Acta Universitatis Sapientiae

Informatica

Volume 9, Number 2, 2017

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 101–118

DOI: 10.1515/ausi-2017-0007

Anomaly detection techniques in cyber-physical systems

Gheorghe SEBESTYEN
Technical University of Cluj-Napoca, Romania
email: Gheorghe.Sebestyen@cs.utcluj.ro

Anca HANGAN
Technical University of Cluj-Napoca, Romania
email: Anca.Hangan@cs.utcluj.ro

Abstract. Nowadays, when multiple aspects of our life depend on complex cyber-physical systems, automated anomaly detection, prevention and handling is a critical issue that influence our security and quality of life. Recent catastrophic events showed that manual (human-based) handling of anomalies in complex systems is not recommended, automatic and intelligent handling being the proper approach. This paper presents, through a number of case studies, the challenges and possible solutions for implementing computer-based anomaly detection systems.

1 Introduction

Anomaly detection in physical processes (form very simple ones like an electric motor toward very complex industrial infrastructures) is not a new task; it is part of the operating and maintenance procedure of that system. Because of the multitude of anomaly sources and consequent system behaviors this task was traditionally left to the experience and intuition of a human operator. But in today's complex cyber-physical systems with thousands of process variables

Computing Classification System 1998: B.3.4, B.8.1
Mathematics Subject Classification 2010: 68M15
Key words and phrases: anomaly detection, outliers detection, cyber-physical systems, time series, spatio-temporal data

involved and multiple automatic control loops the ability of a human operator to identify an abnormal behavior or state is overwhelming. Sometimes the required reaction time to a given event is much under the typical reaction time of a human (which is usually greater than 0.1 s).

There are a number of examples of catastrophic events caused by the fact that an abnormal system behavior was not properly identified and handled. For instance, recently (Aug. 2016) an explosion could have been avoided at the Petromidia petroleum processing plant (in Romania) if the human operators wouldn't have ignored a sequence of alerts that signaled a gas leakage [13]. But the real problem was that more than 1.6 million alerts were generated by the automated system in the last 3 days previous to the explosion, probably a lot of them being false alerts. In front of such a huge number of alerts a human cannot identify and classify the anomalies and threats at their correct risk level.

Therefore automated algorithms and methods are needed to identify and handle in real-time critical system anomalies. But implementing efficient anomaly detection methods is not a trivial task. For example, for a person is rather simple to say that something is wrong with his/her car based just on the sound generated by that car and a specialist can even tell the component that cause the trouble. Transposing such an intuitive detection into an algorithm or automated method is not a straightforward task.

The difficulty starts with the definition of an abnormal behavior or simply of an anomaly. It continues with the multitude of possible anomalies and sources of anomalies. An anomaly may be caused by accidental (non-malicious) causes such as: a communication error, faulty equipment or measuring device, a noisy signal and significant environmental changes; it may also be caused by intentional (malicious) actions, such as: a virus, an intruder or a theorist. There are examples of cybernetic attacks specially designed for very critical cyber-physical and embedded systems (e.g. Stuxnet, Duqu).

It is generally accepted that an anomaly is a deviation from a normal state or behavior; therefore it is important to identify a normal state (or states) as a discriminant for identifying anomalies. As it will be showed in the case studies, most of the proposed anomaly detection mechanisms are trying to identify a number of relevant features of the analyzed system that allows making the difference between normal and abnormal behavior.

Due to abnormal system behavior, monitoring data sets include outliers. The term "outlier" was originally used in the field of statistics and it is [1] defined as an observation that is inconsistent with the set of data it belongs to. Even if they are not quite equivalent, in many cases the terms "outlier

detection" and "anomaly detection" are used with the same meaning. In our view outlier detection is more an off-line process, while anomaly detection is an on-line one.

As presented in a very good survey on anomaly detection methods [2] there are different forms of anomalies, different anomaly sources, different types of systems (system behaviors) and different application domains. Therefore there is a very wide range of methods used for this purpose, "borrowed" from multiple domains, such as: statistics, data mining, machine learning, information theory, signal processing and spectral analysis, etc.

The goal of this paper is to analyze through some examples those methods that are best fitted for the cyber-physical domain. Typical for this domain is the use of sensorial networks and sensorial data, the need for on-line (real-time) analysis and detection and the presence of multiple correlations between the acquired data. As shown in the next chapters, the common feature for the methods applied in different case studies is the identification of an anomaly as a value or a state that breaks the previously detected or learned correlation rules.

This paper is a retrospective survey of our manifold research in the area of anomaly detection applied in different domains and for various purposes.

The rest of the paper is structured as follows: the next section presents some basic concepts and related research in the field of anomaly detection, specific for cyber-physical systems. Section 3 tries to classify the different conceptual approaches for anomaly detection and analyze the possibility to adapt a given method to the specificity of physical systems. The next sections present a number of case studies for different types of anomaly sources and system types. These sections reflect some of our previous results in different areas. The last section presents our conclusions and some future research possibilities.

2 Related work

There are several recent survey papers that try to organize and classify the large amount of research work that has been conducted in the field of outlier or anomaly detection, while highlighting the research issues that still need attention [12, 6, 4, 11].

The authors in [12] identify three large types of outlier detection problems based on outlier sources: fault detection in case of noise and defects, event detection in case of multi-variable systems and intrusion detection in case of malicious attacks. One of the main challenges of outlier detection in sensor networks

is in fact identifying the source of outlier data, since traditional techniques fail to distinguish between errors and events. Other important challenges identified in [12] are related to the scalability and the computational complexity of the detection techniques. The authors classify the outlier detection methods in statistical-based approaches, nearest-neighbor based approaches, clustering-based, classification-based approaches and spectral decomposition-based approaches. They point out that the first two classes can't handle multivariate data sets and the other, more complex methods can't be easily used for large scale sensor networks because of their large resource requirements or computational complexity.

The authors of [6] identify the requirements for an efficient and effective anomaly detection model that include five items: reduction of data, online detection, distributed detection, adaptive detection and correlation exploitation. They point out that current anomaly detection models have important limitations such as the failure of adaptability in dynamic environments, not taking into account spatial and temporal correlations between data and the absence of automated parameter tuning.

Other surveys [4, 11] extensively cover outlier detection techniques that are used for the detection of malicious attacks. The authors in [4] mainly cover the problem of data injections in sensor networks. As they classify the techniques used for the detection of anomalies, they emphasize the importance of attribute, temporal and spatial correlation in solving the problem of multiple compromised sensors that produce anomalous values in a coordinated fashion. In [11], the authors make a classification of security threats in sensor networks and of the outlier detection methods used. They conclude that data mining and computational intelligence based schemes are the strongest in terms of detection generality as long as the adequate attributes are selected. Finally, they identify some potential research areas such as modeling the problem of anomaly detection, attribute selection and the development of a uniform performance evaluation standard.

In this research context, our paper tries to give a more pragmatic approach to the anomaly detection problem. Through the case studies we show that the key for any anomaly detection method is to find the set of features that discriminate between normal and abnormal system states, process variable values or events. It is also important to find correlations between process variables that are broken in case of an anomaly.

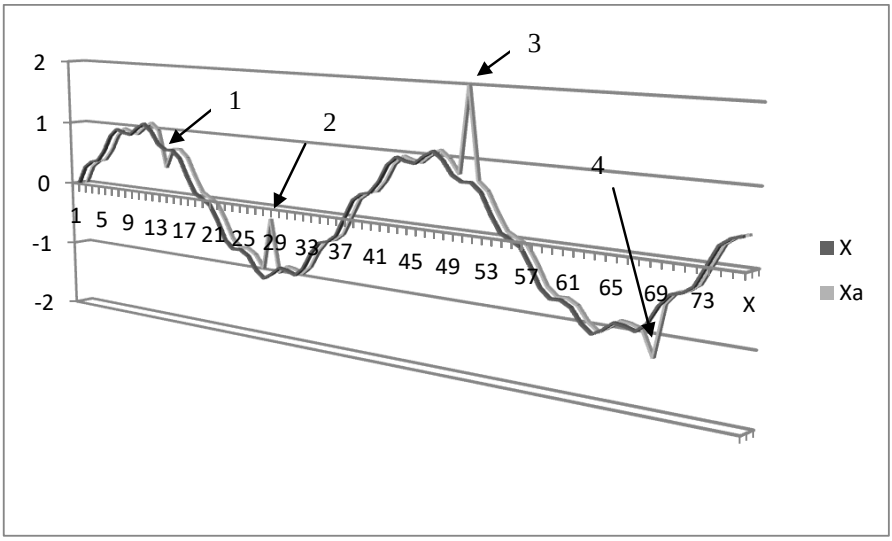


Figure 1: Anomalies in time series (X : original signal, Xa : signal with singular anomalies).

3 Anomaly detection techniques

An important group of anomalies are singular values that do not fit with the rest of the acquired data. In time series these outliers can be seen as values that do not follow the continuous shape of a variable graph. Some values, which are outside of a normal variation range (e.g. dots 3 and 4 in Figure 1), can be detected if a minimum and maximum value is set or detected on a training set. Other values are in the normal range but it is still obvious for a human eye that something is wrong (e.g. dots 1 and 2 in Figure 1). These outlier values can be detected with linear and parabolic prediction or through autocorrelation techniques (see more details in next chapter, case study "a").

Also singular anomalies may be detected in spatially distributed variables. For instance, in environmental monitoring systems, values (e.g. temperature, pressure, humidity) measured in a small vicinity tend to be similar or at least correlated somehow. In such systems (see Figure 2) an outlier is a value that does not fit with the spatial curves of the neighbor values. Linear approximation and spatial correlation techniques may be used for detection. Sometimes time and spatial correlation may be combined for more accurate anomaly detection.

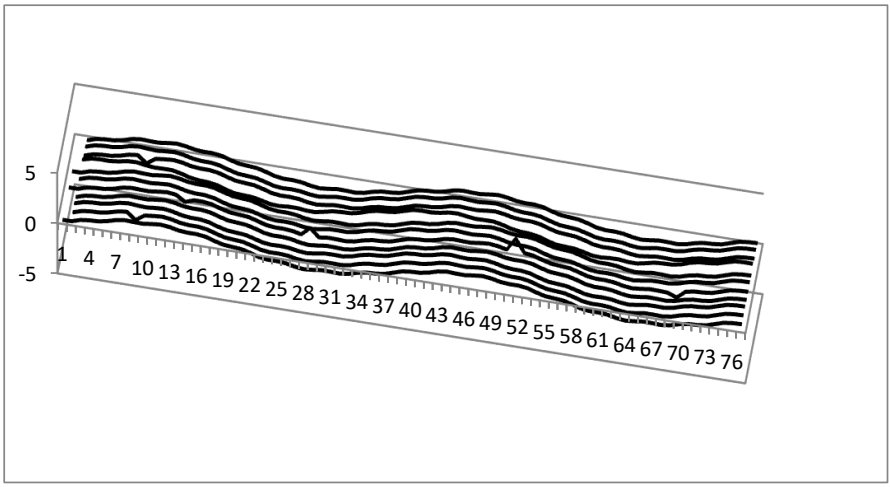


Figure 2: Anomalies in spatially distributed values.

In a cyber-physical system there are multiple functional correlations between process variables, which can be used for anomaly detection. These functional dependencies may be theoretically deducted from the physical and chemical laws that govern that process or they may be determined experimentally from the measured data sets. Figure 3 shows 5 process variables and Table 1 presents computed correlations between some pairs of variables. Through correlation values we can establish that variable ν is mostly correlated with x and z variables and less correlated with y and u. In this case there is a functional relation between ν, x and z, which may be exploited for anomaly detection.

Correlation	x and Y	x and ν	z and ν	y and V	u and ν
Values	-0.66	0.91	0.89	-0.64	0.19

Table 1: Correlations between pairs of variables.

Another category of anomalies (beside singular ones) are those that change the typical shape of a signal. In this case the allowed variation domain or the "continuity" feature of the graph are not violated and therefore other techniques must be applied, techniques that recognize the normal and abnormal shape of the signal. Here, pattern recognition and classification methods are used. For instance, a doctor can recognize a given heart disease based on the specific normal and abnormal ECG signals. A pattern recognition tool (e.g.

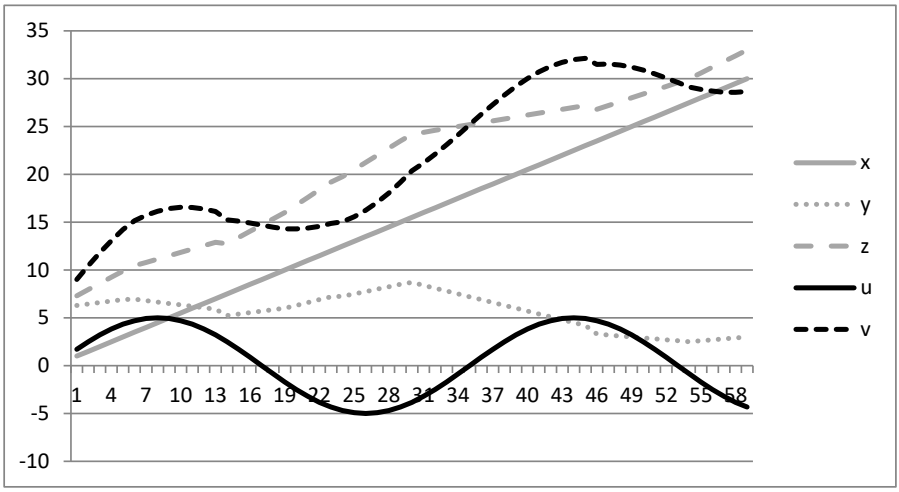


Figure 3: Functional correlations between variables.

neural network, decision tree) is trained with normal and abnormal ECG signal shapes. But in cyber-physical a system, generating abnormal signal shapes is not a trivial task (it may even destroy the system) and there are many abnormal behaviors, most of them not predictable from the design phase.

An interesting approach in this area is to classify in simple terms (e.g. letters or codes) the different slopes of a signal and then identify a normal or abnormal behavior based on the sequence and duration of codes. We used this approach for identifying road anomalies (see case study "d") and also abnormal behavior of elderly persons [8]. Because the human behavior is rather complex, with multiple possible choices, normality was hard to define. Hidden Markov chains were trained in order to classify normal and abnormal behavior.

4 Case studies of anomaly detection

This chapter gathers a number of relevant cases regarding anomaly detection methods developed for different purposes. In every case we analyze the main goal of the detection, possible methods and expected outcomes.

4.1 Anomaly detection in sensorial networks

In case of sensorial networks most of the methods used [2] try to exploit the existing correlations between the data acquired from sensors. Usually there are three types of correlations that may be identified in such systems:

- Time correlation or correlation of a sensor with itself;
- Spatial correlation or correlation between a sensor and its neighbors;
- Functional correlation or a correlation imposed by the functional relations between components of a complex system.

The first kind of correlation is specific for process variables that have a quasi-continuous evolution in time and their future behavior can be predicted from their past values. In this case linear prediction and auto-regression techniques can be used. Linear prediction is an easy and fast method that can be implemented even at the intelligent sensor's level. A predicted value $\bar{X}$ is computed using the last 2 (linear approximation) or 3 samples (parabolic approximation) of the signal. If the difference (ϵ) between the predicted and the last measured value exceeds a given threshold the value is considered a candidate outlier. The threshold can be learned in a training phase as the maximum difference occurred in the training set; the condition is to have a training set without outlier values. Usually the outlier value will be replaced with the predicted one.

The success of this method depends on the granularity of the time sampling (the sampling rate). In order to apply successfully a linear or parabolic approximation the original curve of the signal should be well approximated with line segments or parabolic segments. Our experiments showed that if the sampling frequency is one magnitude (10 times) higher than the highest frequency in the input signal than the approximation error is reasonably small and the error threshold can be kept small. Otherwise, computed differences in the training set will be high and consequently the threshold is too high for a good outlier discriminant. The maximum frequency in the input set can be obtained by applying an FFT on the training set. To avoid false high frequencies generated mainly by noise, the amplitude of the highest frequency in FFT taken into consideration should be a fraction (e.g. 1/10) of the biggest harmonic amplitude. Threshold computation can be done in the initialization phase when no time limits are imposed.

