E) \cdot M_D(D) \cdot M_P(P) \cdot M_H(H) \tag{5.1}$$

In the equation,

$$M_I(I) = \begin{cases} 30I^3 - 15.6I^2 + 0.4, 0 < I \leq 0.4 \\ 36I^3 - 33.3I^2 + 24.77I - 1.86, 0.4 < I \leq 1 \end{cases} \tag{5.2}$$

$$M_E(E) = \begin{cases} 0.1 + 0.25E, E \leq 90/m \\ 0.00334E^{1.96}, E > 90/m \end{cases} \tag{5.3}$$

$$M_D(D) = \begin{cases} 0.45 + 0.31D, D \leq 60s \\ 19.17lnD - 58.44, D > 60s \end{cases} \tag{5.4}$$

$$M_P(P) = \begin{cases} 1.2e^{-0.009P} - 0.2, P < 0 \\ 1, 0 \leq P \leq 30 \\ 1 + 0.00028(P-30)^2, P > 30 \end{cases} \tag{5.5}$$

$$M_H(H) = \begin{cases} 0.2, H \leq 0.05h \\ 0.042H + 0.09lnH + 0.477, H > 0.05h \end{cases} \tag{5.6}$$

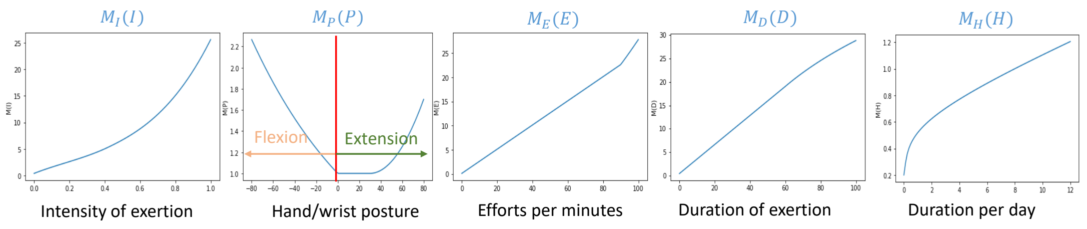


Figure 5.1: Illustration of the multiplier functions

Here, M_S is the multiplier function of ergonomics factor $S, S \in \{I, E, D, P, H\}$. In Figure

5.1, the pattern of the multiplier functions are illustrated. I corresponds to the intensity of exertion (force) (% MVC expressed numerically from 0 to 1.0, or Borg CR-10 rating divided by 10.0). E is corresponding to the exertions per minute (frequency). D corresponds to the duration per exertion by seconds. P is the hand/wrist posture by degrees from the anatomically neutral position. A negative degree means flexion and a positive degree means extension. H is the duration of tasks per day by hours.

The RSI is designed so that a score smaller or equal to 10 is considered 'safe' and a score larger than 10 is considered 'hazardous'. Based on some numerical comparison of the 1995 SI and the RSI, the RSI had much greater discrimination between 'safe' and 'hazardous' tasks for various combinations of force, repetition, and duty cycle. Thus, it is believed to substantially improve the strain index model. Also, the RSI has continuous properties and promotes the convergence of the optimization model for resource allocation.

5.2.2.2 Work Cycle Level

The RSI can be used to define the biomechanical stressors from an unsplittable task. In a work cycle, there can be a sequence of one or more tasks that are repeated and are performed for a certain duration in a work shift [64]. A job consists of one or more tasks performed during a work shift. The Composite Strain Index (COSI) was proposed by Garg *et al.*[39] to capture the biomechanical stressors from a task with two or more tasks. For a task consisting of a single task, COSI is the RSI of the single task.

The COSI is calculated in the following procedure:

- Calculate the RSI for each task and arrange them in a descending order s.t. $RSI_1 \geq RSI_2 \geq RSI_3 \geq \cdots \geq RSI_n$, where n is the number of tasks in the task.

- Derive the COSI score, the COSI score is RSI_1 (peak exposure task) plus an incremental increase in physical exposure of the human worker, ΔRSI, as each subsequent task is added to the peak task: $COSI = RSI_1 + \sum_2^n \Delta RSI_i$. Here, RSI_1 is the peak exposure task which is the task with the highest RSI, and ΔRSI_i is the incremental increase in physical exposure associated with each of the remaining tasks, in the order determined by the ranking of RSI.

- To calculate the incremental increase in physical exposure, we first determine the Frequency-Independent RSI (FIRSI): $FIRSI_i = RSI_i/M_E(E_i), i = 2, \dots, n$. Then, we calculate the incremental efforts per minute multiplier (ΔEM) associated with each task: $\Delta EM_i = M_E(\sum_1^i E_j) - M_E(\sum_1^{i-1} E_j)$, where E_j is the exertions per minute of the jth task. FIRSI omits the frequency of the task and the frequency-independent strain index is a traditional method to measure the incremental physical exposure [139]. Conceptually, ΔEM_i is the differential increase in frequency multiplier when the ith task's frequency is added to the cumulative frequency of the prior tasks, which have higher RSI than the ith task. The ΔRSI of the ith task is the Frequency-Independent RSI of the ith task $FIRSI_i$ multiplied by the incremental efforts per minute multiplier of the ith task (ΔEM_i): $\Delta RSI_i = FIRSI_i \times \Delta EM_i$.

5.2.2.3 Work Shift Level

With the COSI of each task, we can calculate the Cumulative Strain Index (CUSI) of the work shift. The calculation follows three steps:

- Rank the COSI for each task in descending order, s.t. $COSI_1 \geq COSI_2 \geq COSI_3 \geq \cdots \geq COSI_m$, where m is the number of tasks performed in a work shift.

- Derive the CUSI score, the CUSI score is $COSI_1$, which is the peak exposure task's COSI, plus the incremental increase in physical exposure as each subsequent task is added to the peak task: $CUSI = COSI_1 + \sum_2^m \Delta COSI_k$. Here, $\Delta COSI_k$ is the incremental increase in physical exposure associated with the k th task.

- $\Delta COSI_k$ is calculated by the Hours-Independent COSI (HICOSI) of the kth task, defined by: $HICOSI_k = COSI_k/M_H(H_k)$. $M_H(H_k)$ is the hours per day multiplier for the kth task and is calculated by: $\Delta HM_k = M_H(\sum_1^k H_j) - M_H(\sum_1^{k-1} H_j)$, where H_j is the hours per day for task j. The HICOSI ignores the hours per day that the task is performed and ΔHM_k is the differential increase in the hours per day multiplier when the kth task's hours per day is added to the cumulative hours per day of the prior tasks.

According to Garg *et al.*[39], a COSI and CUSI score smaller than or equal to 10.0 is considered to be safe and a score higher than 10.0 is considered to be hazardous or risky. A threshold K is set to avoid a high strain index at a specific workstation. Thus, the following constraint exists:

$$CUSI_u < K, \quad \forall w^u \in \mathcal{W}. \tag{5.7}$$

5.2.3 Ergonomic Change and Optimization Framework

In the baseline model, the strain index and the processing rate parameter of the tasks under human operation are known. Then, based on the following assumptions, the change of processing rate and strain index from the baseline system can be defined if a cobot or robot is allocated to task t^i in workstation w:

- The processing rate of task t^i is changed by coefficient ρ_p^i , $p \in \{r, c\}$. Then we have $\lambda_p^i = \frac{\lambda_h^i}{\rho_p^i}$.
- ergonomics factor H (duration of task per day) of the human worker for task t^i remains the same as in the baseline system so that $H_i = H_i^h$. This is because the human worker in the collaborative operation still follows the general working schedule and the work durations per day are the same under manual and collaborative alternatives.
- Risk factors I (intensity of exertion) and P (hand/wrist posture) of task t^i are not dependent on processing time and their changes are defined by coefficients $\beta_I^i \in (0, 1)$ and β_P^i, thus $S_c^i = S_h^i \beta_S^i, S \in \{I, P\}$.
- Factor D, the duration of the exertion is only changed based on the human operation time on the task t^i. The original D factor for each task is D_h^i. We have the duration of exertion after allocation as $D_c^i = D_h^i \rho_c^i$.
- Efforts per minute E of a task t^i is proportional to the operation rate. $E_c^i = E_h^i / \rho_c^i$.