A more computer-intensive method for anomaly detection is through auto-regression. The predicted value is computed as a weighted some of previous

samples, as follows:

$$\bar{X}_i[k] = \sum_{j=1}^{N} u_{i,j} X_i[k-j],$$

where $u_{i,j}$ is the weighting coefficient of order j of node i in an auto regression model. Computing u_i coefficients is a computer intensive process, which can be performed only in a device with sufficient computing resources (not on a microcontroller or an intelligent sensor). The coefficients can be computed on the training set, but also on the incoming samples. The window of samples on which the auto-regression model is computed must include a relevant period of time in the evolution of the signal, meaning that the window must include seasonal variations of the time series (variations cause by day-night cycles or season changes). Some programing languages (e.g. the "R" language used by us) have very good library functions for auto-regression and linear modeling coefficients computation.

An outlier value is detected if the difference between the predicted and measured value is higher than a threshold; this threshold can be determined based on the "residuals" of the auto-regression model.

For systems that change their behavior in time the auto regression model should be periodically recomputed on the newly collected data.

Another correlation which may be exploited in sensorial data is the spatial correlation. For instance if there is a set of sensors that are collecting temperature values in a given region it is reasonable to suppose that the values generated by a sensor are in a correlation with the values generated by its neighbors. In this case again a linear model or a regressive model can be computed for each node of the network. Now the predicted value of a node is computed using its neighbors values at the same sampling time or at a lagged time. The lag (or time delay) can be determined experimentally or based on a physical propagation formula (e.g. propagation of temperature gradient in a given environment):

$$\bar{X}_i[k] = \sum_{j=1}^{N} u_{i,j} X_{i,j}[k]$$

where

- $u_{i,j}$ is the weighting coefficient of neighbor j,
- N may be 3 to 8 (for pragmatic reasons),
- $X_{i,j}[k]$ the j-th neighbor of node i.

The vicinity of a node in a sensorial network can be obtained using the geographical position of the nodes in the area (e.g. GPS coordinates). If such information does not exist than proximity of a neighbor can be determined through the radio connectivity between nodes and the amplitude (power) of the radio signal. Of course, a triangulation method would improve the precision of selecting the best neighbor candidates. The number of neighbors may vary from 3 to 8 depending on the available time for computation.

The linear approximation technique can be implemented directly on the sensor nodes. Every node hears (through radio transmission) their neighbor reports and can decide if its value is an outlier. In a similar way the detection can be made by the nodes that aggregate data through the acquisition tree.

The regression model on spatially distributed nodes requires more computing power and can be implemented at the "Access point" node or in a central computer (e.g. server). In the formula that predicts the value of a node at moment "k" we can include the weighted sum of the neighbors' values at the same "k" moment as well as values at one, two or more earlier sampling periods.

$$\bar{X}_i[k] = \sum_{j=1}^{N} u_{i,j,0} X_{i,j}[k] + \sum_{j=1}^{N} u_{i,j,1} X_{i,j}[k-1] + \sum_{j=1}^{N} u_{i,j,2} X_{i,j}[k-2] + \ldots$$

where $u_{i,j,l}$ is the weighting coefficient for neighbor j and time delay l.

Functional correlation can be exploited as an alternative for spatial correlation, when the similarity between two variables is in accordance with some functional dependencies between the system's parameters and spatial proximity between two nodes is not relevant. This is the case for a sensor network that collects multiple types of process variable values and there is a correlation between variables in accordance with the physical laws that govern that process. For instance in an electrical energy distribution system the voltage, current, power and energy measurements must be in accordance with the electricity laws (e.g. Kirchhoff's laws).

Functional dependencies between any two process variables can be established on theoretical bases or through an experimental process. In the first case the designer must know a-priory the physical law that govern the process and interconnect the process variables. The system theory shows that finding a true and precise model of a system is not a trivial task and in many cases the multiple external influences (e.g. environmental variations) diverts the system's behavior from the pure theoretical mode. Therefore an experimental approach is more feasible. We can build an experimental model of the system (a process called identification in system theory), or we can compute

correlation functions between pears of process variables. For a variable we can consider as its closest neighbors the "N" variables for which the correlation functions are the highest. This computation can be done off-line in the learning phase, based on some previously collected data.

4.2 Pollution detection in rivers using rule-based systems

The detection of abnormal events in environmental monitoring is based on analyzing the values obtained from sensors and the correlations between these values. In the special case of pollution detection in rivers, time, spatial, as well as functional correlations between different parameters have to be taken into consideration.

Several parameters such as temperature, pH, specific conductivity, dissolved oxygen, turbidity and discharge can be used to assess the quality of water. Some of these parameters are measured using sensors (e.g. temperature, pH) and others are computed based on measured values (e.g. discharge is computed based on pressure and river profile measurements [3]). To be able to take advantage of time and spatial correlations, the measurements, acquired from sensors, have to be made continuously in subsequent locations on the river shore for each of these parameters.

Our approach for detecting events while monitoring water quality parameters is a two-step rule based system. In the first step, the parameter values are labeled based on a set of rules that take into consideration time and space correlations between the values measured for each parameter. In the second step, a second rule-based component assesses the functional correlations between several parameters to detect events such as river shore erosion, floods or chemical pollution.

The first step of the rule-based event detection system is focused on the detection of anomalies in the time series of each measured parameter, at each location. These anomalies can be erroneous measurements provided by faulty sensors or values that are outside the accepted value interval, which may signal an event. Labeling rules are different for each parameter, not only because accepted value intervals and correlation rules differ, but also because some parameters' accepted value intervals are variable based on the context in which they are measured (e.g. normal values for water temperature vary based on season). During labeling, it is important to differentiate between erroneous measurements and actual events. This is done by correlating the values measured at subsequent locations. If an event appears at one location, then the measurements downstream for the same parameter will be correlated. More-

over, values showing events are time correlated. In case of errors, there are no spatial correlations. A faulty sensor can give unpredictable readings. The labels assigned to the measured or computed values place them in one of the following categories: error, normal, low, high.

Labeled values are passed to the second step rule-based component that will be able to detect actual events based on correlations between several parameters values. For example, river shore erosion may be detected based on high turbidity and high discharge. River shore erosion may signal the risk of floods. If the river shore is near an agricultural land, in the presence of a flood, there is a high risk of nitrate and nitrite pollution. High turbidity is usually detected during and after a rainfall and it causes an increase of temperature and a decrease of dissolved oxygen. This will cause damage to the flora and fauna of the river. Conductivity and pH levels are specific to each water stream due to the soil and geology. Therefore, the change in pH and increased conductivity levels signal the presence of polluting chemicals such as nitrate, phosphate or sodium.

By applying similar water quality assessment rules the second step component will be able to identify various types of events. The performance of event detection is heavily influenced by the quality of preliminary value labeling. An increased spectrum of categories (label types) should improve the assessment process. A partial implementation of this system and its integration with a water monitoring system for Somes River is presented in [10].

4.3 Malicious attack detection using system models

Malicious attacks on cyber-physical systems are another source for anomalies and abnormal behavior of some automatically controlled systems. Before a catastrophic failure happens a number of anomalies may indicate an imminent attack on the system. The goal in this case is to identify the initial signs of an attack and counteract in order to avoid total failure.

One possibility is to use traditional virus and intruder detection methods specific for computer systems. But as showed in [5] cyber-physical systems require specific detection methods that take into account the type of equipment involved (sensors, actuators, regulators, PLCs), the gravity of a malicious attack and the inter-correlation between process variables.

The idea promoted in our research is to try to model the physical process and then simulate different attacks in different points of the infrastructure in order to identify and learn malfunctioning patterns. Then these patterns can be used as discriminants for identifying real attacks.

Two kinds of cyber-physical systems were modeled: a chemical process and an electrical distribution network [5]. In both cases the models allowed us to inject false data at different points and measure their effect upon system variables. It was demonstrated that an efficient attack is not one that try to influence major elements in the infrastructure (they can be detected rather simple in an effective time) but an attack that keep its effect stealthy as much time as possible. In the second case the initial variations are too small for a simple anomaly detector and later when the effects are detectable a significant part of the infrastructure is already under the control of the attacker.

A system model allows an anomaly detector to compute the next predicted value based on the previously measured ones. A maliciously injected value will differ significantly from the predicted one, being a candidate for an anomaly.

Another bases for anomaly detection is an inherent redundancy between measured process variables. The system model gives the inter-conditioning relations between different process variables. For instance in the electrical distribution network example, the sum of the currents going in and out of an intersection must be theoretically zero. In practice, because of the energy loses on the electrical lines an error threshold had to be considered. Similar relations can be found between variables of different types (e.g. power, current and voltage). This kind of anomaly detection can be used not just for malicious attacks but also for malfunctioning components. In the second case (faulty component) the effect tends to be permanent.

The designer of the anomaly detector module can define a set of rules or inter-conditioning relations extracted from the system model. If a precise model of the system does not exist the rules may be formulated based on the experience and intuition of the human operator; in this case Fuzzy relations are preferred.

More difficult is to consider dynamic relations between variables, which are described by differential equations. Dynamic behavior is typical for transitions between more or less stable states of the monitored system. Here an experimentally determined transfer function allows us to write a time dependency between an input and an output variable and then this relation is used for anomaly detection. In system theory this process is called system identification and a number of experimental methods are given for determining the transfer function. Again an error threshold must be considered between the predicted (computed) and measured output variable. The level of the error is influenced by the effect of the noise over the analyzed component (which can be determined in the training phase).

The dynamic behavior should be taken into consideration when the transition periods of the system are more dominant over the stable state periods. In our case, static relations were typical for the electrical distribution network model and dynamic relations for the chemical process.

4.4 Anomaly detection through pattern recognition

A forth direction of our research was to identify an abnormal behavior through pattern recognition techniques. The idea is to collect and learn a number of normal and abnormal behaviors of the system variables (e.g. time variation patterns) and then use them as discriminant for abnormal behavior. There are many methods described in the literature [2] that can be used for pattern recognition (e.g. neural networks, frequency analysis, classification and clustering, SVM, etc.), most of them being time consuming. Our goal was to develop simple methods that can be used for on-line (real-time) anomaly detection and they should be deployed on devices with limited resources.

In this case study [9] our idea was to identify anomalies in the road based on the acceleration signals (on 3 directions) collected from a smart phone placed in a car. The goal was twofold: to identify and locate the holes and speed bumps in a road section and also to give a quality measure of a road section. Through crowd sourcing a realistic and up-to-date map of a given region (e.g. city, highway, etc.) can be obtained and users can be notified about anomalies on the roads they are traveling.

For the first part we implemented a sequence of low and high pass filters that allowed us to discriminate between usual trepidations of the car (caused by low quality roads, acceleration/decelerations, engine rotation, etc.) and variations caused by holes or bumps. Then, with an adaptive threshold we determined a region in the curve as candidate for an anomaly. A neural network was trained to identify different categories of road holes and bumps. As input the neural network considers the order and the sign of the curve slopes and the magnitude and the duration of the abnormal period.

For the second goal (road quality evaluation) a number of features were extracted from the acceleration signals, such as dominant frequencies, component amplitudes and frequency of anomalies. Through calibration we reduced the effect of car speed over the measured signals. More details on this research can be found here [9].

This experiment showed us that simple intuitive rules deployed as a sequence of signal processing procedures (e.g. filters, FFT) allowed us to develop an efficient and real-time road anomaly detection system. Similar techniques can

be used for identifying abnormal shapes in the graph of a process variable. Based on our experience we can say as a rule of thumb that if an abnormal shape is recognizable for the human eye than probably a set of signal processing procedures and rules can be implemented in a program that will recognize that shape. This rule applies for anomalies which are a-priori known (as the holes in our experiment).

Anomaly detection based on the signals' shape recognition can be used in cyber-physical systems as well as in many other domains such as: medicine (e.g. ECG complexes, EEG waves, electromyography), electrical and mechanical components maintenance (e.g. early signs of failure), financial processes, meteorology or earth sciences. The methods used are similar but the interpretations are very different.

4.5 Anomaly detection in network traffic

In computer networks the traffic shape and content is very divers because communication applications are run randomly on different computers. Opposed to this case, in networks used for cyber-physical systems the traffic is dominated by periodical data flows. Usually here the data acquisition, processing, storage and visualization are made in a periodical manner and consequently the traffic associated to these activities adopts the same periodicity. Also the order of the activities (tasks) is somehow stable. This quasi-stable state may change if a malicious code tries to infiltrate in the system or if some kind of physical failure occurred and the system reacts with some counter measures. From our point of view both cases can be classified as anomalies.

Based on these observations we can define as an anomaly discriminator a significant change in the pattern of the packages transmitted on the network. The pattern can be identified through the following features:

- The frequencies of different types of packages
- The order of different types of packages
- The lengths of different package types
- Delays between different packages

Through a network sniffer component, in the training phase, the program can identify the types of packages transferred through the network, their periodicity (or their sporadic nature), the typical length of the packages (depending on their type) and the order of the packages. Sometimes these details are a-priori known by the physical-system's designer or by the control systems developer.

Also in some industrial networks (e.g. Profibus, FF, WorldFIP) the traffic pattern is set in the configuration phase and it is strictly imposed through a MAC protocol. Any change in this pattern may be considered an anomaly.

In less restricted networks (e.g. cell industrial networks, Etherbus, CAN, control over Internet) some pattern features can still be detected and transformed into anomaly detection rules. In a training phase a sniffer program can determine all the package types transferred through the network, their length interval (min, max) and repetition frequency. Any significant deviation from normal values is considered candidate for anomaly. Sporadic packages don't have a regular repetition period, but even in this case a minimum frequency can be derived from the physical phenomena or component that initiated it. For instance in a car (on its CAN network), the frequency of packages sent by the rotation sensor placed on the engine cannot exceed the maximum rotation frequency of that engine. Similarly packages reflecting the driver's activity (wheel movement) cannot exceed the reaction time of a human.

In a complex cyber-physical system for reliability and robustness reasons a single anomaly detector is not enough [7]. Multiple detection points must be established in a consistent manner, in different points of the network infrastructure. In [7] we proposed a method for optimal placement of anomaly detectors. The method minimizes a combined cost function that takes into consideration the total coverage of each network node, the transmission overhead and the delays. Further research is needed to identify and express normal and abnormal traffic patterns used by the detector nodes.

5 Conclusions

Analyzing the different cases presented in the paper we can generate a number of rules that may help a developer to select and implement the best anomaly detection solution for a given cyber-physical system. Here are our conclusions:

- Today's cyber-physical systems are becoming so complex and incorporate so many components that a manual (human) anomaly detection is not recommended and in some cases is even impossible;
- Most anomaly detection methods are trying to exploit some regularities or correlations existing between process variables during normal execution;
- The discriminants for detecting anomalies must be built upon a set of signal or system features that mostly change in an abnormal behavior;

- As shown in the case studies, these discriminants are very different, depending on the domain, the source of the anomaly and the complexity of the system;
- In most cases the anomaly detection method must be tolerant with some variations caused by known (e.g. noise) or unknown sources (e.g. Gaussian spread of values);
- In a cyber-physical system multiple anomaly detection points should be spread in the infrastructure and a combination of multiple techniques can coupe better with the multitude of anomaly sources and types.

As future work, based on our previous experiments, we try to develop a platform that will contain a number of anomaly detection tools. This platform will be used by a developer to test the best combination of anomaly detection methods for a given analyzed system. The platform will include facilities for acquiring data from different sources, tools for automatic generation of anomalies and interactive interfaces for flexible result evaluation.

Acknowledgements

The results presented in this paper were obtained with the support of the Technical University of Cluj-Napoca through the research Contract no. 1995/ 12.07.2017, Internal Competition CICDI-2017.

The authors of this paper would like to thank some of our students, whose work was essential for testing some of our ideas and for the implementation of case studies.

References

[1] V. Barnett, T. Lewis, *Outliers in Statistical Data*, New York: John Wiley Sons, 1994. ⇒102

[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, *ACM Computing Surveys* **41,** 3 (2009). ⇒103, 108, 114

[3] P. Deac, M. Muste, O. Creţ, L. Văcariu, H. Hedesiu, A prototype for the continuous and cost-effective measurement of river discharge, *Proc. 2013 19th International Conference on Control Systems and Computer Science (CSCS '13)*, Bucureşti, Romania, 2013, pp. 628–633. ⇒111

[4] V. P. Illiano, E. C. Lupu, Detecting malicious aata injections in wireless sensor networks: a survey, *ACM Computig Surveys* **48,** 2 (2015). ⇒103, 104

[5] I. Kiss, *Security improvement techniques for networked critical infrastructures*, PhD Thesis, Technical University of Cluj-Napoca, Romania, 2016. ⇒112, 113

[6] M. A. Rassam, A. Zainal, M. A. Maarof, Advancements of data anomaly detection research in wireless sensor networks: a survey and open issues, *Sensors* **13,** 8 (2013) 10087–10122. ⇒103, 104

[7] H. Sandor, Gh. Sebestyen, Optimal security design in the Internet of Things, *Digital Forensic and Security (ISDFS)*, 2017. ⇒116

[8] Gh. Sebestyen, I. Stoica, A. Hangan, Human activity recognition and monitoring for elderly people, *Proc. 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj, Romania, 2016, pp. 341–347. ⇒107

[9] Gh. Sebestyen, D. Mureşan, A. Hangan, Road quality evaluation with mobile devices, *Proc. 2015 16th International Carpathian Control Conference (ICCC)*, Bucuresti, Romania, 2015, pp. 458–464. ⇒114

[10] L. Văcariu, A. Hangan, O. Creţ, A. Creţu, A decision support system on the cyberwater platform, *Proc. 2017 21st International Conference on Control Systems and Computer Science (CSCS '17)*, Bucureşti, Romania, 2017, pp. 591–598. ⇒112

[11] M. Xie, S. Han, B. Tian, S. Parvin, Anomaly detection in wireless sensor networks:a survey, *Journal of Network and Computer Applications* **34,** 4 (2011) 1302–1325. ⇒103, 104

[12] Y. Zhang, N. Meratnia, P. Havinga, Outlier detection techniques for wireless sensor networks: a survey, *IEEE Communications Surveys and Tutorials* **12,** 2 (2010) 159–170. ⇒103, 104

[13] ∗ ∗ ∗ Rompetrol Refinery explosion, *www.wall-street.ro*, http://www.wall-street.ro/articol/Companii/205623/rompetrol-rafinare-trimisa-in-judecata-in-dosarul-privind-explozia-de-la-petromidia-soldata-cu-doi-morti-si-doi-raniti.html. ⇒102

Received: September 7, 2017 • Revised: November 17, 2017

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 119–133

DOI: 10.1515/ausi-2017-0008

Further results on color energy of graphs

Prajakta Bharat JOSHI
Christ University
Bengaluru - 560029, India.
email:
prajakta.joshi@res.christuniversity.in

Mayamma JOSEPH
Department of Mathematics
Christ University
Bengaluru - 560029, India.
email:
mayamma.joseph@christuniversity.in

Abstract. Given a colored graph G, its color energy $\mathsf{E_c}(\mathsf{G})$ is defined as the sum of the absolute values of the eigenvalues of the color matrix of G. The concept of color energy was introduced by Adiga et al. [1]. In this article, we obtain some new bounds for the color energy of graphs and establish relationship between color energy $\mathsf{E_c}(\mathsf{G})$ and energy $\mathsf{E}(\mathsf{G})$ of a graph G. Further, we construct some new families of graphs in which one is non-co-spectral color-equienergetic with some families of graphs and another is color-hyperenergetic. Also we derive explicit formulas for their color energies.

1 Introduction

The concept of energy of a graph G was introduced by Gutman [9] in 1978 as the sum of the absolute values of the eigenvalues of the adjacency matrix of the graph G. His focus was to solve a question from theoretical chemistry "*how energy depends on the molecular structure?*" In other words, he tried to find the relation between the energy of a graph and its structure. Gutman has

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05C50
Key words and phrases: energy, color matrix, color eigenvalues, color energy of a graph

obtained an upper and a lower bound for energy of a graph G in terms its size m,

$$2\sqrt{m} \leq E(G) \leq 2m \tag{1}$$

He further characterized graphs for which these bounds are sharp. For details of graph energy we refer to [9, 10, 11].

For many years, researchers have extended the concept of graph energy and continued to work on varieties of graph energy such as Laplacian energy, distance energy etc [11].

Recently Adiga et al. [1] introduced the concept of color energy of a graph based on the color matrix of the graph.

Definition 1 *Let G be a vertex colored graph of order n. Then the color matrix of G is the matrix $A_c(G) = [a_{ij}]_{n\times n}$, whose entries are given by*

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are adjacent with } c(v_i) \neq c(v_j) \\ -1 & \text{if } v_i \text{ and } v_j \text{ are non-adjacent with } c(v_i) = c(v_j) \\ 0 & \text{otherwise.} \end{cases}$$

where $c(v_i)$ is the color of a vertex v_i in G.

If the eigenvalues of $A_c(G)$ are $\lambda_1, \lambda_2, \ldots \lambda_n$, which are also called as color eigenvalues, then the color energy $E_c(G)$ is the sum of their absolute values. That is,

$$E_c(G) = \sum_{i=1}^{n} |\lambda_i|$$

If a graph G is colored with minimum number of colors χ, then $E_\chi(G)$ is the color energy of G, $A_\chi(G)$ is the color matrix, $P_\chi(G, \lambda)$ is the characteristic polynomial and $Spec_\chi(G)$ is the spectrum of the graph G.

They have proved that

$$\sum_{i=1}^{n} \lambda_i^2 = 2(m + m'_c) \tag{2}$$

where m is the size of G and m'_c is the number of pairs of non-adjacent vertices receiving the same color in G.

Further, the authors have derived explicit formulas for color energies of some families of graphs and have obtained bounds for $E_c(G)$. Among those results the following would be used for further discussion.

The color energies of K_n and $K_{1,n-1}$ are

$$E_\chi(K_n) = 2(n-1) \text{ and } E_\chi(K_{1,n-1}) = 2(n-1) \tag{3}$$

respectively with the spectra

$$Spec_\chi(K_n) = \begin{pmatrix} -1 & n-1 \\ n-1 & 1 \end{pmatrix} \text{ and } Spec_\chi(K_{1,n-1}) = \begin{pmatrix} -(n-1) & 1 \\ 1 & n-1 \end{pmatrix} \tag{4}$$

and for any graph G

$$\sqrt{2(m+m'_c) + n(n-1)D^{\frac{2}{n}}} \leq E_c(G) \leq \sqrt{2n(m+m'_c)} \tag{5}$$

where $D = |\det(A_c(G))|$.

In this direction, Rajesh Kanna et al. [12] have proved that the color energy of the friendship graph $F_3^{(k)}$ of order n is $2(n-1)$ with the spectrum

$$Spec_\chi(F_3^{(k)}) = \begin{pmatrix} 0 & 2 & -k \\ k-1 & k & 2 \end{pmatrix} \tag{6}$$

where the friendship graph $F_3^{(k)}$ is the graph obtained by taking k copies of C_3 with a vertex in common.

Research in the area of color energy has seen rapid rise in recent years and concepts such as color Laplacian energy [4, 15], color signless Laplacian energy [5], minimum covering color energy of a graph [13], reduced color energy etc. [2, 3] were added to the literature. Although several studies have been done in this area, no study has been initiated to explore the relation between color energy and energy of graphs.

Apart from these studies, a classification of graphs of order ≤ 6 on the basis of their color energy is found in [16]. Also the lower bounds in terms of the smallest and largest color eigenvalues of a graph G with order n, size m and the number of pairs of the non-adjacent vertices in G receiving same color were obtained in [17].

In this paper, we establish relationships between $E_c(G)$ and $E(G)$ apart from finding new bounds for $E_c(G)$. In addition to this, we introduce non-cospectral graphs that are color-equienergetic with complete graphs and a family of color-hyperenergetic graphs.

All graphs considered in this paper are simple and connected. Our graph theoretic and spectral graph theoretic terminologies follow [6, 7, 8, 18].

2 Bounds for color energy of graphs

In this section, we present some new bounds for the color energy of graphs in terms of the largest positive color eigenvalue λ_1 and the largest absolute value of the color eigenvalue λ_{max} of $A_c(G)$. First we present an upper bound for color energy in terms of λ_1, order n, size m and m'_c which denotes the number of pairs of non-adjacent vertices receiving the same color in G.

Theorem 2 *Let G be a colored graph of order n, size m. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the color eigenvalues of $A_c(G)$. Then*

$$E_c(G) \leq |\lambda_1| + \sqrt{(n-1)[2(m+m'_c) - \lambda_1^2]} \tag{7}$$

where m'_c be the number of pairs of non-adjacent vertices receiving the same color.

Proof. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are color eigenvalues of $A_c(G)$, so by the Cauchy-Schwartz inequality,

$$\left(\sum_{i=2}^{n} |\lambda_i|\right)^2 \leq (n-1)\left(\sum_{i=2}^{n} |\lambda_i|^2\right)$$

$$\sum_{i=2}^{n} |\lambda_i| \leq \sqrt{(n-1)\left(\sum_{i=2}^{n} |\lambda_i|^2\right)}$$

$$= \sqrt{(n-1)[2(m+m'_c) - {\lambda_1}^2]}, \text{ by Equation (2).}$$

Hence,

$$E_c(G) \leq |\lambda_1| + \sqrt{(n-1)[2(m+m'_c) - {\lambda_1}^2]}.$$

□

Remark 3 *The above theorem gives an upper bound for color energy of a graph G. This is an improvement on the upper bound given in the Inequality (5). For example, consider the graph G given in Figure 1 which is a paw. $\chi(G) = 3$ with respect to given coloring. Inequality (5) yields the result that $E_c(G) \leq 6.33$ whereas Inequality (7) shows that $E_c(G) \leq 6.24$. It is to be observed that $\text{Spec}_\chi(G) = \begin{pmatrix} -2 & -1 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix}$ and hence $E_\chi(G) = 6$.*

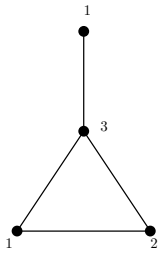


Figure 1: Graph G

Theorem 4 *If G is a graph of order n, size m and λ_{max} is the largest absolute value of eigenvalue of the color matrix of G, then*

$$E_c(G) \geq \left(\frac{2(m + m'_c)}{\lambda_{max}}\right)$$

where m'_c be the number of pairs of non-adjacent vertices receiving the same color.

Proof. Let λ_{max} be the largest absolute value of color eigenvalue of $A_c(G)$. Then

$$\lambda_{max}|\lambda_i| \geq \lambda_i^2$$

holds for $i = 1, 2, \ldots, n$. Then summing over all i's, we get

$$\sum_{i=1}^{n} \lambda_{max}|\lambda_i| \geq \sum_{i=1}^{n} \lambda_i^2$$

$$\lambda_{max} \sum_{i=1}^{n} |\lambda_i| \geq 2(m + m'_c), \text{ by Equation (2).}$$

So that,

$$E_c(G) \geq \left(\frac{2(m + m'_c)}{\lambda_{max}}\right).$$

□

Next we present bounds of the color energy of a graph G in terms of only m and m'_c. In order to prove this result, we require the following lemma.

Lemma 5 ([14]) *If A is a real or complex $n \times n$ matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, then for $1 \leq k \leq n$*

1. $S_k = (-1)^k c_k$.

2. S_k *is the sum of the* $k \times k$ *principal minors of* A.

where c_k*s are the coefficients in characteristic polynomial of* A *and* S_k*, the* k^{th} *symmetric function of* $\lambda_1, \lambda_2, \ldots, \lambda_n$ *is the sum of the products of the eigenvalues taken* k *at a time.*

Theorem 6 *If* G *is a colored graph of order* n*, size* m *and* m'_c *is the number of pairs of non-adjacent vertices receiving the same color, then*

$$2\sqrt{(m + m'_c)} \leq E_c(G) \leq 2(m + m'_c).$$

Proof. Consider,

$$\begin{aligned}(E_c(G))^2 &= \left(\sum_{i=1}^{n} |\lambda_i|\right)^2 \\ &= \sum_{i=1}^{n} |\lambda_i|^2 + \sum_{i \neq j} |\lambda_i||\lambda_j| \\ &= \sum_{i=1}^{n} |\lambda_i|^2 + 2\sum_{i<j} |\lambda_i||\lambda_j|. \end{aligned} \tag{8}$$

By Lemma 5, S_2 is the sum of all 2×2 principal minors of $A_c(G)$. Therefore, we get

$$\begin{aligned}\sum_{1 \leq i < j \leq n} \lambda_i \lambda_j &= \sum_{1 \leq i < j \leq n} \begin{vmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{vmatrix} \\ &= \sum_{1 \leq i < j \leq n} (a_{ii}a_{jj} - a_{ij}a_{ji}).\end{aligned}$$

As $A_c(G)$ is the color matrix, $a_{ij} = a_{ji}$ and $a_{ii} = 0\ \forall i$. Thus,

$$\begin{aligned}\sum_{1 \leq i < j \leq n} \lambda_i \lambda_j &= \sum_{1 \leq i < j \leq n} -(a_{ij})^2 \\ &= -(m + m'_c).\end{aligned} \tag{9}$$

We know that

$$\sum_{i<j} |\lambda_i||\lambda_j| \geq |\sum_{i<j} \lambda_i \lambda_j| \tag{10}$$

therefore, from Equation (9) and (10), we have

$$\sum_{i<j} |\lambda_i||\lambda_j| \geq |m + m'_c|. \tag{11}$$

Using this together with Equations (2), (8) and (11), we get

$$\begin{aligned}(E_c(G))^2 &\geq 2|(m + m'_c)| + 2|(m + m'_c)| \\ &\geq 4|(m + m'_c)|.\end{aligned}$$

Taking positive square-root, we get

$$E_c(G) \geq 2\sqrt{(m + m'_c)}. \tag{12}$$

Now, for all connected graphs, $n \leq 2m \leq 2(m + m'_c)$.
Thus,

$$\sqrt{2n(m + m'_c)} \leq \sqrt{4(m + m'_c)^2}.$$

Taking positive square-root, we get

$$\sqrt{2n(m + m'_c)} \leq 2(m + m'_c).$$

Therefore, from Equation (5), we can write

$$E_c(G) \leq 2(m + m'_c) \tag{13}$$

and the result follows from Equations (12) and (13). □

Remark 7 *The inequality* $2\sqrt{(m + m'_c)} \leq E_c(G)$ *is true for disconnected graphs also.*

3 Relationship between color energy and energy of a graph

Although several aspects of color energy have been studied, relationship between color energy and energy was not taken into account. The color energy of K_n is $2(n - 1)$ which is same as its energy, whereas the color energy and the energy of $K_{1,n-1}$ are not same. So, it is interesting to find the relationship between color energy and energy and in this section an attempt is made to obtain this relationship.

Theorem 8 *If* G *is a graph of order* n *and size* m*, then*

$$[E_c(G)]^2 \geq E(G).$$

Proof. From Equations (1) and (5), we know that

$$2m \geq E(G) \text{ and } E_c(G) \geq \sqrt{2(m + m'_c) + n(n-1)D^{\frac{2}{n}}}.$$

Therefore,

$$\begin{aligned}[E_c(G)]^2 &\geq 2(m + m'_c) + n(n-1)D^{\frac{2}{n}} \\ &\geq 2(m + m'_c) \\ &\geq 2m \\ &\geq E(G).\end{aligned}$$

□

Next theorem tells us about the relation between $E_c(G), E(G)$ and the largest absolute value of the eigenvalue of the color matrix of G.

Theorem 9 *If* G *is a colored graph and* λ_{max} *is the largest absolute value of the eigenvalue of the color matrix of* G*, then* $E_c(G) \geq \dfrac{E(G)}{\lambda_{max}}$.

Proof. From Theorem 4, we have

$$E_c(G) \geq \left(\frac{2(m + m'_c)}{\lambda_{max}}\right).$$

Thus,

$$\begin{aligned}\lambda_{max} E_c(G) &\geq 2(m + m'_c) \\ &\geq 2m \\ &\geq E(G), \text{ by Equation (1).}\end{aligned}$$

Therefore,

$$E_c(G) \geq \frac{E(G)}{\lambda_{max}}.$$

□

Computation of $E_c(G)$ and $E(G)$ have shown that $E(G) \leq E_c(G)$ and hence we state the following conjecture.