Then, a mathematical formulation to optimize the integrated performance measure of the system can be expressed as

$$\begin{aligned}
&\underset{\mathcal{X},ST}{\text{Minimize}} \sum_{u,i} x^i_{u,r}C_r + \sum_{u,i} x^i_{u,c}C_c + \\
&\sum_{u,i} x^i_{u,h}C_h + NC_w \\
s.t.\quad &\sum_{t^i\in\mathcal{T}}\sum_{w^u\in\mathcal{W}} x^i_{u,r} \le q_r; \quad \sum_{t^i\in\mathcal{T}}\sum_{w^u\in\mathcal{W}} x^i_{u,c} \le q_c; && (5.8a)\\
&\sum_{p\in\mathcal{P}} x^i_{u,p} = 1,\ \mathit{for}\, w^u \in \mathcal{W},\ i = 1,\dots,n; && (5.8b)\\
&\sum_{w^u\in\mathcal{W}}\sum_{p\in\mathcal{P}} x^i_{u,p} = 1,\ \mathit{for}\, i = 1,\dots,n; && (5.8c)\\
&ST^i_{u,p} = \max_{i:d^{i,j}=1} \sum_{p'\in\mathcal{P}} x^j_{u,p'}FT^j_{u,p'}, \\
&\mathit{for}\, w^u \in \mathcal{W},\ i = 1,\dots,n,\ p \in \mathcal{P}; && (5.8d)\\
&FT^i_{u,p} = ST^i_{u,p} + 1/\lambda^i_p, \\
&\mathit{for}\, w^u \in \mathcal{W},\ i = 1,\dots,n,\ p \in \mathcal{P}; && (5.8e)\\
&ST^i_{u,p} \ge FT^j_{u,p'}\ \mathit{or}\, ST^j_{u,p'} \ge FT^i_{u,p}, \\
&\mathit{for}\, x^i_{u,p} = 1,\ x^j_{u,p'} = 1,\ p,p' \in \{h,c\},\ w^u \in \mathcal{W} && (5.8f)\\
&FT_u = \max_{i:x^i_{u,p}=1} FT^i_{u,p} \le CT,\ w^u \in \mathcal{W} && (5.8g)\\
&SI_u = \prod_{s\in\mathcal{S}} S^p_{u,s} < K,\ w^u \in \mathcal{W}; && (5.8h)\\
&\lambda^i_p = \frac{\lambda^h_i}{\rho^i_p}, \quad p \in \{r,c\},\ i = 1,\dots,n; && (5.8i)\\
&D^i_c = D^i_h\rho^i_c, H_i = H^h_i, i = 1,\dots,n && (5.8j)\\
&E^i_c = E^i_h/\rho^i_c, i = 1,\dots,n; && (5.8k)\\
&S^i_c = S^i_h\beta^i_S, S \in \{I,P\}, i = 1,\dots,n && (5.8l)\\
&CUSI_u = CUSI(COSI_u) < K, w^u \in \mathcal{W} && (5.8m)\\
&COSI_u = COSI(\{M_S(S^i_p x^i_{u,p})|i = 1,\dots,n \\
&S \in \{I,E,D,P,H\}\}) && (5.8n)
\end{aligned}$$

Now, we have the optimization problem established, where the independent variables are the indicators of task allocation and the objective function is the sum of the cost of production operations and the set-up cost of workstations. With the formulation, we can perform our proposed algorithm to solve the problem and derive the optimal allocation in the production system.

5.3 Reinforcement learning Guided Evolutionary Algorithm

In this section, we illustrate the proposed method of solving the formulated optimization problem. First, we provide a combinatorial representation for the optimization problem in Section 5.3.1. Then, the background and application of the evolutionary algorithm are introduced in Section 5.3.2. Later, the intuition and the implementation of using reinforcement learning to guide the evolutionary algorithm are presented in Section 5.3.3. Lastly, in Section 5.3.4, we discuss the impact of hyperparameters in the proposed algorithm and provide some insights we derive from some simulations.

5.3.1 Combinatorial Optimization and Representation

With the optimization problem formulated in Section 5.2, the problem becomes a combinatorial optimization problem. The problem is an extension and more complicated variant of the line balancing problem, which is considered as NP-hard. It is hard for the existing optimization solvers to find the global optimal or even feasible solution when the number of tasks is large. Instead of traditional optimization methods, we can adopt a heuristic algorithm to solve the combinatorial optimization problem. There are some advantages and benefits of using a heuristic optimization algorithm.

Before we go through the heuristic algorithm we implement, we want to introduce the combinatorial representation of the allocation solution. Instead of the binary representation of the allocation, we will use a combinatorial representation for the allocation result for easy solvement of the optimization problem. One example of the allocation result is shown in

Figure 5.2, there are two components or rows in the representation. The first row is the sequence of the tasks in the entire production system, the i th element represents the task number that is the i th to be operated in the serial production line. In the example, the 5 th task will be the first to be operated and the 6 th task to be implemented lastly in the system. The second row is the operation alternative of the corresponding task. In the example, task 4 is assigned to be operated by a robot, and task 3 is operated by a cobot in the allocation. Later, with the sequence and the operation alternative decided in the allocation, we can derive the actual processing time of different task and the workstation can be split by the fixed cycle time limit of the entire production system. In this example, the first workstation can hold three tasks and the other three tasks will be assigned to the second workstation. If the sum of the actual processing times of the last three tasks exceeds the cycle time limit, then one more workstation will be set up and additional set-up cost will be generated.

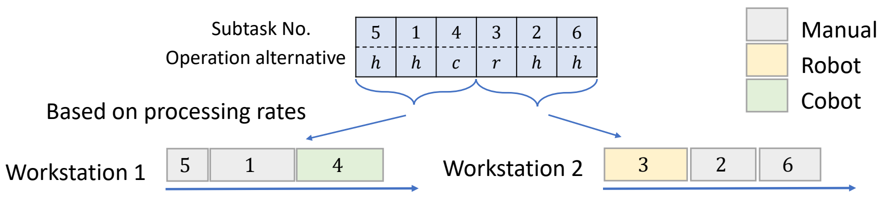


Figure 5.2: Example of the representation

5.3.2 Evolutionary Algorithm

One popular heuristic algorithm for the combinatorial optimization problem is an evolutionary algorithm (EA). EA is widely used in optimization, search, and machine learning tasks, particularly in scenarios where traditional approaches struggle due to the complexity or nonlinearity of the problem. EA mimics the process of natural selection, where a population of candidate solutions evolves over successive generations through the application of various operators. Along the generation, EA maintains a pool of potential solutions, which are iteratively refined to approximate optimal or near-optimal solutions to a given problem. The typical operators applied to generate new or child solution from the parent solution include selection, reproduction, crossover, and mutation. In our work, we utilize three types of operators: crossover and mutation for the task sequence and the switch operator for the operation alternative change. The detailed operators and how they generate new solutions are explained as follows:

5.3.2.1 Crossover operators

There are various kinds of crossover operators and they can generate different child solutions with different focuses. In this work, we decide to apply five types of classical crossover operators on the parent solutions. The crossover operators considered include (a) two-point operators; (b) single-point operators; (c) order-based operators; (d) position-based operators; and (e) cycle-based operators. In the crossover operators, there is one parent solution and one reference solution. The specific cross operators will be implemented on the parent solution with the reference solution as a reference or a crossover object. With the illustration examples shown in Figure 5.3, the detailed implementations of the operators are illustrated as follows:

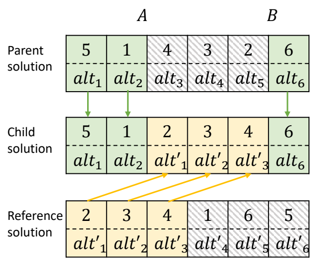
(a) Two-point operators: First, two random points A and B, as shown in Figure 5.3a, are selected. The sequence elements before A or after B in the parent solution are kept in the child solution. The operation alternatives are also kept. The elements between A and B in the child solution are replaced by the unused elements in the reference solution.

(b) Single-point operators: First, one random point A is selected, as shown in Figure 5.3b. The sequence elements before A in the parent solution are kept in the child solution. The elements after A in the child solution are replaced by the unused elements in the reference solution.

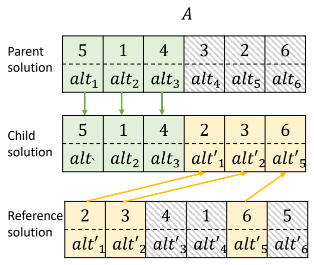
(c) Order-based operators: With the random points A and B selected, as shown in Figure 5.3c, the sequence elements between A and B in the parent solution are kept in the child solution. The elements before A or after B in the child solution are replaced by the unused elements in the reference solution.

(d) Position-based operators: A random number of positions are selected randomly, and the sequence elements in these positions of the parent solution are reserved in the child solution. The unused sequence elements are filled in the child solution later with the same position as the reference solution. The operation alternatives are kept based on the sequence elements.

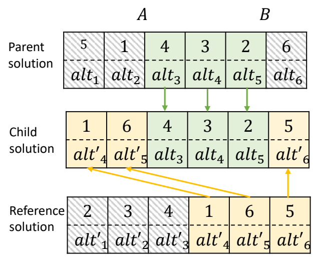
(e) Cycle-based operators: First, a random position A is selected in this type of operators. Then, as shown in Figure 5.3e, the sequence element in the reference solution of position A is kept in the child solution. The sequence element will refer to a new position the corresponding sequence element is kept in the child solution. This becomes an iterative process until the suggested position is back to A. The unspecified positions in the child solution will be filled with the corresponding sequence elements in the parent solution. The operation alternatives will be filled based on the sequence elements.