Conjecture 10 *If* G *is a graph, then*

$$E(G) \leq E_c(G).$$

4 Non-co-spectral color-equienergetic graphs

The color-co-spectral graphs are the graphs having same color eigenvalues [1]. Obviously these graphs are color-equienergetic. It is interesting to note that K_n and $K_{1,n-1}$ graphs are color-equienergetic with $E_\chi = 2(n-1)$. However, they are not color-co-spectral.

In this section, we introduce a new family of unicyclic graphs which is color-equienergetic with some families of graphs and a family of bicyclic graphs which is color-hyperenergetic. Also we present explicit formulas for color energies of these graphs.

Theorem 11 *If $\mathscr{S} = K_{1,n-1} + e$ is a unicyclic graph of order n and size m obtained by adding a single edge between two pendant vertices of the star graph $K_{1,n-1}$, then $E_\chi(\mathscr{S}) = 2(n-1)$.*

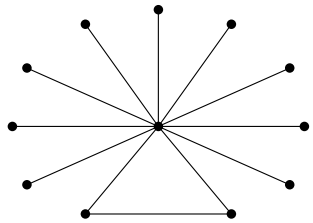


Figure 2: $\mathscr{S} = K_{1,n-1} + e$

Proof. $\mathscr{S}$ is a unicyclic graph of order n with $n-3$ pendant vertices. $\chi(\mathscr{S}) = 3$, as it contains a C_3.

The color matrix of $\mathscr{S}$ of order $n \times n$,

$$A_\chi(\mathscr{S}) = \begin{pmatrix} 0 & 1 & 1 & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & \dots & 1 & 1 \\ -1 & 0 & 1 & 0 & -1 & -1 & \dots & -1 & -1 \\ -1 & 0 & 1 & -1 & 0 & -1 & \dots & -1 & -1 \\ -1 & 0 & 1 & -1 & -1 & 0 & \dots & -1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & 0 & 1 & -1 & -1 & -1 & \dots & 0 & -1 \\ -1 & 0 & 1 & -1 & -1 & -1 & \dots & -1 & 0 \end{pmatrix}_{n\times n}$$

The characteristic polynomial

$$P_\chi(\mathscr{S}, \lambda) = \det(\lambda I - A_\chi(\mathscr{S})).$$

That is,

$$P_\chi(\mathscr{S},\lambda)=\begin{vmatrix} \lambda & -1 & -1 & 1 & 1 & 1 & \dots & 1 & 1 \\ -1 & \lambda & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & -1 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & -1 & \lambda & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & \lambda & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & 1 & \lambda & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & \lambda & 1 \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & 1 & \lambda \end{vmatrix}_{n\times n}$$

In order to get the characteristic polynomial we apply a series of row transformations.

Adding first three rows of $P_\chi(\mathscr{S},\lambda)$ to take the factor $(\lambda-2)$ out of the first row.

$$P_\chi(\mathscr{S},\lambda)=(\lambda-2)\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & \lambda & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & -1 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & -1 & \lambda & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & \lambda & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & 1 & \lambda & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & \lambda & 1 \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & 1 & \lambda \end{vmatrix}_{n\times n}.$$

Adding first row of $P_\chi(\mathscr{S},\lambda)$ to its second row and taking the factor $(\lambda+1)$ out of the second row, we get

$$P_\chi(\mathscr{S},\lambda)=(\lambda-2)(\lambda+1)\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & -1 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & -1 & \lambda & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & \lambda & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & 1 & \lambda & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & \lambda & 1 \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & 1 & \lambda \end{vmatrix}_{n\times n}.$$

Now adding second row of $P_\chi(\mathscr{S},\lambda)$ to its third row, we get

$$P_\chi(\mathscr{S},\lambda) = (\lambda-2)(\lambda+1)\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & -1 & \lambda & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & \lambda & 1 & \dots & 1 & 1 \\ 1 & 0 & -1 & 1 & 1 & \lambda & \dots & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & \lambda & 1 \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & 1 & \lambda \end{vmatrix}_{n\times n}.$$

Adding the third row of $P_\chi(\mathscr{S},\lambda)$ to its q^{th} row where $q \geq 4$ and taking the factor $(\lambda-1)$ common from q rows, we get

$$P_\chi(\mathscr{S},\lambda)=(\lambda-2)(\lambda+1)(\lambda-1)^{(n-3)}\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 0 & 0 & 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \end{vmatrix}_{n\times n}.$$

Adding the first row and q rows of $P_\chi(\mathscr{S},\lambda)$ to its third row and subtracting the second row from its third row, then take the factor $[\lambda+(n-2)]$ out of the third row

$$P_\chi(\mathscr{S},\lambda)=(\lambda-2)(\lambda+1)(\lambda-1)^{(n-3)}[\lambda+(n-2)]\begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \end{vmatrix}_{n\times n}.$$

Now subtracting the second and third row of $P_\chi(\mathscr{S}, \lambda)$ from its first row and subtract the third row from q^{th} row where $q \geq 4$, we get

$$P_\chi(\mathscr{S}, \lambda) = (\lambda - 2)(\lambda + 1)(\lambda - 1)^{(n-3)}[\lambda + (n-2)] \det(I)$$

where I is the identity matrix.

Thus,

$$P_\chi(\mathscr{S}, \lambda) = (\lambda - 2)(\lambda + 1)(\lambda - 1)^{(n-3)}[\lambda + (n-2)].$$

Therefore,

$$\mathrm{Spec}_\chi\, \mathscr{S} = \begin{pmatrix} -(n-2) & -1 & 1 & 2 \\ 1 & 1 & n-3 & 1 \end{pmatrix}. \tag{14}$$

Hence, $E_\chi(\mathscr{S}) = 2(n-1)$. □

Remark 12 *We observe that, the family of graphs* $\mathscr{S}, F_3^{(k)}, K_n$ *and* $K_{1,n-1}$ *are color-equienergetic. From Equations (4), (6) and (14), clearly we can see their spectra are not same. So, these families of graphs are non-co-spectral color-equienergetic.*

It is interesting to note that addition of an edge between a pendant vertex and a vertex of degree two to $\mathscr{S}$ brings a significant difference in its energy.

Theorem 13 *If* $\mathscr{H} = \mathscr{S} + e$ *is a bicyclic graph of order* n *and size* m *obtained by adding an edge between a pendant vertex and a vertex of degree two of the graph* $\mathscr{S} = K_{1,n-1} + e$*, then* $E_\chi(\mathscr{H}) = 2(n - 3 + \sqrt{5})$.

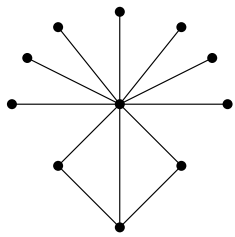


Figure 3: $\mathscr{H} = \mathscr{S} + e$

Proof. $\mathscr{H}$ is bicyclic graph of order n with $n - 4$ pendent vertices. Thus, $\chi(\mathscr{H}) = 3$, as it contains two copies of C_3 with two vertices in common.

The color matrix of $\mathscr{H}$ of order $n \times n$,

$$A_\chi(\mathscr{H}) = \begin{pmatrix} 0 & 1 & 1 & -1 & -1 & -1 & \dots & -1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & -1 \\ 1 & 1 & 0 & 1 & 1 & 1 & \dots & 1 & 1 \\ -1 & 0 & 1 & 0 & -1 & -1 & \dots & -1 & 0 \\ -1 & 0 & 1 & -1 & 0 & -1 & \dots & -1 & 0 \\ -1 & 0 & 1 & -1 & -1 & 0 & \dots & -1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -1 & 0 & 1 & -1 & -1 & -1 & \dots & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}_{n\times n}$$

The characteristic polynomial

$$P_\chi(\mathscr{H}, \lambda) = \det(\lambda I - A_\chi(\mathscr{H})).$$

That is,

$$P_\chi(\mathscr{H}, \lambda) = \begin{vmatrix} \lambda & -1 & -1 & 1 & 1 & 1 & \dots & 1 & -1 \\ -1 & \lambda & -1 & 0 & 0 & 0 & \dots & 0 & 1 \\ -1 & -1 & \lambda & -1 & -1 & -1 & \dots & -1 & -1 \\ 1 & 0 & -1 & \lambda & 1 & 1 & \dots & 1 & 0 \\ 1 & 0 & -1 & 1 & \lambda & 1 & \dots & 1 & 0 \\ 1 & 0 & -1 & 1 & 1 & \lambda & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & -1 & 1 & 1 & 1 & \dots & \lambda & 0 \\ -1 & 1 & -1 & 0 & 0 & 0 & \dots & 0 & \lambda \end{vmatrix}_{n\times n}$$

Thus,

$$P_\chi(\mathscr{H}, \lambda) = [\lambda + (n-3)](\lambda + \sqrt{5})(\lambda - \sqrt{5})(\lambda - 1)^{(n+1)}$$

Therefore,

$$\mathrm{Spec}_\chi\, \mathscr{H} = \begin{pmatrix} -(n-3) & -\sqrt{5} & 1 & \sqrt{5} \\ 1 & 1 & (n-3) & 1 \end{pmatrix}.$$

Hence, $E_\chi(\mathscr{H}) = 2(n - 3 + \sqrt{5})$.

□

Remark 14 *The above theorem gives us the color energy of the family of graphs $\mathscr{H} = \mathscr{S} + e$ of order n which is $2(n - 3 + \sqrt{5})$. We observe that $E_\chi(\mathscr{H}) > 2(n-1)$. As a color-hyperenergetic graph has the color energy greater than $2(n - 1)$ [1], $\mathscr{H}$ is a family of color-hyperenergetic graphs.*

5 Concluding remarks and scope

In this study, we have explored new bounds for the color energy of graphs and have been successful in finding better bounds than those found in the literature. However, it remains as an open problem to determine graphs for which these bounds are sharp. Further, new bounds for $E_c(G)$ in terms of its order n can be determined. Another interesting area would be explore the relation between $E_c(G)$ and topological indices, and identify graphs for which they match. As we have observed the color energy is defined in terms of its color matrix which of course depends upon the coloring scheme. Therefore, study of color energy with respect to a specific coloring is yet another area to be investigated.

References

[1] C. Adiga, E. Sampathkumar, M. A. Sriraj, Shrikanth A. S., Color energy of a graph, *Proc. Jangjeon Math. Soc.*, **16,** 3 (2013) 335–351. ⇒119, 120, 127, 131

[2] K. S. Betageri, The reduced color energy of graphs, *J. Comp. and Math. Sci.* **7,** 1 (2016) 13–20. ⇒121

[3] K. S. Betageri and G. H. Mokashi, A note on reduced color energy of graphs, *J. Comp. and Math. Sci.* **7,** 4 (2016) 203–212. ⇒121

[4] P. G. Bhat and S. D'souza, Color Laplacian energy of a graph, *Proc. Jangjeon Math. Soc.* **18,** 3 (2015) 321–330. ⇒121

[5] P. G. Bhat and S. D'souza, Color signless laplacian energy of graphs, *AKCE Int. J. of Graphs and Comb.* (2017) http://dx.doi.org/10.1016/j.akcej.2017.02.003. ⇒121

[6] N. Biggs, *Algebraic Graph Theory*, Cambridge Univ. Press, Cambridge, UK, 1993. ⇒121

[7] S. A. Choudum, Graph theory, A NPTEL Course, (2012) http://nptel.ac.in/courses/111106050. ⇒121

[8] D. M. Cvetković, M. Doob, H Sachs, *Spectra of Graphs- Theory and Application*, Academic Press, New York, 1980. ⇒121

[9] I. Gutman, The energy of a graph, *Ber. Math. Stat. Sekt. Forschungsz. Graz*, **103** (1978) 1–22. ⇒119, 120

[10] I. Gutman, The energy of a graph: old and new results, *Combinatorics and Applications*, A. Betten, A. Khoner, R. Laue and A. Wassermann, eds., *Springer*, Berlin, (2001) 196–211. ⇒120

[11] X. Li, Y. Shi and I. Gutman, *Graph Energy*, Springer, New York, 2012. ⇒120

[12] M. R. Rajesh Kanna, R. Pradeep Kumar and M. R. Farhani, Specific energies of friendship graph, *Asian Acad Res J Multidiscip.* **3,** 1 (2016) 189–196. ⇒121

[13] M. R. Rajesh Kanna, R. Pradeep Kumar and R. Jagadeesh, Minimum covering color energy of a graph, *Asian Acad Res J Multidiscip.* **9,** 8 (2015) 351–364. ⇒ 121
[14] R. Rehman and I. C. F. Ipsen, Computing characteristic polynomials from eigenvalues, *SIAM J. Matrix Anal. Appl.* **32,** 1 (2011) 90–114. ⇒123
[15] V. S. Shigehalli and K. S. Betageri, Color laplacian energy of graphs, *J. Comp. and Math. Sci.* **6,** 9 (2015) 485–494. ⇒121
[16] V. S. Shigehalli and K. S. Betageri, A note on color energy graphs, *Bull. Math. and Stat. Res.* **4,** 4 (2016) 47–50. ⇒121
[17] V. S. Shigehalli and K. S. Betageri, A note on color energy and color laplacian energy of graphs, *Int. J. Math. Arch.* **7,** 10 (2016) 199–204. ⇒121
[18] D. B. West, *Introduction to Graph Theory*, Pearson, New Jersey, 2001. ⇒121

Received: August 28, 2017 • Revised: November 21, 2017

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 134–143

DOI: 10.1515/ausi-2017-0009

An optical solution for the set splitting problem

Mihai OLTEAN
1 Decembrie 1918 University of Alba Iulia
Alba Iulia, Romania
email: mihai.oltean@gmail.com
https://mihaioltean.github.io/optical

Abstract. We describe here an optical device, based on time-delays, for solving the set splitting problem which is well-known NP-complete problem. The device has a graph-like structure and the light is traversing it from a start node to a destination node. All possible (potential) paths in the graph are generated and at the destination we will check which one satisfies completely the problem's constrains.

1 Introduction

Recently, an increased number of difficult (most of them being NP-complete [3]) problems have been proposed to be solved by using optical devices. Hamiltonian path [2, 25, 29, 30], traveling salesman [2, 13, 14], subset sum [2, 17, 23, 26, 22], exact cover [22, 27], Diophantine equations [24, 22], 3-SAT [2, 4, 8, 18], SAT [6, 9, 11, 28], dominating set [7], prime factorization [20, 21], security [16], independent sets [19], graph colorability [5], k-clique [12], ultrafast arithmetic [15], abstract machines [10] are just few of the problems whose solution can be computed by using an optical device.

Here we show how to solve another problem, namely set splitting, which is an known NP-complete problem [3]. The underlying mechanism is to use

Computing Classification System 1998: F.1.1
Mathematics Subject Classification 2010: 68Q05, 68W10
Key words and phrases: optical computing, set splitting, NP-complete

delays for encoding possible solutions. The approach here is derived from the solution for the subset sum problem [26]. The novelty consists in a special set of delays attached to each arc and a special set of moments when the solution is expected in the destination node.

The paper is organized as follows: Section 2 contains a description of the problem to be solved. Properties of the signal useful for our research and the operations performed on that signal are described in section 3. Section 4 deeply describe the proposed device. Details about the physical implementation are given in section 5. Complexity is discussed in section 6. The size of the instances that can be solved with this method is computed in section 7. A short discussion of the weaknesses of this method is given in section 8. Section 9 concludes our paper.

2 The set splitting problem

"Given a family F of subsets of a finite set A, decide whether there exists a partition of A into two subsets A_1, A_2 such that all elements of F are split by this partition, i.e., none of the elements of F is completely in A_1 or A_2." [3, 31].

Set splitting is a NP-complete problem. No polynomial-time algorithm is known to exist for it.

The optical solution of the set splitting is based on the solution for the subset sum problem whose definition is given below:

Given a set of positive numbers $A = \{a_1, a_2, ..., a_n\}$ and another positive number B. Is there a subset of A whose sum equals B? [3]

3 Time-delay systems

Time-delay systems have been proposed in [25]. They are based on 2 properties of the signals (optical, electrical, etc):

- The signal can be delayed by forcing it to pass through a cable of a certain length.
- The signal can be easily divided into multiple signals of smaller intensity/power which run within the same cables. This ensures a high parallelism of the method.

The proposed device has a graph-like structure with a start and destination node. Nodes are connected through cables made of optical fiber.

The system works as follows: A beam of light is sent to the start node. The beams travel through arcs and is divided inside nodes. Because arcs have a length greater than 0, the light is delayed by them. At the destination node we will have different signals arriving at various moments of time. One of them will tell us if we have a solution to our problem.

Since we work with continuous signal we cannot expect to have discrete output at the destination node. This means that beams arrival is notified by fluctuations in the intensity of the light. These fluctuations will be transformed, by a photodiode, in fluctuations of the electric power which will be read by an oscilloscope.

4 The optical device for the set splitting problem

This section deeply describes the proposed system.

Since the solution for set splitting is based on the solution from the subset sum, we first briefly describe (see section 4.1) the device which solves the subset sum problem. This device has been proposed in [26]. Later, in section 4.2 we describe the graph for the set splitting problem.

4.1 The graph for the subset sum problem

As described in [26] the numbers from the given set A represent the delays induced to the signals (light) that passes through the device. For instance, if numbers a_1, a_3 and a_7 generate the expected subset, then the total delay of the signal should be $a_1 + a_3 + a_7$. Recall that, when using light, we can easily induce some delays by forcing the beam to pass through an optical cable of a given length.

The device has been designed [26] as a simple directed graph. There are $n + 1$ nodes connected by arcs. In each node (but the last one) we place a beam-splitter which will split a beam into 2 sub-beams of smaller intensity. Arcs are implemented by using optical cables and have lengths proportional to the numbers from the given set A. A mechanism for skipping a value from the given set is also needed. A possible way for achieving this is to add an extra arc, of length 0, between any pair of consecutive nodes. A beam leaving a node will have the possibility to either traverse an arc representing a number from the given set or to skip it (by traversing the arc of 0 length).

The graph for solving the subset sum problem is depicted in Figure 1.

In the destination node we will have signals (fluctuation in the intensity of the signal) arriving at moments representing the sum of elements of all

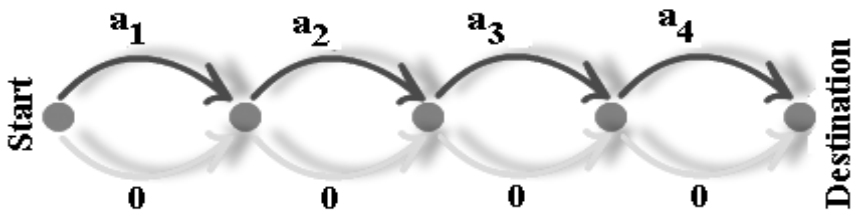


Figure 1: The device for solving the subset sum problem. Each arc delays the beam by the amount of time written on it. Image taken from [26].

subsets. For instance if the lights travels through the top arcs only we will have the subset containing all elements: $a_1, a_2, ..., a_n$. If the light travels through bottom arc only we will have the empty subset. Thus the device will generate all possible subsets of A. Each subset will delay one of the beams by an amount of time equal to the sum of the lengths of the arcs in that path. If there is a fluctuation in the intensity of the signal at moment B it means that we have a solution to our problem.

Note that in practice we cannot have cables of length 0 in practice, thus we add a small ϵ to the length of all arcs and the final solution is expected at moment $B + n * \epsilon$.

4.2 The graph for the set splitting problem

We have seen that the device described in section 4.1 generates all possible subsets of the given set A, so theoretically we could use it for the set splitting problem. But, there are some issues:

- In the case of subset sum we already have some values for the elements in set A. Here, in the set splitting, we have no predefined values in A.
- If 2 values from the set A are identical, it is not possible to uniquely identify a subset at the destination node.

In order to overcome these issues we introduce here a new set of delays. The main attribute of this delaying system is to allow a unique identification of the generated subsets.

The smallest set of n numbers such that each subset has a different value for the sum of its elements is the set of the first n powers of 2: $A = \{2^0, 2^1, \dots 2^{n-1}\}$.

Thus, we assign the values $\{2^0, 2^1, \dots 2^{n-1}\}$ as lengths for the top arcs.

The graph for solving the set splitting problem is depicted in Figure 2.

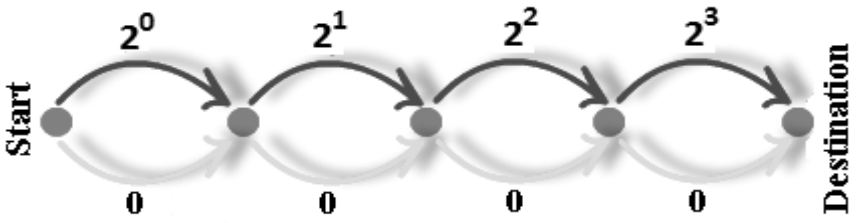


Figure 2: The device for solving the set splitting problem. Each arc delays the beam by the amount of time written on it.

Now we can easily identify a subset at the destination node: having the moment of arrival (name it k) we know exactly what elements belong to that subset (because each number (k) is uniquely decomposed as sum of powers of 2).

Knowing the subset, we can easily find if it represents or not a solution to our problem. Let's denote by A_1 the subset that has just arrived in the destination node. We also compute its complement: $A_2 = A - A_1$.

Now, we want to know if the split of A into sets (A_1 and A_2) represents or not a solution to our problem, that is, whether one of the sets in F is completely included in either A_1 or A_2. For this we check if any of them (A_1 or A_2) includes any set from F. If they do, it means that the corresponding split does not represent a solution for our problem.

We can check the inclusion property very easily by working only with delay moments: for each set in F we compute and cache the moments when any superset (a set that includes the current one) of it arrives at the destination node. Let's denote by M those moments. If a signal arrives at a moment not belonging to M, that set (who generated the signal) is a solution for our problem. If M contains all integer numbers between 0 and $2^n - 1$ it means that the instance does not have a solution.

We can do a small optimization here: we want to minimize the number of moments when we wait for a solution at the destination node. So, we check the size of M. If it is less than 2^{n-1} we wait for numbers belonging to M. Otherwise we wait for numbers not belonging to M.

4.3 Example of solution

Let's suppose that we have a set A made from 4 elements: $A = \{a_1, a_2, a_3, a_4\}$ and a set of subsets $F = \{F_1 = \{a_1, a_2\}, F_2 = \{a_1, a_3\}\}$. We want to find if it is

possible to split A into 2 subsets (A_1 and A_2) such that neither F_1 nor F_2 is completely included in A_1 or A_2.

To each number from A we assign a power of 2 as follows: $a_1 = 2^0, a_2 = 2^1, a_3 = 2^2$ and $a_4 = 2^3$.

The supersets which includes F_1 are:

$$\{\{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}, \{a_1, a_2, a_3, a_4\}\}$$

which have the following delay times: $M_1 = \{3, 7, 11, 15\}$ (which were computed as sum of elements).

The supersets including F_2 are:

$$\{\{a_1, a_3\}, \{a_1, a_3, a_2\}, \{a_1, a_3, a_4\}, \{a_1, a_3, a_2, a_4\}\}$$

which have the following delay times: $M_2 = \{5, 7, 13, 15\}$.

Set M which represents the union of M_1 with M_2 is $M = \{3, 5, 7, 11, 13, 15\}$.

If a beam of light arrives in the destination at any of the moments from M it does not encode a subset representing a solution to our problem. However, if a beam arrives at another moment in the integer interval [1...15] but does not belong to M, it represents a solution to our problem. For instance, if a beam arrives at moment 1 it represents a solution because it encodes the split $A_1 = \{a_1\}$ and $A_2 = \{a_2, a_3, a_4\}$ and none of the sets in F is included into either A_1 or A_2.

5 Physical implementation

As discussed in [26] for the physical implementation we need the following components:

- A source of light (laser),
- Several beam-splitters for dividing light beams into 2 sub-beams.
- A high speed photodiode for converting light into electrical power. The photodiode is placed in the destination node,
- A tool for detecting fluctuations in the intensity of electric power generated by the photodiode (oscilloscope),
- A set of optical fiber cables having lengths proportional with the numbers $2^0, 2^1, \ldots 2^{n-1}$ (plus constant ϵ) and another set of n cables having fixed length ϵ.

6 Build and running complexity

The time required to build the device has $\theta(n * 2^n)$ complexity because we have to build n arcs having at most 2^{n-1} length.

The running time complexity is $\theta(2^n)$.

The intensity of the signal decreases exponentially with the number of nodes (because it is divided in 2 in each node). This is why the required power is proportional to 2^n (exponential).

7 Instance size

We want to find the size of the instances (how many numbers can we have in set A) that can be solved by our device.

For computing this number we use the same reasoning as for the Hamiltonian Path problem [30] because in both cases the delays are powers of 2. (Note that only delays are similar with Hamiltonian Path solution. The devices (graphs) are extremely different.)

Assuming that the best oscilloscope available on the market has the rise-time in the range of picoseconds (10^{-12} seconds - this is also the smallest difference between 2 consecutive moments when 2 solution may arrive in the destination) and also knowing the speed of light ($3 \cdot 10^8 m/s$) we can compute the minimal cable length that should be traversed by the light in order to be delayed by 10^{-12} seconds. This is 0.0003 meters [25].

This length is for the shortest cable. All other cables have lengths obtained by multiplying this shortest length with powers of 2.

It is easy to compute the maximal number of nodes that a graph can have in order to have the total delay equal by 1 second. This results from the equation (taken from [25]):

$$2^n \cdot 0.0003 = 3 \cdot 10^8$$

This gives us a number of nodes equal to 39 nodes, so in one second we can solve instances having 39 numbers. However, the length of the optical fibers used for inducing the largest delay for this graph is huge: about $8 \cdot 10^8$ meters. We cannot expect to have such long cables for our experiments [25].

But, shorter cables (of several hundreds of kilometers) are already available in the internet networks. They can be easily used for our purpose. Assuming that the longest cable that we have is about 300 kilometers we may solve instances with about 26 numbers. The amount of time required to obtain a solution is about 10^{-6} seconds [25].

The maximum number of nodes can be increased by increasing the precision of the instruments (oscilloscope, photodiode etc).

In [1] the authors have solved with DNA another problem with a 2^n complexity, namely the SAT problem. The largest instance solved was with 20

nodes. Here we have shown that theoretically we can solve an instance with 26 nodes with the available resources. This is with almost 2 orders of magnitude larger that the DNA solution presented in [1] for the SAT problem.

8 Weaknesses of the solution

The proposed device has some weaknesses:

- it contains exponential delays,
- the number of possible moments when a solution can appear is exponential in the number of elements in the initial set. This is different compared to subset sum (see [26]) where only one moment was enough for detecting the solution.

9 Conclusion

We have shown how the set splitting problem can be attacked with an optical device. The solution was derived from the graph for the the subset sum problem by adding a new set of delays and a new set of moments when the solution is expected at the destination node. Weak points of the proposed solution are: the use of optical cables of exponential length, and the exponential amount of energy required to power the device.

References

[1] R.S. Braich, N. Chelyapov, C. Johnson, P. Rothemund, L. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer , *Science* **296,** 5567 (2002) 499–502. ⇒140, 141

[2] S. Dolev, H. Fitoussi, The traveling beams optical solutions for bounded NP-complete problems, *Int. Conf. on Fun with Algorithms, FUN 2007*, *LNCS* **4475**, pp. 120–134, 2007. ⇒134

[3] M. R. Garey, D. S. Johnson, *Computers and intractability: A guide to NP-Completeness*, Freeman & Co, San Francisco, CA, 1979. ⇒134, 135

[4] S. Goliaei. S. Jalili, An optical wavelength-based solutionto the 3-SAT problem, *Int. Workshop on Optical Supercomputing OSC 2009*, *LNCS* **5882,** pp. 77–85, Springer-Verlag Heidelberg, 2009. ⇒134

[5] S. Goliaei. S. Jalili, Optical graph 3-colorability, *Int. Workshop on Optical Supercomputing, OSC 2010 LNCS* **6748,** pp. 16–22, Springer-Verlag Heidelberg, 2011. ⇒134

[6] S. Goliaei, M-H. Foroughman-Araabi, Lower bounds on the complexity of the wavelength-based machine, *Int. Conference on Unconventional Computing and Natural Computation, UCNC 2012 LNCS* **7445,** pp. 94–105, Springer-Verlag Heidelberg, 2012. ⇒134

[7] S. Goliaei, S. Jalili, J. Salimi, Light-based solution for the dominating set problem, *Applied Optics* **51,**, 29 (2012) 6979–6983. ⇒134

[8] S. Goliaei, S. Jalili, An optical solution to the 3-SAT problem using wave-length based selectors, *Int. Journal of Supercomputing* **62,** 2 (2012) 663–672. ⇒134

[9] S. Goliaei, S. Jalili, An optical polynomial time solution for the satisfiability problem, *Int. Workshop on Optical Supercomputing, OSC 2012*, *LNCS* **7715,** pp. 15–24, Springer-Verlag Heidelberg, 2013. ⇒134

[10] S. Goliaei, M-H. Foroughman-Araabi, Light ray concentration reduces the complexity of the wavelength-based machine on PSPACE languages, *Int. Conf. on Unconventional Computing and Natural Computation, UCNC 2013*, *LNCS* **7956,** pp. 90–101, Springer-Verlag Heidelberg, 2013. ⇒134

[11] S. Goliaei, M-H. Foroughman-Araabi, On the complexity of nonuniform wavelength-based machine, *Natural Computing* **13,** 2 (2014) 269–283. ⇒134

[12] S. Goliaei, S. Jalili, Computation with optical sensitive sheets, *Natural Computing* **14,** 3 (2015) 437–450. ⇒134

[13] T. Haist, W. Osten, An optical solution for the traveling salesman problem, *Opt. Express* **15,** 16 (2007) 10473–10482. ⇒134

[14] T. Haist, W Osten, An optical solution for the traveling salesman problem:erratum, *Opt. Express* **15,** 20 (2007) 12627–12627. ⇒134

[15] T. Haist, W. Osten, Ultrafast digital-optical arithmetic using wave-optical computing, *Int. Workshop on Optical Supercomputing, OSC 2008*, *LNCS* **5172,** pp. 33–45, 2008. ⇒134

[16] T. Haist, W. Osten, Proposal for secure key distribution using classical optics, *Int. Workshop on Optical Supercomputing, OSC 2009, LNCS* **5882,** pp. 99–101, 2009. ⇒134

[17] R. Hasan, S. Rahman, Computing a solution for the subset sum problem with a light based device, *Int. Workshop on Optical Supercomputing, OSC 2009, LNCS* **5882,** pp. 70–76, 2009. ⇒134

[18] T. Head, Parallel computing by xeroxing on transparencies, *Algorithmic Bioprocesses*, pp. 631–637, 2009. ⇒134

[19] T. Head, Computing transparently: the independent sets in a graph, *Natural Computing*, **10,** 1 (2011) 129-138. ⇒134

[20] K. Nitta, N. Katsuta, O. Matoba, Improvement of a system for prime factorization based on optical interferometer, *Int. Workshop on Optical Supercomputing, OSC 2009, LNCS* **5882,** pp. 124–129, 2009. ⇒134

[21] K. Nitta, N. Katsuta, O. Matoba, A method for modulo operation by use of spatial parallelism, *Int. Workshop on Optical Supercomputing, OSC 2008, LNCS* **5172,** pp. 98–103, 2008. ⇒134

[22] O. Muntean, *Optical Solutions for NP-complete problems* (graduation thesis), Faculty of Mathematics and Computer Science, Babeş–Bolyai University, Cluj-Napoca, Romania (defended 3rd of July, 2007). ⇒134

[23] O. Muntean, M. Oltean, Using light for solving the unbounded subset-sum problem, *Int. J. of Innovative Computing, Information and Control* **5,** 8 (2009) 2159–2167. ⇒134

[24] O. Muntean, M. Oltean, Deciding whether a linear Diophantine equation has solutions by using a light-based device, *J. Optoelectronics and Advanced Materials*, **11,** 11 (2009) 1728–1734. ⇒134

[25] M. Oltean, A light-based device for solving the Hamiltonian path problem, *Int. Conf. on Unconventional Computation, UC 2006, LNCS* **4135,** Springer-Verlag, pp. 217–227, 2006. ⇒134, 135, 140

[26] M. Oltean M., O. Muntean, Solving the subset-sum problem with a light-based device, *Natural Computing*, **8,** 2 (2009) 321–331. ⇒134, 135, 136, 137, 139, 141

[27] M. Oltean, O. Muntean, Exact cover with light, *New Generation Computingn*, **26,** 4 (2008) 327–344. ⇒134

[28] M. Oltean, O. Muntean, An optical solution for the SAT problem, *Int. Workshop on Optical Supercomputing, OSC 2010*, *LNCS* **6748,** pp. 53–62, 2010. ⇒134

[29] N. T. Shaked, S. Messika, S. Dolev, J. Rosen, Optical solution for bounded NP-complete problems, *Applied Optics* **46,** 5 (2007) 711–724. ⇒134

[30] N. Shakeri, S. Jalili, V. Ahmadi, A. R. Zali, S. Goliaei, A 2-dimensional optical architecture for solving Hamiltonian path problem based on micro ring resonators, *Optics & Laser Technology*, **65,** 1 (2015) 56–65. ⇒134, 140

[31] ∗∗∗ Set splitting problem on Wikipedia, `https://en.wikipedia.org/wiki/Set_splitting_problem`, last accessed on 2017.10.09 ⇒135

Received: November 7, 2017 • Revised: November 17, 2017

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 144–161

DOI: 10.1515/ausi-2017-0010

Object-oriented backtracking

Tibor GREGORICS
ELTE, Eötvös Loránd University
Budapest
email: gt@inf.elte.hu

Abstract. Several versions of the backtracking are known. In this paper, those versions are in focus which solve the problems whose problem space can be described with a special directed tree. The traversal strategies of this tree will be analyzed and they will be implemented in object-oriented style. In this way, the traversal is made by an enumerator object which iterates over all the paths (partial solutions) of the tree. Two different "backtracking enumerators" are going to be presented and the backtracking algorithm will be a linear search over one of these enumerators. Since these algorithms consist of independent objects (the enumerator, the linear search and the task which must be solved), it is very easy to exchange one component in order to solve another problem. Even the linear search could be substituted with another algorithm pattern, for example, with a counting or a maximum selection if the task had to be solved with a backtracking counting or a backtracking maximum selection.