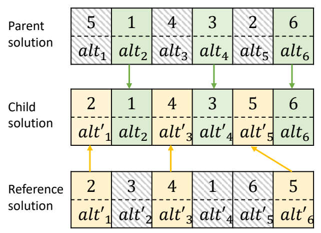
(a) Two-point crossover



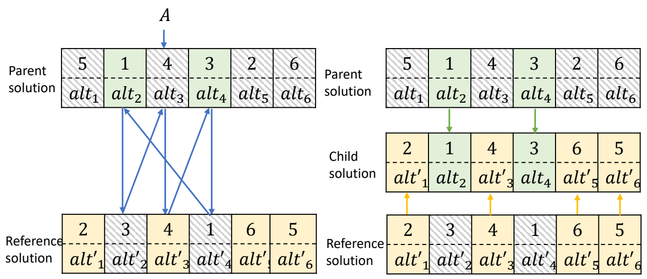
(b) Single-point crossover



(c) Order-based crossover



(d) Position-based crossover



(e) Cycle-based crossover

Figure 5.3: Illustration of crossover operators considered

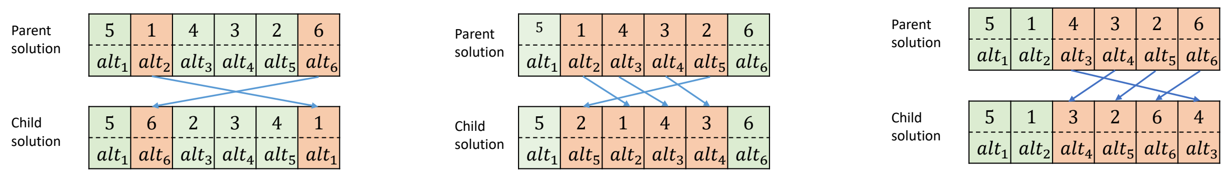
5.3.2.2 Mutation operators

Besides the crossover operators, we also consider three mutation operators, including (a) two-position mutation; (b) forward insert; and (c) backward insert. Different from the crossover operators, the mutation operators have no reference solutions. The operators are applied to a parent solution and the child solution has different task sequences from the parent solution. With the illustration examples shown in Figure 5.4, the detailed explanations are as follows:

(a) Two-position mutation: First, two different positions i and j are selected randomly. The sequence elements and the operation alternatives of the two positions in the parent solution are exchanged in the child solution. As shown in Figure 5.4a, the elements in the two positions swapped with each other. The other positions in the child solution are the same as the parent solution.

(b) Forward insert mutation: First, two different positions i and j are selected randomly. The sequence element and the operation alternative of position i in the child solution are the same as those of position j in the parent solution. The sequence elements and the operation alternatives of positions between $i+1$ and j in the child solution are the same as those of positions between i and $j-1$ in the parent solution. More specifically, the elements move one position forward in the child solution, as shown in Figure 5.4b.

(c) Backward insert mutation: First, two different positions i and j are selected randomly. Opposite to the forward insert mutation, the elements move one position backward in the child solution, as shown in Figure 5.4c. The other positions in the child solution are the same as the parent solution.


Parent solution
5 1 4 3 2 6
alt_1 alt_2 alt_3 alt_4 alt_5 alt_6
Child solution
5 6 2 3 4 1
alt_1 alt_6 alt_3 alt_4 alt_5 alt_1
Parent solution
5 1 4 3 2 6
alt_1 alt_2 alt_3 alt_4 alt_5 alt_6
Child solution
5 2 1 4 3 6
alt_1 alt_5 alt_2 alt_3 alt_4 alt_6
Parent solution
5 1 4 3 2 6
alt_1 alt_2 alt_3 alt_4 alt_5 alt_6
Child solution
5 1 3 2 6 4
alt_1 alt_2 alt_4 alt_5 alt_6 alt_3

(a) Two-position mutation (b) Forward insert mutation (c) Backward insert mutation

Figure 5.4: Illustration of mutation operators considered

5.3.2.3 Switch operator

The crossover operator and the mutation operators perform operations on the sequence element and the operation alternative together. Specifically for the operation alternative, we implement the switch operators, which only operate on the operation alternative of the tasks. The illustrations of the switch operators are shown in Figure 5.5. There are 6 kinds of switch operators, including changing the alternative from p to p', where $p, p' \in \{h, c, r\}$. In each kind of operator, the position of the switch alternative is selected randomly. Taking switch operator (a) as an example, in the operator, we first select a task randomly in the parent solution conducted by a robot, and then the selected task in the child solution will be conducted by a human worker. The sequence elements of the tasks in the child solution are the same as those in the parent solution.

By performing different types of operators to generate new solutions iteratively, the evolutionary algorithm can navigate the potential solution set and reach the optimal solution when the number of iterations is large enough.

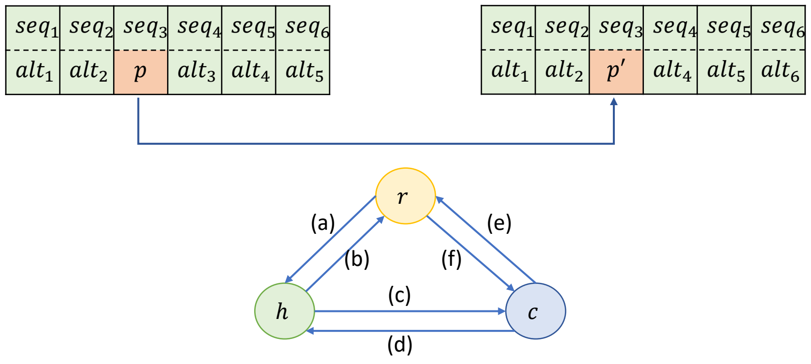

seq_1 seq_2 seq_3 seq_4 seq_5 seq_6
alt_1 alt_2 p alt_3 alt_4 alt_5
seq_1 seq_2 seq_3 seq_4 seq_5 seq_6
alt_1 alt_2 p' alt_4 alt_5 alt_6
r
h
c
(a)
(b)
(c)
(d)
(e)
(f)

Figure 5.5: Switch operators

5.3.3 Q-learning algorithm

Nevertheless, the advantages and the flexibility of the EA, the solution navigation can be and the convergence may take time. The reason that causes this scenario is illustrated in the upper part of Figure 5.6. Since the solution generation is a purely random process, it is possible to generate visited solutions but not new solutions during the navigation process. Two solutions can generate each other and the algorithm can be lost in a cycle, wasting the iterations during the convergence. To resolve this challenge, we propose to provide some guidance in the solution navigation process with reinforcement learning. The effect of the RL guidance is shown in the bottom part of Figure 5.6. With the guidance, the EA algorithm can find the optimal solution faster. Instead of selecting the operators randomly in the solution

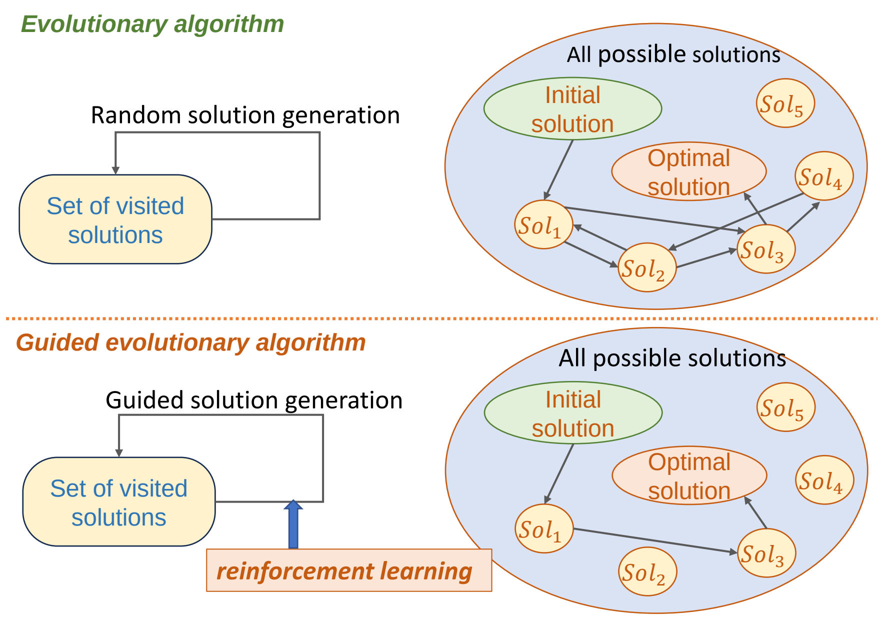


Figure 5.6: Illustration of the contribution of the RL guidance

generation procedure, we apply a Q-learning based strategy in the operator selection. Q-learning is a fundamental concept in the field of reinforcement learning, making sequential decisions to optimize the objective. Q-learning is particularly powerful in situations where an agent interacts with an environment, learning through trials to achieve long-term goals. Q-learning is a model-free reinforcement learning algorithm, that requires no knowledge of the environment's dynamics or transition probabilities. The basic idea of Q-learning is to learn a function called the Q-function (or Q-value), which estimates the expected cumulative reward of taking a particular action in a specific state. The Q-value of a state-action pair, denoted as $Q(s,a)$, represents the long-term utility of taking action a in state s, captured in the Bellman equation (5.9):

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[R(s,a) + \gamma max_{a'}Q(s',a')], \tag{5.9}$$

where $Q(s,a)$ is the Q-value of taking action a in state s and $R(s,a)$ is the immediate reward obtained after taking action a in state s. γ is the discount factor, which determines the importance of future rewards and α is the learning rate or step size which determines to what extent newly acquired information overrides old information. s' is the resulting or new state after taking action a in state s. Following equation (5.9), the Q-learning algorithm iteratively updates Q-values based on observed experiences. These updates gradually refine the estimates of Q-values, guiding the agent toward selecting actions that lead to higher cumulative rewards in a dynamic manner. During the learning process, the agent explores the environment by selecting actions according to an exploration-exploitation strategy, such as epsilon-greedy, balancing between trying new actions and exploiting the current knowledge. As the agent gains more experience, the exploration rate usually decreases, allowing the agent to exploit the learned knowledge more effectively. In our problem, the actions of Q-learning are the solution generation operators and we define different Q-value functions for different types of operators. More specifically, the crossover Q-value function has 5 subtypes of operators. The mutation Q-value function has 3 subtypes of operators and the switch Q-value function has 6 subtypes of operators. Embedded the Q-learning strategy, the algorithm of the solution generation procedure in the t-th iteration is shown in Algorithm 7.