1 Introduction

The backtracking can solve the tasks that can be made possible to be viewed as a path-finding problem. In this case, the solution of a task is searched in a directed graph which may contain even infinite nodes but the number of the outgoing arcs of each node (i.e. branching factor) is finite. This is the so called

Computing Classification System 1998: F.3.1
Mathematics Subject Classification 2010: 68N30
Key words and phrases: path-finding problem, backtracking algorithm, enumerator, algorithm pattern

δ-graph [10]. The problem space of the task consists of the paths going out from the start node. Among these paths, the solution path must be found, which drives from the start node to any goal node.

There exist several versions of the backtracking algorithm [9] and always that one must be applied which best fits the special features of the directed graph describing the problem [7].

The most general version [5, 7, 10] can be used in arbitrary δ-graphs but it needs a so-called depth bound to terminate for sure. In this way, if there exists a solution path whose length is not greater than the depth bound, the search must find such a solution. However, the determination of the appropriate depth bound is not simple. If this depth bound is too small, a solution will not be found; if it is too big, the computational complexity will grow. The backtracking algorithms over only finite and acyclic directed graphs [5, 7, 10] do not need the depth bound to terminate and they can find solution if there is one.

The best-known versions work in a special directed tree (finite, rooted by the start node, with the same branching factor of the nodes being at the same level). Its branches make up the problem space of the task which is represented by this tree.

This paper focuses on these latest versions of the backtracking. In section 2, those tasks will be defined which can be solved with these versions and there it will be shown how the problem space of these tasks can be symbolized with a special so-called backtracking tree. section 4 presents two different methods that can traverse this backtracking tree in order to enumerate its branches (that is, the elements of the problem space). These traversals are based on well-known concepts but the separation of these traversals and the search will be novel. section 4 shows how these traversals can be implemented as an isolated object-oriented enumerator. In the section 5 a model will be sketched where the backtracking is a collaboration between three objects: a backtracking enumerator, a commonly linear search [3, 4], and the task that is wanted to solve. The advantages of this approach are discussed in section 6.

2 Model of backtracking tasks

The state-space representation is a well-known general modeling technique which can treat the tasks as a path-finding problem [5]. It requires a state space (the set of tuples of values of the essential data corresponding to the task), the initial and final states, and the operators over the state space. The solution of a task is the sequence of operators which can transform the initial

state to any final state. It is obvious that a state space representation can be mapped to a directed graph where the nodes represent the states, and the directed arcs symbolize the effects of the operators. In this way, in order to solve the problem it is enough to find a path which derives from the start node corresponding to the initial state to any goal node describing the final states. This is the solution path.

When a task has got particular features, the following modeling technique, which are a special state space representation, might be used [1, 6, 8, 9].

Definition 1 (Backtracking task) *There are given* n *finite sets (*$n \in \mathbb{N}$*):* $D_1, \ldots, D_n$*. Let us consider the Cartesian product* $D = D_1 \times \ldots \times D_n$*. The aim is to find the element of the set* D *that satisfies the statement* $\rho : D \to \mathbb{L}$ *(*$\mathbb{L} = \{\mathtt{false}, \mathtt{true}\}$*) where this statement can be defined by a series of statetments* $\rho_0, \rho_1, \ldots, \rho_n : D \to \mathbb{L}$ *with the conditions below:*

1. $\rho_0 \equiv \mathtt{true}$
2. $\rho_n \equiv \rho$
3. $\rho_0, \rho_1, \ldots, \rho_n$ *is monotone, i.e.* $\forall u \in D : \rho_i(u) \Rightarrow \rho_{i-1}(u) (i = 1, \ldots, n)$
4. $\rho_i(u)$ *depends on only the first* $i \in \{1, \ldots, n\}$ *components of* $u \in D$*, i.e.* $\forall i \in \{1, \ldots, n\}$ *and* $\forall u, v \in D : \rho_i(u) = \rho_i(v)$ *if* $\forall j, 1 \leq j \leq i : u_i = v_i$.

The tasks that can be modeled in this way are called backtracking tasks. Other path-finding problems might also be solved with backtracking but the backtracking tasks can be solved with a special version of the backtracking algorithm that will be defined below.

Many times, the series $\rho_0, \rho_1, \ldots, \rho_n$ can be defined so that $\forall i \in \{1, \ldots, n\} : \rho_i \equiv \rho_{i-1} \wedge \beta_i$ where $\beta : D \to \mathbb{L}$ and $\beta_i(u)$ depends on the first $i \in \{1, \ldots, n\}$ components of $u \in D$. In this way, the (3) and (4) conditions of the backtracking tasks automatically hold.

A classic example for this model is given by the n-queen problem where n queens must be placed onto an $n \times n$ chessboard without being able to attack each other. One queen attacks any piece in the same row, column and diagonals. Each row of the board must contain exactly one queen. The possible positions of the i^{th} queen put on the i^{th} row are included by the set $D_i = \{0, \ldots, n-1\}$. (As you can see, the columns are numbered from zero up to $n-1$. The reason for this will be clarified soon.) Let us fix that $\rho_0 \equiv \rho_1 \equiv \mathtt{true}$ and $\forall i \in \{2, \ldots, n\}$ and $\forall u \in D : \rho_i(u) = \rho_{i-1}(u) \wedge \forall j \in \{1, \ldots, i-1\} : (u_i \neq u_j \wedge |u_i - u_j| \neq |i - j|)$.

Definition 2 (Backtracking tree) *Let us consider a backtracking task. Its backtracking tree can be constructed in the following way. The nodes on the first level (children of the root) represent the elements of* D_1*. The nodes on the second level symbolize the elements of* $D_1 \times D_2$ *so that the first component of a node on this level is equal to its parent node. In general, the nodes on the* i^{th} *level (*$i = 1, \ldots, n$*) are the elements of* $D_1 \times \ldots \times D_i$*. The branching factor of the nodes on the* $i-1^{th}$ *level is the cardinality of the set* D_i*. The first* $i-1$ *components of a node on the* i^{th} *level give just the parent of this node; in other words, the parent node is the prefix of its children. The leaves of this tree are the elements of set* D*.*

According to this definition, a backtracking task can be treated as a path-finding problem in its backtracking tree. In order to solve this problem, it is sometimes enough to find the leaf which satisfies the statement ρ; sometimes that path (branch) which drives from the root (this is the start node) to the leaf satisfying the statement ρ (this is the goal node) must be sought.

Let m_i denote the cardinality of D_i for all $i \in \{1, \ldots, n\}$. Since the elements of D_i might be numbered from 0 up to m_{i-1}, these elements can be referred with their ordinal numbers.

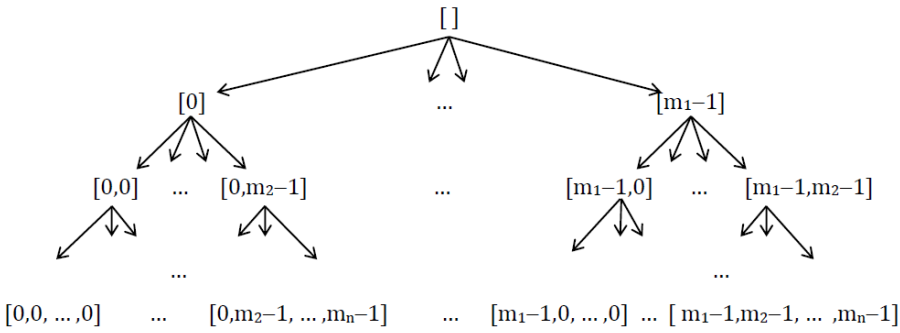


Figure 1: Backtracking tree

This serialization of the set D_i defines a bijection between the set D and the set $\{0, \ldots, m_1 \cdot m_2 \cdots m_n - 1\}$. One element of D can be mapped to a number in a positional numeral system in mixed bases. The base of the i^{th} digit of such numbers is $m_{i+1} \cdots m_n$ and the value of the i^{th} digit might be between 0 and $m_i - 1$ for all $i \in \{1, \ldots, n\}$. It is obvious that the value of an n-digit

natural number is between 0 and $m_1 \cdot m_2 \cdots m_n - 1$. Formally, if $\nu_i : D_i \to \{0, \ldots, m_{i-1}\}$ is a bijection ($i \in \{1, \ldots, n\}$), then $\nu : D \to \{0, \ldots, \prod_{i=1\ldots n} m_i\}$ where $\forall u \in D : \nu(u) = \sum_{i=1\ldots n} \nu_i(u_i) \cdot \prod_{j=i+1\ldots n} m_j$ is a bijection, too. Finally, let us denote ϕ the inverse of ν.

Thus each node on the i^{th} ($i \in \{1, \ldots, n\}$) level of the backtracking tree can be substituted with a code which is a natural number in a positional numeral system in mixed bases. The base of the j^{th} digit of this number is $m_{j+1} \cdots m_i$ and the value of the j^{th} digit might be between 0 and m_{j-1} for all $j \in \{1, \ldots, i\}$. The root node is labeled with the special blank.

This backtracking tree can be shown in the Figure 1.

3 Traversals of the problem space

3.1 Depth-first traversal

Perhaps the best known traversals method of the problem space of the backtracking tasks is the depth-first strategy. This strategy could not work only in special directed trees but also in general directed graphs. Nevertheless, we must underline that there is some difference between the well-known depth-first graph-traversal strategy [2] and the depth-first traversal of a backtracking[10]. Firstly, the latter one does not enumerate only the nodes of the graph but it searches for the first appropriate path driving from the start node. Secondly, the standard depth-first graph-traversal explicitly needs the whole graph which must be traversed but a backtracking algorithm uses much less memory: it stores only the current path. This property is very useful when a typical artificial intelligence problem must be solved and the graph of the problem space is so huge (sometimes infinite) that the total graph could not be stored explicitly. Thirdly, the backtracking could not record all nodes of the graph which have been touched earlier. Thus, a node which have been checked might be checked again when the algorithm rediscovers the very node via another path outgoing from the start node. Fortunately, this unpleasant phenomenon cannot occur if the graph is a tree as it can be seen in our case. In the following, the depth-first traversal means this memory-efficient version of the depth-first strategy.

The depth-first traversal of a directed tree means that the search systematically examines the paths outgoing from the root (these are the branches) from left to right. At each moment, only one partial branch (a path) is stored. This is the current path and its last node is the current node. In each step, the traversal tries to go forward (downward in the tree). If it is not able to or it is not worth going forward, it steps back (upward) to the parent node and selects

the next branch outgoing from this parent. If there are no other unchecked branches going from this parent, then it steps backward (upward) while it finds a parent with untested outgoing branches. In order to implement this process, the untested branches outgoing from the nodes of the current path must be recorded.

This depth-first traversal is even easier over the backtracking tree where the nodes are represented with numbers in a numeral system with mixed bases. For this traversal, it is enough to store only the current node that can be represented with an n-length array ν ($\nu : \mathbb{N}^n$) and the natural number $\mathtt{ind}$. The label of the current node is the $(\mathtt{ind}-1)$-length prefix of ν (i.e. $\nu[1, \ldots, \mathtt{ind}-1]$) and it is always supposed that the $\rho_{\mathtt{ind}-1}(\phi(\nu))$ holds and each element of ν after the position $\mathtt{ind}$ is zero. From this information, all outgoing untested branches of the current path can be read out.

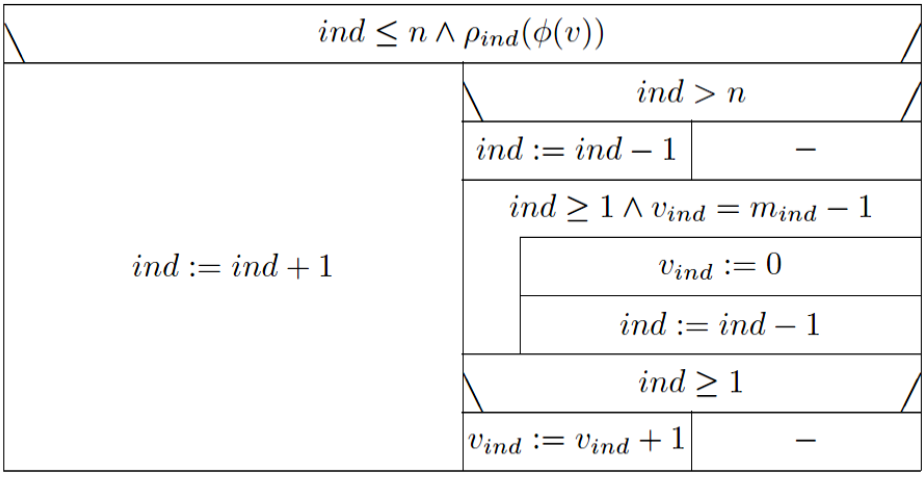


Figure 2: One step of the depth-first traversal

Initially the value of the number $\mathtt{ind}$ is 1 and each element of the array ν is zero. The next step of the traversal depends on the statement $\rho_{\mathtt{ind}}(\phi(\nu))$ (Figure 2). If it holds and $\mathtt{ind} \leq n$, then the traversal steps forward with the increase of ind. Otherwise, the traversal steps back – as if the part of the current branch below the $\mathtt{ind} - 1^{\text{th}}$ level had been cut – and it looks for the node on the current path that has got an unchecked successor and then selects the first such one as a new current node. The same happens when $\mathtt{ind} = n+1$,

although, in this case, the traversal does not need to continue since $\rho_n(\phi(v))$ holds, i.e. $\rho(\phi(v))$ holds. The traversal must stop definitely if $\mathtt{ind} = 0$. This event indicates that the traversal is over.

3.2 Increasing traversal

The problem space of the backtracking tasks in our focus can be traversed with another strategy. It is enough to enumerate the leaves of the backtracking tree (see Figure 1) from left to right. Since the nodes can be represented with natural numbers in a positional numeral system in mixed bases, each leaf is a natural number and they can be generated in increasing order. However, it is not worth enumerating them all one by one because those numbers which may not be goal nodes can be skipped. Yet, how can it be decided without their examination?