Algorithm 7 Q-learning embedded solution generation procedure in iteration t

```
1: The current solution set $\mathcal{S}$ with $n_s$ solutions and the Q functions for different type of operators $Q_c^t, Q_m^t, Q_s^t$.
2: Initialize temporary solution set $\mathcal{S}^t = \emptyset$
3: for $Sol_i$ in $\mathcal{S}$ do
   Solution generation with crossover operators
4:     if $rand(0,1) < R_c^t$ then
5:         Take reference solution $Sol_r$
6:         Determine optimal crossover operator $o_c$ from the Q-value $Q_c^t$: $o_c = argmax_{a'} Q_c^t(s, a')$
7:         if $rand(0,1) < \epsilon^t$ then
8:             Perform the optimal crossover operator $o_c$ on $Sol_i$ and derive a new solution $Sol_{i,c}^t$
9:         else
10:            Perform random crossover operator $o_c$ on $Sol_i$ and derive a new solution $Sol_{i,c}^t$
11:        end if
12:        Update the state of the environment $s^{t+1}$ based on action $o_c$
13:        Update temporary solution set $\mathcal{S}^t = \mathcal{S}^t \cup Sol_{i,c}^t$
14:        Calculate the immediate reward $R(s^t, o_c)$ based on the objective function improvement of the temporary solution set
15:        Update Q-value function: $Q_c^{t+1}(s^t, o_c) = (1 - \alpha)Q_c^t(s^t, o_c) + \alpha[R(s^t, o_c) + \gamma max_{a'} Q_c^t(s', a')]$
16:    end if
17:    if $rand(0,1) < R_m^t$ then
18:        Perform solution generation with mutation operators similar to the crossover operators.
19:    end if
20:    if $rand(0,1) < R_s^t$ then
21:        Perform solution generation with switch operators similar to the crossover operators.
22:    end if
23: end for
```

The solution generation will go through the element in the current solution set, taking each of them as the parent solution. We will perform the crossover operators, the mutation operators, and the switch operators at a certain rate in sequence. R_c^t, R_m^t and R_s^t are the rates of taking crossover, mutation, and switch operators in the t-th iteration. The detailed discussion will be provided in Section 5.3.4. Taking the crossover operator as an example, the operator type will be chosen based on the Q-value with the rate $1 - \epsilon_t$, where ϵ_t is the hyperparameter represents the rate of taking actions randomly in the Q-learning. Once a new action o_c is taken at the current state s^t, the immediate reward $R(s^t, o_c)$ can be calculated and the Q-value can be updated. Then, the suggested action of the next iteration will be based on the new Q-value function. The overall architecture of the Q-learning guided EA is shown in Figure 5.7.

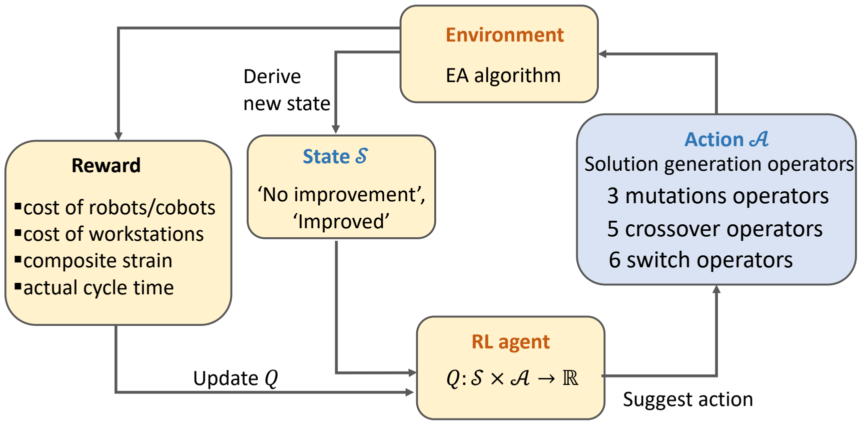


Figure 5.7: Overview of Q-learning guided Evolutionary algorithm

Here, the state of the reinforcement environment will be the improvement status of the evolutionary algorithm. The improvement status means whether the potential solution set of the problem has an improvement in the overall objective after the new solution is generated from the current operator. There are two states: 'improved from the currently generated solution' and 'not improved from the currently generated solution'. Similarly, the algorithm will learn which mutation operator and which switch operator to take during the solution generation procedure. In this way, the Q-learning algorithm guides the evolutionary algorithm dynamically during the optimization procedure.

5.3.4 Dynamic Hyperparameters Adjustment

In both evolutionary algorithm and Q-learning, there are some hyperparameters that will influence algorithms. These parameters play a crucial role in performance and convergence, governing various aspects of the algorithm's behavior and can significantly impact its effectiveness in finding optimal solutions. In this section, we will discuss the effect of choosing different hyperparameters in the proposed algorithm and briefly provide some conclusions from our simulated studies.

In the evolutionary algorithm, the hyperparameters we consider include the number of candidate solutions in each generation, the choice of parent solutions, and the rate of performing different operators. The number of candidate solutions represents the population size in each iteration and decides the number of potential optimal solutions we keep in each iteration. A larger population size can enhance the exploration rate but may increase the computational overhead. In our simulated study, we take 5, 10, 15 into consideration. Population size 10 has a better performance overall. The choice of parent solutions determines how visited solutions are selected to reproduce the next generation of solutions. Popular selection methods include proportional selection, tournament selection, and rank-based selection. Since the objective function can be calculated directly with the selected solution, the rank-based selection method fits more in our problem setting. The parent solution is selected based on its objective functions. Regarding the rate of different operators, the mutation operator tends to improve the convergence in the later stage of the optimization convergence and the crossover operator achieves better improvement in the earlier stage of the algorithm [2, 146]. We assign a dynamic crossover and mutation rate along the running time of the algorithm. Specifically, $R_c^t = 0.7 - \frac{t\epsilon_0}{mr}$, $R_c^t = \frac{t\epsilon_0}{mr}$, where mr is the maximum iteration length. When it comes to the switch rate, we utilize a fixed value $R_s^t = 0.3$ and maintain a rate of switching operation alternatives during the entire optimization procedure.

In the Q-learning, the hyperparameters we consider include the discount factor, the learning rate, and the $\epsilon-$greedy value. The discount factor determines the importance of cumulative rewards in the Q-value updates. In the simulated study, the influence of different discount factors is subtle and we will use a discount factor of 0.5. The learning rate decides the importance of the reward based on the current action. According to the simulations, a learning rate of 0.5 fits the algorithm well. The value of ϵ changes the rate of using the suggested action from the Q-learning. A higher ϵ will add more randomness and enhance the exploration in the algorithm, while a lower value will speed up the update of Q-value and improve the exploitation in the algorithm. We use a dynamic ϵ value which decreases along the running time of the algorithm: $\epsilon^t = \epsilon_0 - \frac{t\epsilon_0}{mr}$, where $\epsilon_0 = 0.2$.

5.4 Real World Scenario Based Case study

In this section, we implement the proposed allocation method in a case study based on a real world scenario. In an assembly production line of a car factory, we focus on a case where human workers and a robot/cobot can work together and the allocation of the resources can be optimized. The total number of tasks in this case is 15 and three workstations are set up to split the work to different workers. The processing times of different tasks are collected from the plant's schedule panel and the ergonomics factors related to different actions are estimated based on recorded videos of multiple trials of tasks conducted by human workers. The list of tasks is shown in Table 5.2 and the precedence relationship is shown in Figure 5.8. In this case study, based on the budget and development plan of the plant, it is possible to have two available robots and one available cobot in the future. Also, based on the plant production schedule, the cycle time of each workstation can not exceed 60s. In the task descriptions, we need to notice that the half shaft is a relatively heavy part of the car body and the lift and movement of the half shaft will cause a large body burden to the human workers. Intuitively, we should allocate the cobot or robot resources to the half shaft related tasks, including task f, j, h, i. In the precedence relationship diagram, in Figure 5.8, we can

Table 5.2: Tasks descriptions

	Task description
a	Get two bracket bolts & engine bracket
b	Get blue spline protector
c	Place spline protector to transmission seal
d	Place engine bracket behind high-pressure wire
e	Finger start two bolts for engine bracket
f	Get half shaft from stock container
g	Insert half shaft into transmission
h	Seat half shaft
i	Place half shaft on white rotor
j	Aside protector
k	Dispose of blue protector
l	Get two stabilizer bolts
m	Get scanner
n	Scan barcode on half shaft
o	Aside scanner

observe that most of the tasks have precedent tasks and they have close relationships with each other. With the large number of constraints caused by the precedence relationship, it is very likely for the traditional optimization solvers to provide us with infeasible solutions. On the other hand, the proposed data-driven Q-learning based EA will discover the solution gradually and only the feasible solutions will be kept in the potential solution set. In Figure

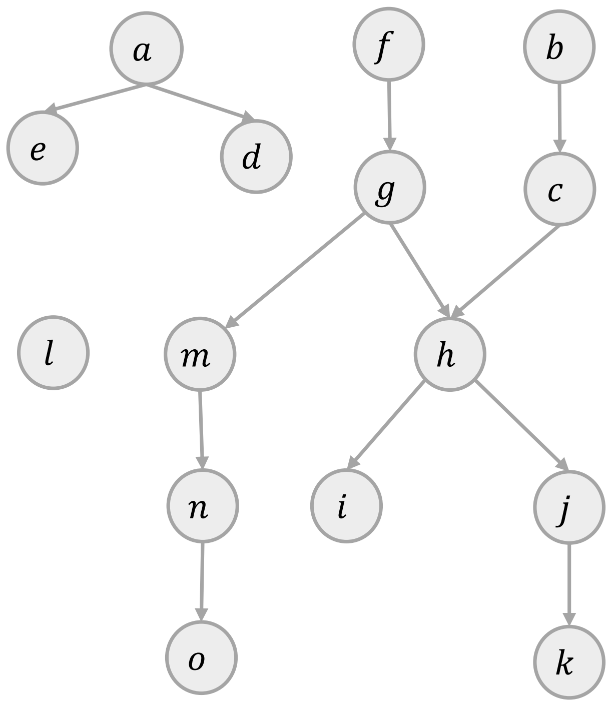


Figure 5.8: Precedence relationship between the tasks

5.9, one of the optimal final allocations derived from the Q-learning based EA is presented. There are only two workstations needed, given the system-level cycle time of 60s. The first workstation has an actual cycle time of 57s and the second workstation has an actual cycle time of 54s, which results in a well-balanced production system. Also, the available robot and cobot resources are allocated to the tasks with heavy ergonomic strain involving the half shaft which meets our expectations. The computational time of the algorithm is relatively short, taking around 10 minutes to finish 10,000 iterations. The algorithm can be sped up if the initial solution set includes one or more feasible solutions when the constraints in the algorithm are hard to satisfy. The convergence performance of Q-learning based EA (Q-EA) is compared with traditional EA. 10 replications are completed using both Q-EA algorithm

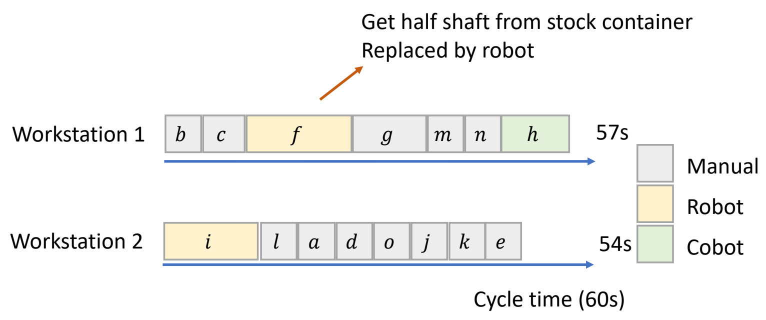


Figure 5.9: Optimal allocation

and EA algorithm. The results are the average objective function along the iterations. As shown in Figure 5.10, the Q-EA algorithm has a faster convergence speed and achieves a higher average optimal objective value. This validates the effectiveness of the guidance of solution generation of the Q-learning strategy.

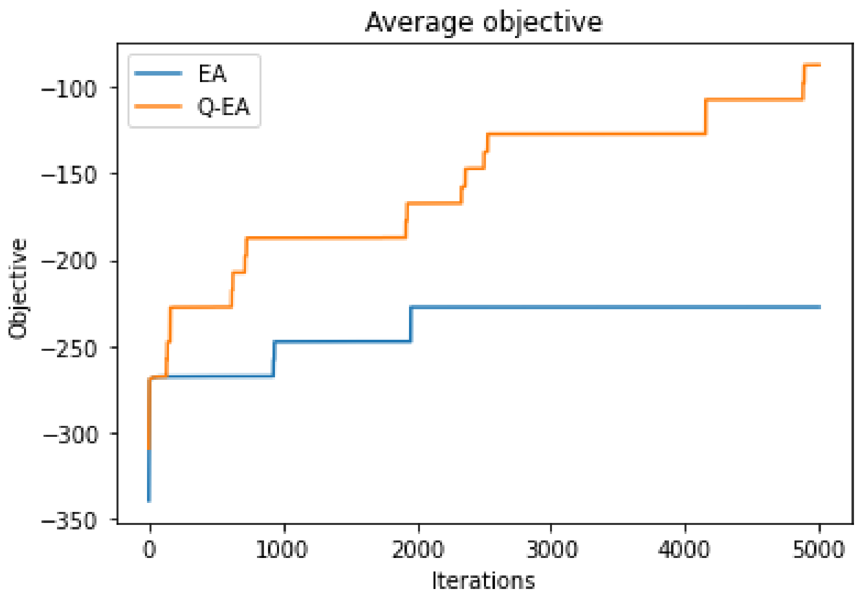


Figure 5.10: Objective change along the iterations

5.5 Conclusion

In this work, we formulate and solve an optimal allocation problem of resources in a collaborative serial line of production system with multiple workstations. This study can be used as a prerequisite study for companies that would like to incorporate their production systems with novel robotics resources. The precedence relationship among different tasks, the ergonomic strain of human workers, and the fixed cycle time of the entire system are taken into consideration. The objective is to optimize the overall performance of the production system, including the cost, the worker's well-being, and the actual throughput of the system. With the problem well formulated as a combinatorial problem, we propose to adopt a Q-learning guided Evolutionary Algorithm (Q-EA) to solve the problem. Simulated cases are conducted and the hyperparameter choice is discussed. Finally, the proposed algorithm is applied to a real-world case study in a car assembly plant. The proposed Q-EA achieves higher convergence speed and better objective value compared to the traditional EA algorithm. The final solution is validated by our expectation and can be explained by human knowledge.

In the future, there are several directions we will continue to work on. First, we plan to extend the settings of the problem to production systems with buffers. The buffer allocation will be studied to improve the performance of the system. Second, real-time data can be collected and accommodated into our algorithm to capture uncertainty in the real-world production system. Lastly, a more interpretable optimization algorithm can be developed to further provide insights of the solution navigation process. This will also help the existing manufacturing plant to improve their current resource allocation.

Chapter 6

Conclusion and Future Work

Smart systems become increasingly ubiquitous in modern industrial systems, with the invention of advanced sensors and intelligent smart components. The development provides opportunities for further improvement in the system's performance and the utilization of resources. On the other hand, it poses challenges to modeling techniques in engineering systems. In this thesis, novel data-driven and physics-based modeling techniques are developed for monitoring, diagnosis, prognosis, and design tasks in smart systems. The objectives of the tasks are of vital importance and worth investigating for the adaption of smart systems in the current industrial era. Three main research works are completed and the objectives are achieved in this dissertation.

In ***Chapter 2***, a data-driven monitoring method using a deep learning model with time-to-event data is proposed. Recurrent Neural Network is applied to fit the sequential structure of time-associated data and an encoding technique is utilized to feed time-to-event data to the deep learning model. Window techniques are also implemented to deal with the properties of real-world time-to-event datasets. The prediction error of the target event occurrence time is reduced, compared to the traditional benchmark models. Furthermore, the underlying relationships between different event types are inferred from an interpretation strategy.

In ***Chapter 3***, a data-driven and physics-based diagnosis method in complex structures was developed. This work handles the existing difficulties of fault diagnosis in complex engineering structures. A Bayesian Optimization method is incorporated with a finite element model (FE) to solve the structural damages prognosis problem. A nonseparable Multi-output Gaussian process is used as the surrogate function in the Bayesian Optimization to take the correlation between damages located in neighbor segments. This model takes the indirect measurements of impedance/admittance from the structure to identify the location and severity of the damage. Thompson sampling approach is used to guide the search for the structural damage in the proposed model. With the proposed method, fault identification can be completed with fewer measurements, and the accuracy of fault detection is promoted.

In ***Chapter 4***, a physics-based design optimization model in production systems with collaborative robots as the smart components is presented. This model allocates the tasks in a production system with multiple workstations and different operating machines. It provides a framework for task allocation with smart components and solves the optimization model with a scalable algorithm. The specialties of production systems with smart components are taken into consideration and the constraints involved from the collaborative robots are included in the optimization model. Also, the influences of smart components on traditional production systems are analyzed and illustrated.

In ***Chapter 5***, instead of allocating robots/cobots to workstations with a fixed set of tasks, we allocate the tasks to different operating alternatives and form the workstations based on the updated processing time. The precedence relationship between different tasks is considered and the cost of implementing robots/cobots and setting up new workstations are included in the optimization objective. Additionally, we propose a reinforcement learning-guided evolutionary algorithm to effectively address the combinatorial optimization problem. This approach allows reinforcement learning to successfully learn the optimal solution generation operations and guide the solution navigation of the evolutionary algorithm, as demonstrated in our case study.

In the future, we can focus on data-driven models and optimization in industrial engineering systems. Here are some potential directions:

- Techniques for integrating physical knowledge into AI learning model. By integrating existing physical knowledge of the system into learning models, the model can achieve much higher accuracy with fewer data.

- Interpretation techniques for industrial AI models. For industrial environments, people only trust the output of AI models if they understand it and can interpret it. Thus, interpreting the output of the AI model is an important aspect of industrial AI.