Let us suppose that the number v has been examined and the statement $\rho_{\mathtt{ind}-1}(\phi(v))$ holds but the statement $\rho_{\mathtt{ind}}(\phi(v))$ does not. It is obvious that each natural number whose first $\mathtt{ind}$ digits are identical to the first $\mathtt{ind}$ digits of v does not satisfy $\rho_{\mathtt{ind}}$, too. Thus, the enumeration might ignore these numbers. In order to get the next number of the enumeration, it is enough to increase the $\mathtt{ind}^{\text{th}}$ digit of v by one. In the case when $\rho(\phi(v))$ holds there is no index $\mathtt{ind}$ where array v can be changed, i.e. $\mathtt{ind} = n+1$). If the enumeration should be continued, the value of the variable $\mathtt{ind}$ must be changed to n.

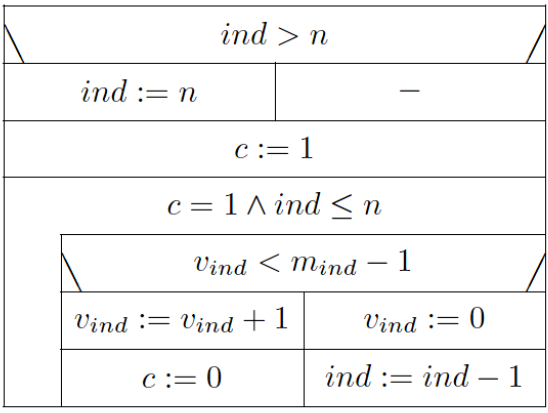


Figure 3: One step of the increasing traversal

This increasing is the task of the algorithm of the Figure 3 [6, 9]. This is a positional addition where variable $\mathtt{ind}$ indicates the position which must be increased. The variable $\mathtt{c}$ contains the carry digit of the addition. If this algorithm terminates with $\mathtt{c} = 0$, then the variable ν will contain the next element of the enumeration. The termination with $\mathtt{c} = 1$ indicates the overflow of the addition and it means that the enumeration is over.

This algorithm must be embedded into the environment which analyzes the current ν and looks for the first $\mathtt{ind}$ for which the statement $\rho_{\mathtt{ind}}(\phi(\nu))$ is false. (If there is no such $\mathtt{ind}$, then $\rho_n(\phi(\nu))$ and thus $\rho(\phi(\nu))$ is true.) This $\mathtt{ind}$ must be passed to the above algorithm.

3.3 Comparison of the traversals

It is worth comparing the two traversal techniques. Let us look at, for example, the problem space of the 4-queens problem (see Figure 4). In this tree, the depth-first traversal moves vertically while the increasing traversal enumerates a part of the leaves horizontally.

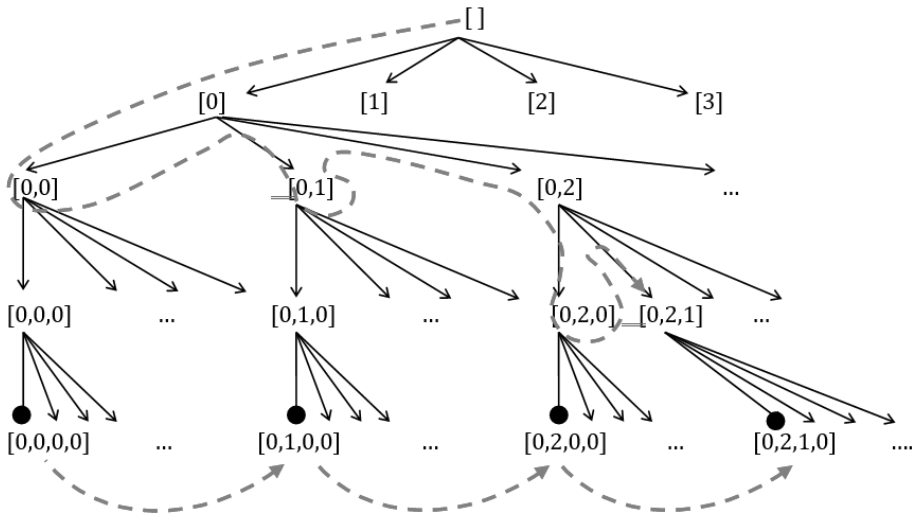


Figure 4: Two kinds of traversals over the 4-queen problem

The backtracking steps can be observed well in the depth-first traversal. If a number of a node at the level $\mathtt{ind}$ does not satisfy the statement $\rho_{\mathtt{ind}}$, i.e. $\rho_{\mathtt{ind}}(\phi(\nu))$ is false where ν is the label of the node, then the traversal

steps back. During the increasing traversal, the backtracking steps are totally hidden. When this enumeration selects a new leaf, then it might jump over several leaves. Each jump is corresponded to the series of backtracking steps of the depth-first traversal which selects the next branch for examination.

The depth-first traversal does not need the examination of ρ. It relies on the statements $\rho_{\mathtt{ind}}$ only. The algorithm of the increasing traversal does not use directly the statements $\rho_{\mathtt{ind}}$ but it is supposed that, before calling this algorithm, the $\rho(\phi(v))$ has been examined. If it is false, the statement $\rho_{\mathtt{ind}-1}(\phi(v))$ holds but the statement $\rho_{\mathtt{ind}}(\phi(v))$ does not; this $\mathtt{ind}$ must be given to the algorithm as input parameter beside the v.

4 Backtracking enumerators

Now the above traversals are going to be implemented as independent enumerator objects. These enumerators iterate over the elements of the problem space.

The problem space ($D = D_1 \times \ldots \times D_n$) can be modeled by the class `Task` (see Figure 5). This class provides the method `rho()` which can decide whether an element satisfies ρ_i or does not. Certainly, this method is abstract; it must be overridden when the concrete task becomes known. A task can be represented by the pair of array v and array m. These are the members of the class `Task`. The array v contains one element of the set of D. The $m[i]$ gives upper limit of the elements of $v[i]$, i.e. for all $i \in [1, \ldots, n] : 0 \leq v[i] < m[i]$. This is the invariant of this representation.

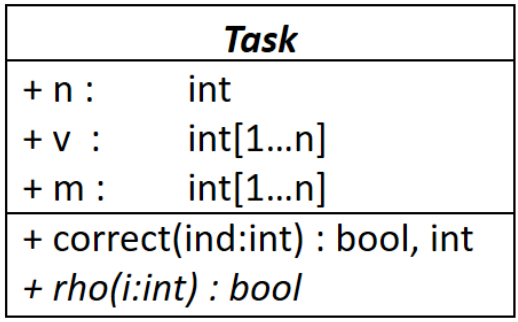


Figure 5: Abstract class of the backtracking task

This class is extended with the method `correct()` that decides whether the

current element of the problem space satisfies the statement $\rho(\phi(u.v))$ or does not.

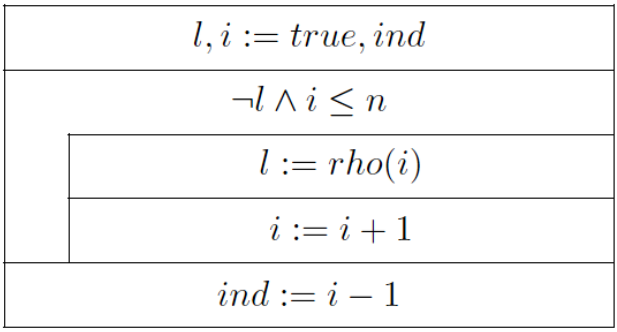


Figure 6: The method correct()

This examination is a special (optimistic) linear search [3, 4] which must check $\rho(i)$ where i goes from 0 up to n and must give the first i for which $\rho(i)$ is false. This process can be accelerated if the index ind ($ind \in [1, \ldots, n]$) is known where $\rho_{ind-1}(\phi(u.v))$ is true. In this case, it is enough to start the search from the index ind instead of 1 (see Figure 6). If $\rho(\phi(u.v))$ is false, it is useful to give back the ind where $rho(ind-1)$ is true but $rho(ind)$ is false.

4.1 Depth-first enumerator

The class DepthFirstEnum describes the object of dept-first enumerator (see Figure 7). It provides the enumeration operators: first(), next(), current(), end() [3, 4]. These operators iterate over the partial branches of the backtracking tree of the problem space of the backtracking task.

Each partial branch can be represented with its ending node, which is an element of $D_1 \times \ldots \times D_{ind-1}$. It can be described with the members of the variable u of Task and the variable ind of (N). The values of ind are between 0 and n. Thus these are members of the class DepthFirstEnum. We suppose that $\rho_{ind-1}(\phi(u.v))$ and for all $i \in [ind+1..n] : u.v[i] = 0$. The method end() indicates the end of the traversal. This is implemented by $ind = 0$ when the traversal has stepped back from the root because it could not find a solution. The method current() returns the current node that is represented by the members. The method first() sets the initial values of the members. In the case $n < 1$ the traversal must be finished immediately, i.e. $ind := 0$. Otherwise

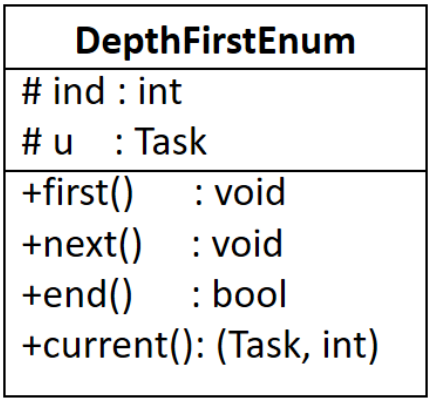


Figure 7: The class of depth-first enumerator

the initialization $u.v := [0, \ldots, 0] : ind := 1$ is needed. The method $next()$ does one step in the problem space according to the depth-first traversal (see Figure 2 with the following corresponding: $u.rho(ind)$ instead of $\rho_{ind}(\phi(v))$, $u.v[ind]$ instead of v_{ind}, and $u.m[ind]$ instead of m_{ind}).

4.2 Increasing enumerator

The class $IncreasingEnum$ describes the object of increasing enumerator (Figure 8). It provides the enumeration operators: $first()$, $next()$, $current()$, $end()$. These operators iterate over some leaves of the backtracking tree. These leaves must be enumerated in increasing order according to their value in the positional numeral system in mixed bases, which has been mentioned before.

Each leaf can be represented by the variable u of $Task$. It is worth introducing the member c of $\{0, 1\}$ that is the overflow digit of the increasing process (see Figure 8). Its value 1 indicates the end of the traversal.

The method $end()$ checks the value of overflow digit c. The method $current()$ returns the current leaf. The method $first()$ initializes $u.v$ and c. The enumeration starts with the element described by the number $[0, \ldots, 0]$ (this is the first value of the variable u) and the overflow c is 0 except for the case $n < 1$ when the traversal must be finished immediately, i.e. $c := 1$. The method $next()$ does one step in the problem space according to the increasing traversal (see Figure 2). Its input parameter is the position ind ($ind \in [1, \ldots, n]$), which shows which position of the number u must be increased according to

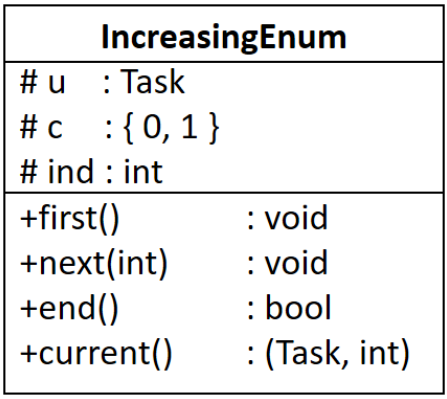


Figure 8: The class of increasing enumerator

the rules of the positional numeral system in mixed bases.

The input of the method next() is provided by the method correct() (see the class Task) and that method requires the index ind produced by the method next() where $\rho_{ind-1}(\phi(u.v))$ is true but $\rho_{ind}(\phi(u.v))$ is false. Thus the members of the class of the increasing enumerator might be completed with the index ind so that either ind is zero or $\rho_{ind-1}(\phi(u.v))$ holds. In addition each element of u.v behind the position ind is zero. Initially the method first() gives the index ind a value (ind := 0), this index is changed based on its input parameter and then its value is recomputed by the method next(), and its value can be queried by the method current().

5 Component-oriented backtracking

Based on a backtracking enumerator, the backtracking algorithm can be composed easily. In object-oriented environment, the backtracking algorithm is the result of the cooperation of three objects (see Figure 9): the object of the backtracking enumerator, the object of the task, and the object of the linear search over enumerator [3, 4].

The classes of two kinds of enumerators have been derived from the abstract class Enumerator (see Figure 11). This super class includes an object of the class Task and an index. The role of this index has been discussed earlier. This index is needed for the method cond() of the linear searching.

Under increasing enumeration the method next() uses this index: its initial

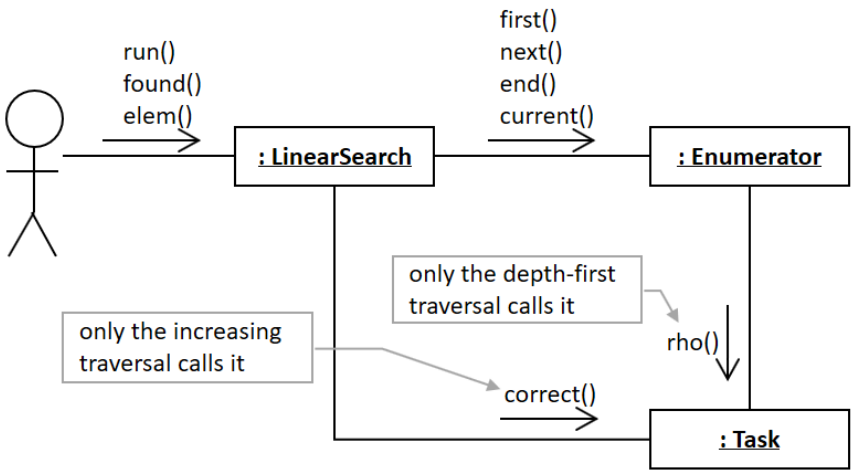


Figure 9: Collaboration of components of the backtracking algorithm

value is got from the linear searching and then the method next() modifies it. However, this would be an irregular implementation of the method next() because it usually has no external input. Thus – instead of changing the interface of the method next() – a "setter" method of the index has been introduced (setInd()). We do the same with the extra output of the method current(). Since this output is required by the linear search a "getter" method has been also implemented with this very index (getInd()). These new methods are defined in the super class Enumerator.

The super class LinearSearch (see in Figure 11) provides the method run() that encapsulates the schema of the algorithm of linear searching, furthermore the method found() and the method elem() that give the result of the search. Two versions of this algorithm can be differed depending on the way of the traversal. (see Figure 10) Under depth-first traversal the solution can be found if the index of the enumerator is greater than n. This makes calling the method correct() unnecessary. The index ind can be asked from the class DepthFirstEnum with its "getter" and the value n can be reached through the object u. Under the increasing traversal the method correct() requires the index of the enumerator (this can be asked with the "getter" of the class IncreasingEnum), modifies this index, and gives it to the method next() through the "setter" of the enumerator. These differences can be written in

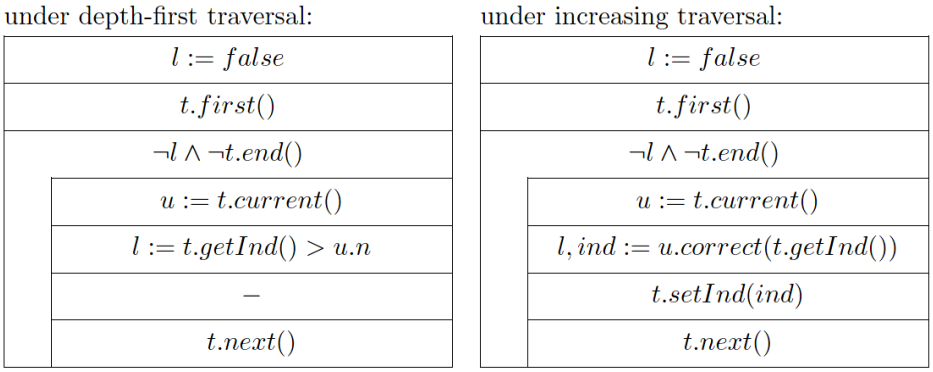


Figure 10: Two versions of linear search

the overridden method `cond()` of the class `DepthFirstLinSearch` and the class `IncreasingLinSearch` thus the method `run()` which calls this method `cond()` can be implemented independently on the way of enumeration. Here the "template function" design pattern is applied. (We must remark that the assignment `u := t.current()` in the method `run()` requires a deep copy.)