- Decision-making models that can handle uncertainty and randomness. Integrated with data-driven AI models and simulation techniques, optimal decisions can be made under uncertainty, which is essential due to the stochastic nature of real-world industrial systems.
- Scalable and time-efficient decision-making models. With the connection of smart systems, the volume of real-world decision-making is increasing and online decisions become necessary. Advanced industrial AI methods, such as reinforcement learning models can be developed to make real-time decisions on large-scale industrial systems.

Bibliography

[1] Collaborative robots market report (2021). `https://www.marketsandmarkets.com/Market-Reports/collaborative-robot-market-194541294.html`. Accessed: 2022-05-03.

[2] Mohamed Abdel-Basset, Reda Mohamed, and Mohamed Abouhawwash. Balanced multi-objective optimization algorithm using improvement based reference points approach. *Swarm and Evolutionary Computation*, 60:100791, 2021.

[3] Satyabrata Aich, Sabyasachi Chakraborty, Mangal Sain, Hye-in Lee, and Hee-Cheol Kim. A review on benefits of iot integrated blockchain based supply chain management implementations across different sectors with case study. In *2019 21st international conference on advanced communication technology (ICACT)*, pages 138–141. IEEE, 2019.

[4] Oguz Akkas, Cheng Hsien Lee, Yu Hen Hu, Carisa Harris Adamson, David Rempel, and Robert G Radwin. Measuring exertion time, duty cycle and hand activity level for industrial tasks using computer vision. *Ergonomics*, 60(12):1730–1738, 2017.

[5] Ahmed Z Al-Garni and Ahmad Jamal. Artificial neural network application of modeling failure rate for boeing 737 tires. *Quality and Reliability Engineering International*, 27(2):209–219, 2011.

[6] Mauricio Alvarez and Neil D Lawrence. Sparse convolved gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64, 2009.

[7] George B Arfken and Hans-Jurgen Weber. *Mathematical methods for physicists.* Academic Press Harcourt Brace Jovanovich, San Diego, 1967.

[8] Jamal Arkat, Mehdi Hosseinabadi Farahani, and Fardin Ahmadizar. Multi-objective genetic algorithm for cell formation problem considering cellular layout and operations scheduling. *International Journal of Computer Integrated Manufacturing*, 25(7):625–635, 2012.

[9] Michael Baldea, Cara R Touretzky, Jungup Park, Richard C Pattison, and Iiro Harjunkoski. Handling input dynamics in integrated scheduling and control. In *2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–6. IEEE, 2016.

[10] Logan DR Beal, Daniel C Hill, R Abraham Martin, and John D Hedengren. Gekko optimization suite. *Processes*, 6(8):106, 2018.

[11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, volume 24, pages 2546–2554. Neural Information Processing Systems Foundation, 2011.

[12] Giovanni Boschetti, Maurizio Faccio, Mattia Milanese, and Riccardo Minto. C-alb (collaborative assembly line balancing): a new approach in cobot solutions. *The International Journal of Advanced Manufacturing Technology*, 116:3027–3042, 2021.

[13] Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pages 217–224, 2005.

[14] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found. Trends Mach. Learn.*, 5(1):1–122, 2012. doi: 10.1561/2200000024. URL `https://doi.org/10.1561/2200000024`.

[15] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(10):2879–2904, 2011.

[16] Sait Cakmak, Di Wu, and Enlu Zhou. Solving bayesian risk optimization via nested stochastic gradient estimation. *IISE Transactions*, pages 1–13, 2021.

[17] Pei Cao, David Yoo, Qi Shuai, and J Tang. Structural damage identification with multi-objective direct algorithm using natural frequencies and single mode shape. In *Health Monitoring of Structural and Biological Systems 2017*, volume 10170, page 101702H. International Society for Optics and Photonics, 2017.

[18] Pei Cao, Shuai Qi, and Jiong Tang. Structural damage identification using piezoelectric impedance measurement with sparse inverse analysis. *Smart Materials and Structures*, 27(3):035020, 2018.

[19] Fei Chen, Kosuke Sekiyama, Ferdinando Cannella, and Toshio Fukuda. Optimal subtask allocation for human and robot collaboration within hybrid assembly system. *IEEE Transactions on Automation Science and Engineering*, 11(4):1065–1075, 2013.

[20] Yin-Yann Chen. A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines. *Journal of manufacturing systems*, 42:196–209, 2017.

[21] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24(2):361–370, 2016.

[23] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4):912–926, 2009.

[24] Stefano Conti and Anthony O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference*, 140(3): 640–651, 2010.

[25] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[26] Michela Dalle Mura and Gino Dini. Designing assembly lines with humans and collaborative robots: A genetic approach. *CIRP Annals*, 68(1):1–4, 2019.

[27] Geoffrey C David. Ergonomic methods for assessing exposure to risk factors for work-related musculoskeletal disorders. *Occupational medicine*, 55(3):190–199, 2005.

[28] Akash Deep, Shiyu Zhou, and Dharmaraj Veeramani. A data-driven recurrent event model for system degradation with imperfect maintenance actions. *IISE Transactions*, 54(3):271–285, 2022.

[29] Akash Deep, Shiyu Zhou, Dharmaraj Veeramani, and Yong Chen. Hmm-based joint modeling of condition monitoring signals and failure event data for prognosis. *IEEE Transactions on Reliability*, 72(3):878–888, 2022.

[30] Ana M Djuric, RJ Urbanic, and JL Rickli. A framework for collaborative robot (cobot) integration in advanced manufacturing systems. *SAE International Journal of Materials and Manufacturing*, 9(2):457–464, 2016.

[31] Takeshi Emura and Yi-Hau Chen. *Analysis of Survival Data with Dependent Censoring: Copula-based Approaches*. Springer, 2018.

[32] Maurizio Faccio, Riccardo Minto, Giulio Rosati, and Matteo Bottin. The influence of the product characteristics on human-robot collaboration: a model for the performance of collaborative robotic assembly. *The International Journal of Advanced Manufacturing Technology*, 106(5):2317–2331, 2020.

[33] Maurizio Faccio, Irene Granata, and Riccardo Minto. Task allocation model for human-robot collaboration with variable cobot speed. *Journal of Intelligent Manufacturing*, 35(2):793–806, 2024.

[34] Manuel Fechter, Carsten Seeber, and Shengjian Chen. Integrated process planning and resource allocation for collaborative robot workplace design. *Procedia CIRP*, 72: 39–44, 2018.

[35] Daniela Fogli, Luigi Gargioni, Giovanni Guida, and Fabio Tampalini. A hybrid approach to user-oriented programming of collaborative robots. *Robotics and Computer-Integrated Manufacturing*, 73:102234, 2022.

[36] Peter I Frazier. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs, 2018.

[37] Thomas E Fricker, Jeremy E Oakley, and Nathan M Urban. Multivariate gaussian process emulators with nonseparable covariance structures. *Technometrics*, 55(1):47–56, 2013.

[38] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[39] Arun Garg, J Steven Moore, and Jay M Kapellusch. The composite strain index (cosi) and cumulative strain index (cusi): methodologies for quantifying biomechanical stressors for complex tasks and job rotation using the revised strain index. *Ergonomics*, 60(8):1033–1041, 2017.

[40] Arun Garg, J Steven Moore, and Jay M Kapellusch. The revised strain index: an improved upper extremity exposure assessment model. *Ergonomics*, 60(7):912–922, 2017.

[41] Hermes Giberti, Tommaso Abbattista, Marco Carnevale, Luca Giagu, and Fabio Cristini. A methodology for flexible implementation of collaborative robots in smart manufacturing systems. *Robotics*, 11(1):9, 2022.

[42] Nikola Gjeldum, Amanda Aljinovic, M Crnjac Zizic, and Marko Mladineo. Collaborative robot task allocation on an assembly line using the decision support system. *International Journal of Computer Integrated Manufacturing*, 35(4-5):510–526, 2022.

[43] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings*

of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1487–1495, 2017.

[44] Nima Gorjian, Lin Ma, Murthy Mittinty, Prasad Yarlagadda, and Yong Sun. A review on reliability models with covariates. In *Engineering Asset Lifecycle Management*, pages 385–397. Springer, 2010.

[45] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[46] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks, 2013.

[47] Luca Gualtieri, Ilaria Palomba, Erich J Wehrle, and Renato Vidoni. The opportunities and challenges of sme manufacturing automation: safety and ergonomics in human–robot collaboration. In *Industry 4.0 for SMEs*, pages 105–144. Palgrave Macmillan, Cham, 2020.

[48] Liang Guo, Naipeng Li, Feng Jia, Yaguo Lei, and Jing Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[49] Robert Gwadera, Mikhail J Atallah, and Wojciech Szpankowski. Reliable detection of episodes in event sequences. *Knowledge and Information Systems*, 7(4):415–437, 2005.

[50] Houman Hanachi, Christopher Mechefske, Jie Liu, Avisekh Banerjee, and Ying Chen. Performance-based gas turbine health monitoring, diagnostics, and prognostics: A survey. *IEEE Transactions on Reliability*, 67(3):1340–1363, 2018.

[51] Kimmo Hatonen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Knowledge discovery from telecommunication network alarm databases. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 115–122. IEEE, 1996.

[52] Kimmo Hatonen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Tasa: Telecommunication alarm sequence analyzer or how to enjoy faults in your network. In *Proceedings of NOMS'96-IEEE Network Operations and Management Symposium*, volume 2, pages 520–529. IEEE, 1996.

[53] Anqi He and Xiaoning Jin. Deep variational autoencoder classifier for intelligent fault diagnosis adaptive to unseen fault categories. *IEEE Transactions on Reliability*, 2021.

[54] J Hedengren, J Mojica, W Cole, and T Edgar. Apopt: Minlp solver for differential and algebraic systems with benchmark testing. In *Proceedings of the INFORMS National Meeting, Phoenix, AZ, USA*, volume 1417, page 47, 2012.

[55] Michael Andreas Herzog, Tshilidzi Marwala, and Philippus Stephanus Heyns. Machine and component residual life estimation through the application of neural networks. *Reliability Engineering & System Safety*, 94(2):479–489, 2009.

[56] Sahar Heydaryan, Joel Suaza Bedolla, and Giovanni Belingardi. Safety design and development of a human-robot collaboration assembly process in the automotive industry. *Applied Sciences*, 8(3):344, 2018.

[57] Bin Hu and Jing Chen. Optimal task allocation for human–machine collaborative manufacturing systems. *IEEE Robotics and Automation Letters*, 2(4):1933–1940, 2017.

[58] Congfang Huang, Shiyu Zhou, Jingshan Li, and Robert G Radwin. Allocating robots/cobots to production systems for productivity and ergonomics optimization. *IEEE Transactions on Automation Science and Engineering*, 2023. doi: 10.1109/TASE.2023.3270207.

[59] Yong Huang, James L Beck, and Hui Li. Bayesian system identification based on hierarchical sparse bayesian learning and gibbs sampling with application to structural damage assessment. *Computer Methods in Applied Mechanics and Engineering*, 318: 382–411, 2017.

[60] Lee T Jacobsen, Benjamin J Spivey, and John D Hedengren. Model predictive control

with a rigorous model of a solid oxide fuel cell. In *2013 American Control Conference*, pages 3741–3746. IEEE, 2013.

[61] Salman Jahani, Shiyu Zhou, Dharmaraj Veeramani, and Jeff Schmidt. Multioutput gaussian process modulated poisson processes for event prediction. *IEEE Transactions on Reliability*, 70(4):1569–1580, 2021. doi: 10.1109/TR.2021.3088094.

[62] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79 (1):157–181, 1993.

[63] Kelin Jose and Dilip Kumar Pratihar. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems*, 80:34–42, 2016.

[64] Jay M Kapellusch, Arun Garg, Stephen S Bao, Barbara A Silverstein, Susan E Burt, Ann Marie Dale, Bradley A Evanoff, Frederic E Gerr, Carisa Harris-Adamson, Kurt T Hegmann, et al. Pooling job physical exposure data from multiple independent studies in a consortium study of carpal tunnel syndrome. *Ergonomics*, 56(6):1021–1037, 2013.

[65] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deep survival: A deep cox proportional hazards network. *stat*, 1050: 2, 2016.

[66] Wazir Zada Khan, MH Rehman, Hussein Mohammed Zangoti, Muhammad Khalil Afzal, Nasrullah Armi, and Khaled Salah. Industrial internet of things: Recent advances, enabling technologies and open challenges. *Computers & Electrical Engineering*, 81:106522, 2020.

[67] J-T Kim and Norris Stubbs. Crack detection in beam-type structures using frequency data. *Journal of sound and vibration*, 259(1):145–160, 2003.

[68] Jinki Kim and KW Wang. An enhanced impedance-based damage identification method using adaptive piezoelectric circuitry. *Smart materials and structures*, 23(9): 095041, 2014.

[69] Minhee Kim and Kaibo Liu. A bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions*, 53(3):326–340, 2020.

[70] Yong-Sik Kim, Nicholas G Dagalakis, Jeremy Marvel, and Geraldine Cheok. Design and testing of wireless motion gauges for two collaborative robot arms. *Measurement Science Review*, 22(2):84–91, 2022.

[71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[72] John P Klein and Melvin L Moeschberger. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, 2005.

[73] Mika Klemettinen, Heikki Mannila, and A Inkeri Verkamo. Association rule selection in a data mining environment. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 372–377. Springer, 1999.

[74] Raed Kontar, Shiyu Zhou, Chaitanya Sankavaram, Xinyu Du, and Yilu Zhang. Nonparametric modeling and prognosis of condition monitoring signals using multivariate gaussian convolution processes. *Technometrics*, 60(4):484–496, 2018.

[75] Pradeep Lall, Ryan Lowe, and Kai Goebel. Extended kalman filter models and resistance spectroscopy for prognostication and health monitoring of leadfree electronics under vibration. *IEEE Transactions on Reliability*, 61(4):858–871, 2012.

[76] Rémi Lam, Matthias Poloczek, Peter Frazier, and Karen E Willcox. Advances in bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference*, page 1656, 2018.

[77] Nicholas R Lewis, John D Hedengren, and Eric L Haseltine. Hybrid dynamic optimization methods for systems biology with efficient sensitivities. *Processes*, 3(3):701–729, 2015.

[78] Zhigang Li, Ming-Tan Sun, Margaret H Dunham, and Yongqiao Xiao. Improving the web site's effectiveness by considering each page's temporal information. In *International Conference on Web-Age Information Management*, pages 47–54. Springer, 2003.

[79] Zhiguo Li, Shiyu Zhou, Suresh Choubey, and Crispian Sievenpiper. Failure event prediction using the cox proportional hazard model driven by frequent failure signatures. *IIE transactions*, 39(3):303–315, 2007.

[80] Aijun Liu, Michele Pfund, and John Fowler. Scheduling optimization of task allocation in integrated manufacturing system based on task decomposition. *Journal of Systems Engineering and Electronics*, 27(2):422–433, 2016.

[81] Chunfeng Liu, Jufeng Wang, Joseph Y-T Leung, and Kai Li. Solving cell formation and task scheduling in cellular manufacturing system by discrete bacteria foraging algorithm. *International Journal of Production Research*, 54(3):923–944, 2016.

[82] Li Liu, Andrew J Schoen, Curt Henrichs, Jingshan Li, Bilge Mutlu, Yajun Zhang, and Robert G Radwin. Human robot collaboration for enhancing work activities. *Human Factors*, page 00187208221077722, 2022.

[83] Li Liu, Andrew J Schoen, Curt Henrichs, Jingshan Li, Bilge Mutlu, Yajun Zhang, and Robert G Radwin. Human robot collaboration for enhancing work activities. *Human Factors*, 66(1):158–179, 2024.

[84] Yuhang Liu, Qi Shuai, Shiyu Zhou, and Jiong Tang. Prognosis of structural damage growth via integration of physical model prediction and bayesian estimation. *IEEE Transactions on Reliability*, 66(3):700–711, 2017.

[85] Isabela Maganha, Cristovao Silva, Nathalie Klement, Amélie Beauville dit Eynaud, Laurent Durville, and Samuel Moniz. Hybrid optimisation approach for sequencing and assignment decision-making in reconfigurable assembly lines. *IFAC-PapersOnLine*, 52 (13):1367–1372, 2019.

[86] Anandamayee Majumdar and Alan E Gelfand. Multivariate spatial modeling for geostatistical data using convolved covariance functions. *Mathematical Geology*, 39(2): 225–245, 2007.

[87] Jianing Man and Qiang Zhou. Remaining useful life prediction for hard failures using joint model with extended hazard. *Quality and Reliability Engineering International*, 34(5):748–758, 2018.

[88] Alberto Pliego Marugán, Ana María Peco Chacón, and Fausto Pedro García Márquez. Reliability analysis of detecting false alarms that employ neural networks: A real case study on wind turbines. *Reliability Engineering & System Safety*, 191:106574, 2019.

[89] Tariq Masood and Paul Sonntag. Industry 4.0: Adoption challenges and benefits for smes. *Computers in Industry*, 121:103261, 2020.

[90] Arman Melkumyan and Fabio Ramos. Multi-kernel gaussian processes. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1408, 2011.

[91] Jennifer E Michaels and Thomas E Michaels. Guided wave signal processing and image fusion for in situ damage localization in plates. *Wave motion*, 44(6):482–492, 2007.

[92] George Michalos, Jason Spiliotopoulos, Sotiris Makris, and George Chryssolouris. A method for planning human robot shared tasks. *CIRP journal of manufacturing science and technology*, 22:76–90, 2018.

[93] Jiyoung Min, Seunghee Park, Chung-Bang Yun, Chang-Geun Lee, and Changgil Lee. Impedance-based structural health monitoring incorporating neural network technique for identification of damage type and severity. *Engineering Structures*, 39:210–220, 2012.

[94] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[95] Mahdi Mokhtarzadeh, Reza Tavakkoli-Moghaddam, Behdin Vahedi-Nouri, and Azadeh Farsi. Scheduling of human-robot collaboration in assembly of printed circuit boards: a constraint programming approach. *International Journal of Computer Integrated Manufacturing*, 33(5):460–473, 2020.

[96] Hoora Moradian, Denis Larocque, and François Bellavance. Survival forests for data with dependent censoring. *Statistical methods in medical research*, 28(2):445–461, 2019.

[97] G Mossa, F Boenzi, S Digiesi, G Mummolo, and VA Romano. Productivity and ergonomic risk in human based production systems: A job-rotation scheduling model. *International Journal of Production Economics*, 171:471–477, 2016.

[98] Rainer Müller, Matthias Vette, and Aaron Geenen. Skill-based dynamic task allocation in human-robot-cooperation with the example of welding application. *Procedia Manufacturing*, 11:13–21, 2017.

[99] Dang Nguyen, Sunil Gupta, Santu Rana, Alistair Shilton, and Svetha Venkatesh. Bayesian optimization for categorical and category-specific continuous inputs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5256–5263, April 2020. doi: 10.1609/aaai.v34i04.5971. URL `https://doi.org/10.1609/aaai.v34i04.5971`.

[100] Shimon Y Nof. *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[101] Changyong Oh, Efstratios Gavves, and Max Welling. Mixed variable bayesian optimization with frequency modulated kernels. In *Uncertainty in Artificial Intelligence*, pages 950–960. PMLR, 2021.

[102] Xinyan Ou, Qing Chang, Nilanjan Chakraborty, and Junfeng Wang. Gantry scheduling for multi-gantry production system by online task allocation method. *IEEE Robotics and Automation Letters*, 2(4):1848–1855, 2017.

[103] Margaret Pearce, Bilge Mutlu, Julie Shah, and Robert Radwin. Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes. *IEEE transactions on automation science and engineering*, 15(4):1772–1784, 2018.

[104] Ricardo Perera, Sheng-En Fang, and Antonio Ruiz. Application of particle swarm optimization and genetic algorithms to multiobjective damage identification inverse problems with modelling errors. *Meccanica*, 45(5):723–734, 2010.

[105] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[106] Grandys Frieska Prassida and Ully Asfari. A conceptual model for the acceptance of collaborative robots in industry 5.0. *Procedia Computer Science*, 197:61–67, 2022.

[107] Peter Z G Qian, Huaiqing Wu, and CF Jeff Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008.

[108] SM Mizanoor Rahman and Yue Wang. Mutual trust-based subtask allocation for human–robot collaboration in flexible lightweight assembly in manufacturing. *Mechatronics*, 54:94–109, 2018.

[109] Fabian Ranz, Vera Hummel, and Wilfried Sihn. Capability-based task allocation in human-robot collaboration. *Procedia Manufacturing*, 9:182–189, 2017.

[110] John Rosecrance, Robert Paulsen, and Lelia Murgia. Risk assessment of cheese processing tasks using the strain index and ocra checklist. *International Journal of Industrial Ergonomics*, 61:142–148, 2017.

[111] Binxin Ru, Ahsan Alvi, Vu Nguyen, Michael A Osborne, and Stephen Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *International Conference on Machine Learning*, pages 8276–8285. PMLR, 2020.

[112] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[113] Ilya O Ryzhov. On the convergence rates of expected improvement methods. *Operations Research*, 64(6):1515–1528, 2016.

[114] Md Omar Faruque Sarker, Torbjørn S Dahl, Elsa Arcaute, and Kim Christensen. Local interactions over global broadcasts for improved task allocation in self-organized multi-robot systems. *Robotics and Autonomous Systems*, 62(10):1453–1462, 2014.

[115] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[116] SM Seyedpoor, S Shahbandeh, and O Yazdanpanah. An efficient method for structural damage detection using a differential evolution algorithm-based optimisation approach. *Civil Engineering and Environmental Systems*, 32(3):230–250, 2015.

[117] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[118] Qi Shuai, K Zhou, Shiyu Zhou, and J Tang. Fault identification using piezoelectric impedance measurement and model-based intelligent inference with pre-screening. *Smart Materials and Structures*, 26(4):045007, 2017.

[119] Jannis Sinnemann, Marius Boshoff, Raphael Dyrska, Sebastian Leonow, Martin Mönnigmann, and Bernd Kuhlenkötter. Systematic literature review of applications and usage potentials for the combination of unmanned aerial vehicles and mobile robot manipulators in production systems. *Production Engineering*, pages 1–18, 2022.

[120] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012.

[121] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29:4134–4142, 2016.

[122] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit

setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012. doi: 10.1109/tit.2011.2182033. URL `https://doi.org/10.1109/tit.2011.2182033`.

[123] Kathryn E Stecke and Mahdi Mokhtarzadeh. Balancing collaborative human–robot assembly lines to optimise cycle time and ergonomic risk. *International Journal of Production Research*, 60(1):25–47, 2022.

[124] J Steven Moore and Arun Garg. The strain index: a proposed method to analyze jobs for risk of distal upper extremity disorders. *American Industrial Hygiene Association Journal*, 56(5):443–458, 1995.

[125] Shengjing Sun, Xiaochen Zheng, Javier Villalba-Díez, and Joaquín Ordieres-Meré. Indoor air-quality data-monitoring system: Long-term monitoring benefits. *Sensors*, 19 (19):4157, 2019.

[126] Shozo Takata and Takeo Hirano. Human and robot allocation method for hybrid assembly systems. *CIRP annals*, 60(1):9–12, 2011.

[127] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[128] Diego A Tibaduiza Burgos, Ricardo C Gomez Vargas, Cesar Pedraza, David Agis, and Francesc Pozo. Damage identification in structural health monitoring: A brief review from its implementation to the use of data-driven applications. *Sensors*, 20(3-733): 1–32, 2020.

[129] Panagiota Tsarouchi, Alexandros-Stereos Matthaiakis, Sotiris Makris, and George Chryssolouris. On a human-robot collaboration in an assembly cell. *International Journal of Computer Integrated Manufacturing*, 30(6):580–589, 2017.

[130] Anastasios Tsiatis. *Semiparametric theory and missing data*. Springer Science & Business Media, 2007.

[131] Mark J Van Der Laan and Ian W McKeague. Efficient estimation from right-censored data when failure indicators are missing at random. *Annals of Statistics*, pages 164–182, 1998.

[132] Jonathan L Vandergrift, Judith E Gold, Alexandra Hanlon, and Laura Punnett. Physical and psychosocial ergonomic risk factors for low back pain in automobile manufacturing workers. *Occupational and environmental medicine*, 69(1):29–34, 2012.

[133] Jay M Ver Hoef and Ronald Paul Barry. Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69 (2):275–294, 1998.

[134] Nicolas Vignais, Markus Miezal, Gabriele Bleser, Katharina Mura, Dominic Gorecky, and Frédéric Marin. Innovative system for real-time ergonomic feedback in industrial manufacturing. *Applied ergonomics*, 44(4):566–574, 2013.

[135] David M Vock, Julian Wolfson, Sunayan Bandyopadhyay, Gediminas Adomavicius, Paul E Johnson, Gabriela Vazquez-Benitez, and Patrick J O'Connor. Adapting machine learning techniques to censored time-to-event health record data: A general-purpose approach using inverse probability of censoring weighting. *Journal of biomedical informatics*, 61:119–131, 2016.

[136] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Back. A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512. IEEE, 2017.

[137] Haowei Wang, Jun Yuan, and Szu Hui Ng. Gaussian process based optimization algorithms with input uncertainty. *IISE Transactions*, 52(4):377–393, 2020.

[138] Ying Wang and Hong Hao. Damage identification scheme based on compressive sensing. *Journal of Computing in Civil Engineering*, 29(2):04014037, 2015.

[139] Thomas R Waters, Vern Putz-Anderson, Arun Garg, and Lawrence J Fine. Revised niosh equation for the design and evaluation of manual lifting tasks. *Ergonomics*, 36 (7):749–776, 1993.

[140] Christian Weckenborg, Karsten Kieckhäfer, Christoph Müller, Martin Grunewald, and

Thomas S Spengler. Balancing of assembly lines with collaborative robots. *Business Research*, 13(1):93–132, 2020.

[141] Johannes Wiebe, Inês Cecílio, Jonathan Dunlop, and Ruth Misener. A robust approach to warped gaussian process-constrained optimization, 2020. URL `https://arxiv.org/abs/2006.08222`.

[142] Yongqiao Xiao and Margaret H Dunham. Efficient mining of traversal patterns. *Data & Knowledge Engineering*, 39(2):191–214, 2001.

[143] Zhaoyi Xu, Yanjie Guo, and Joseph Homer Saleh. Accurate remaining useful life prediction with uncertainty quantification: A deep learning and nonstationary gaussian process approach. *IEEE Transactions on Reliability*, 2021.

[144] Yuan Yuan, Shiyu Zhou, Crispian Sievenpiper, Kamal Mannar, and Yibin Zheng. Event log modeling and analysis for system failure prediction. *IIE Transactions*, 43 (9):647–660, 2011.

[145] Ya-Jun Zhang, Li Liu, Ningjian Huang, Robert Radwin, and Jingshan Li. From manual operation to collaborative robot assembly: an integrated model of productivity and ergonomic performance. *IEEE Robotics and Automation Letters*, 6(2):895–902, 2021.

[146] Zikai Zhang, Qiuhua Tang, Manuel Chica, and Zixiang Li. Reinforcement learning-based multiobjective evolutionary algorithm for mixed-model multimanned assembly line balancing under uncertain demand. *IEEE Transactions on Cybernetics*, 2023.

[147] Qiang Zhou, Peter ZG Qian, and Shiyu Zhou. A simple approach to emulation for computer models with qualitative and quantitative factors. *Technometrics*, 53(3):266–273, 2011.