The backtracking algorithm is an instance of the class `BacktrackSearch` (see Figure 11). It owns the enumerator which includes the task and creates the appropriate object of the linear searching depending on the kind of the enumerator. The method `run()` calls the same named method of linear searching. Here the "bridge" design pattern is applied.

6 Discussion

The fact that the backtracking consists of three, well-separated components makes the algorithm very flexible. By changing components, it is very easy to change the properties of the search.

In order to solve a new task, it is enough to derive a new class from the super class `Task` and only the abstract method `rho(i)` must be overridden. The object `u` of the enumerator will be an instance of this new class while the other two objects (the enumerator and the linear search) do not change; they are reused.

The object of linear search must be exchanged when the task does not look for one solution but it wants to count how many solutions there are or it wants to look for the best solution according to a given point of view. In these cases,

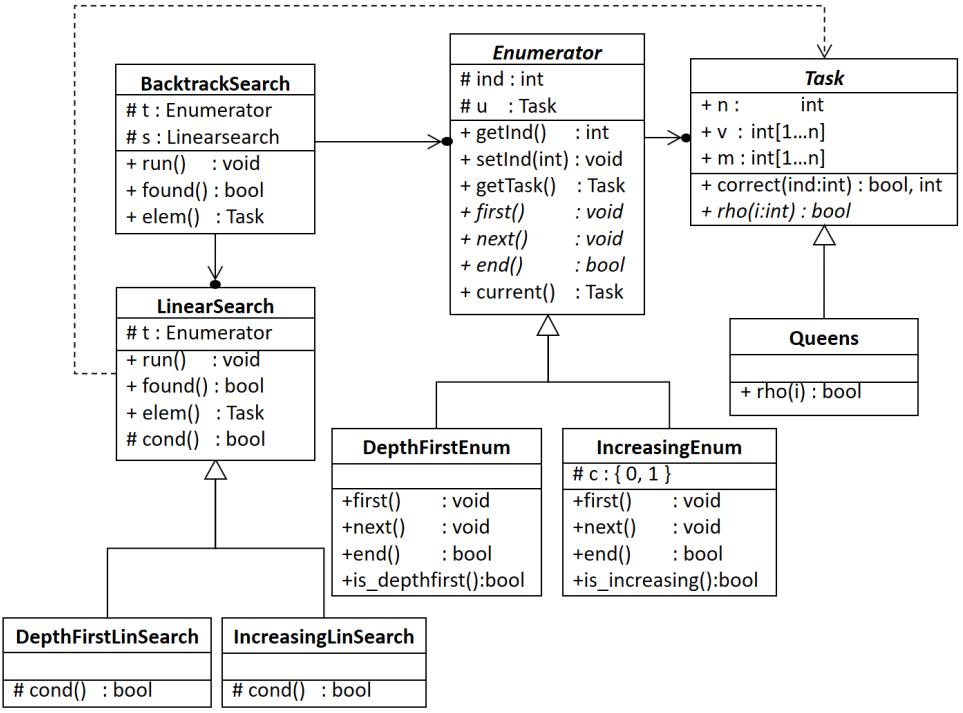


Figure 11: Class diagram of the backtracking algorithm

it is enough to use a counting or a conditional maximum search instead of the linear search.

The "backtracking counting" comes from the counting [3, 4] over enumerators if it uses a backtracking enumerator. Sometimes, only certain solutions must be counted. To solve this task, a logical function $\beta_i : D \to \mathbb{L}$ is needed to check this certain property. (see Figure 12)

The "backtracking maximum search" is the conditional maximum search [3, 4] with a special enumerator. (see Figure 13) The function $f : D \to H$ maps to the well-ordered set H, thus the states of D can be compared.

Only the class Task has to be modified if the model of the task which is wanted to solve slightly differs from the model of backtracking tasks. Many times, the problem space of a path-finding task can be described with the directed tree where the number of the outgoing arcs of the nodes can differ on the same level and the goal nodes may be inner nodes.

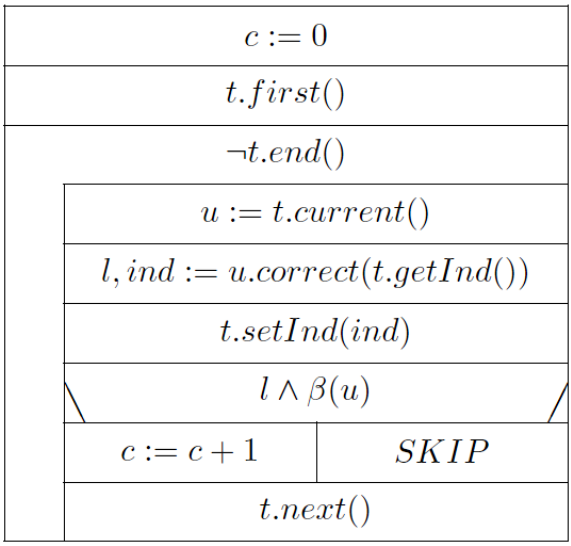


Figure 12: Backtracing counting with increasing enumerator

In the first case, the tree can be extended with alibi nodes so that the branching factor becomes constant inside one level of the tree. Certainly, the semantic of the statement ρ_i must be changed so that it gives false on these alibi (fake) children node.

In the second case, the criterion $\rho \equiv \rho_n$ is substituted with the criterion $\rho \equiv \exists i \in 1, \ldots, n : \rho_i$ or, in a more general way, the criterion $\rho \equiv \exists i \in 1, ldots, n : \rho_i \wedge \gamma_i$ where $\gamma_i : D \to \mathbb{L}$ is the logical function so that the value of $\gamma_i(u)$ depends only on the first i components of the state u. During depth-first traversal, the method correct() of the linear search must be overridden as $\gamma_{ind}(u)$. The value of $\gamma_{ind}(u)$ can be computed by an appropriate new method gamma() of the class Task. During increasing traversal, the method correct() must be overridden so that it results in true if there exists an $ind \in 1, \ldots, n$ where $\rho_{ind}(u) \wedge \gamma_{ind}(u)$ holds, otherwise it results in the first index $ind \in 1, \ldots, n$ where $\rho_{ind}(u)$ false.

At the end, we mention the didactical advantage which appears when this component-oriented backtracking is introduced into education. At this stage of the syllabus, the linear search (and other algorithm patterns) has been well known. Only the backtracking enumerator is the novelty. It can help if students have already met the concept of the enumerator; moreover, they have

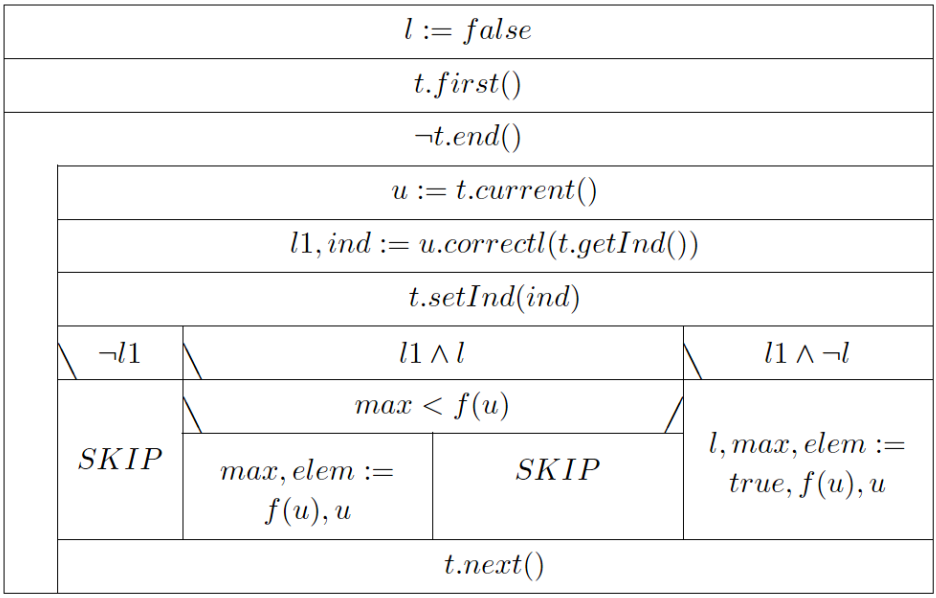


Figure 13: Backtracking maximum search with increasing enumerator

used various enumerators. Certainly, the description of the backtracking tasks which can be solved in this way must be introduced but it is not avoidable in other syllabi.

References

[1] K.A. Berman, J. L. Paul, *Fundamentals of Sequential Algorithms*, PWS Publishing, 1996. ⇒146

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition), The MIT Press, 2009. ⇒148

[3] T. Gregorics, *Programozás 1.kötet Tervezés*, ELTE Eötvös Kiadó, 2013. (in Hungarian) ⇒145, 153, 155, 158

[4] T. Gregorics, Programming theorems on enumerator, *Teaching Mathematics and Computer Science*, **8,** 1 (2010) 89–108. ⇒145, 153, 155, 158

[5] I. Fekete, T. Gregorics, S. Nagy, *Bevezetés a mesterséges intelligenciába*, LSI, 1990. (in Hungarian) ⇒145

[6] Á. Fóthi, *Bevezetés a programozásba*, ELTE Eötvös Kiadó, 2005. (in Hungarian) ⇒146, 151

[7] I. Futó (ed), *Mesterséges intelligencia*, Aula, 1999. (in Hungarian) ⇒145
[8] D. E. Knuth, Estimating the efficiency of backtrack programs, *Mathematics of Computation* **29** (1975) 121–136. ⇒146
[9] K. I. Lörentey, Fekete, Á. Fóthi, T. Gregorics, On the wide variety of backtracking algorithms, *ICAI'05 Eger, Hungary, January 28–February 3.* 2005, pp. 165–174. ⇒145, 146, 151
[10] N. J. Nilsson, *Principles of Artificial Intelligence*, Springer-Verlag, Berlin, 1982. ⇒145, 148

Received: November 2, 2017 • Revised: November 27, 2017

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 162–194

DOI: 10.1515/ausi-2017-0011

Necessity and complexity of order picking routing optimisation based on pallet loading features

Tamás BÓDIS
Department of Logistics and Forwarding,
Széchenyi István University,
Egyetem tér 1. Győr, 9026, Hungary
email: bodis.tamas@sze.hu

János BOTZHEIM
Department of Automation,
Széchenyi István University,
Egyetem tér 1. Győr, 9026, Hungary
email: dr.janos.botzheim@ieee.org

Péter FÖLDESI
Department of Logistics and Forwarding,
Széchenyi István University,
Egyetem tér 1. Győr, 9026, Hungary
email: foldesi@sze.hu

Abstract. Order picking is the most labour-intensive and costly activity of warehouses. The main challenges of its improvement are the synchronisation of warehouse layout, storage assignment policy, routing, zoning, and batching. Furthermore, the competitiveness of the warehouse depends on how it adapts to the unique customer demands and product parameters and the changes. The operators usually have to manage the picking sequence based on best practices taking into consideration the product stacking factors and minimising the lead time. It is usually necessary to support the operators by making effective decisions. Researchers of the pallet loading problem, bin packing problem, and order picking optimisation provide a wide horizon of solutions but their results are rarely synchronised.

Computing Classification System 1998: F.2.2
Mathematics Subject Classification 2010: 90B05
Key words and phrases: order picking, pallet loading problem, bin packing, complexity, routing, simulated annealing

The research defines the order picking routing problem based on Pallet Loading Feature (PLF). It describes measurement and product stacking rule evaluation methods to highlight when the PLF based optimisation is necessary. The paper shows that in order picking problems based on PLF, the number of combinations a brute-force search algorithm has to examine grows exponentially, which highlights the importance of meta-heuristic optimisation. The study describes a Simulated Annealing algorithm for order picking based on PLF.

1 Introduction

Satisfying the customers from a warehouse with the right products at the right place and time with low cost requires a synchronised and optimised warehousing system. The general warehousing goals are to handle and store items in the storage system and prepare the ordered Unit Loads (UL) for transport.

Order picking is the most labour and capital intensive operation when the operators collect the ordered items and build transport units. As Gamberini et al. highlighted, its cost can reach 65% of the total warehouse management expense [14]. The order picking system (OPS) design depends on several elements: products, customer orders, different types of functional areas, different combination of equipment types, and operating policies for each functional area [2, 12, 14].

The layout design, storage location assignment methods, routing methods, order batching, and zoning are the main decision fields during order picking processes (OPP) development [12]. The main influencing factors of order picking time are moving, searching, picking, and preparation. While travelling time gives 50% of the whole picking time, the primary optimisation task is the routing. Its aim is to sequence the items on the order picking list in order to get the shortest order picking route length [12].

"The Storage Location Assignment (SLA) optimisation is responsible for allocating products to storage locations for the purpose of lowering routing distance, travelling time, material handling cost and improving space utilisation." [10]. It enables us to take into consideration the ordering frequency of items and product parameters.

During order picking, the operators collect and allocate products to Stock Keeping Units (SKU) where positioning is a general problem. SKU can be for example a pallet, box or bin, which is responsible for forming a material handling unit, protecting the products and supporting material handling. The

Container Loading Problem algorithms are responsible for assigning three-dimensional small items to three-dimensional rectangular large objects (i.e., truck, containers, pallet). Its aim is to hold the basic geometric feasibility conditions and reach the defined problem specific objective function [6].

The basic geometric feasibility conditions are: [6]

- the small items are positioned within the container,
- the small items do not overlap.

Bortfeldt et al. collected and structured the main objective functions and problem types of the Container Loading Problem, where Bin Packing is a minimisation problem. Generally it is responsible for assigning strongly heterogeneous items into a minimum number of containers. In the case of warehouses, Bin Packing algorithms are used, for example, when the customer order has to be separated to ULs, because of, for example, a large quantity order or a high number of items [6].

The Pallet Loading Problem (PLP) is a maximisation problem, which is responsible for packing the maximum number of identical rectangular boxes onto a rectangular pallet [1]. In the case of warehouses the PLP answers the question of how to position the items on the pallet. These items can be defined to this pallet by a Bin Packing algorithm.

The various SKU properties of the ordered products and the specified pallet loading requirements of different partners make the Bin Packing and Pallet Loading Problem complex.

While each influencing parameter of order picking has an impact on the others with a different importance, these factors should be synchronised. It highlights the complexity of the warehousing system development. For example, Webster et al. examined the impact of SLA on warehouse throughput in the case of bucked brigade order picking [24]. Many researchers work on the different segments of the OPP, Bin Packing or PLP development. A huge amount of research has been done in the field of routing optimisation in warehouses (e.g., [23, 9]). Many solutions have been defined for harmonising SLA and routing to decrease the routing distances and times (e.g., [19, 8]). However, while the physical product parameters (dimensions, weight, SKU type) and product stacking properties influence the physically possible picking sequence in order to build stable ULs, researchers rarely take into account these aspects during SLA and routing optimisation. Furthermore, many researchers have attained valuable results in the fields of Pallet Loading and Bin Packing Problem (e.g., [18, 17, 3, 21, 11]), but the solutions are rarely harmonised with SLA and routing algorithms. Shiau et al. solved the multi-container loading problem and

defined the order picking sequence but they avoided the SLA [22]. Molnar et al. highlighted the importance of a well sequenced order picking list to support well structured and stable ULs to avoid product damages [20]. While Molnar et al. developed routing optimisation by considering product stacking properties, they determined picking sequence of product classes. Their algorithm minimises the difference from the defined sequence and minimises the distance but sometimes a more flexible and more complex sequencing rule definition could be required, which depends not only on the product parameters [20].

While the product stacking property based order picking is not relevant at every warehouse, the first goal of the proposed research is to define a methodology for determining its relevance for a given warehouse. The second research goal is to support the order picking operators in order for them to make more objective decisions to decrease the order picking lead time and to build stable transport units that avoid product damages, when stacking property is an important factor during order picking.

We already partly discussed the research problem, methods and results in papers [4] and [5]. The aim of [5] was to describe the pallet loading rule modelling, product classification, and decision matrix defining methodology. It examined the complexity of a simple order picking routing case with pallet loading, where the products are stored in separated warehouse zones. The examination highlighted, when the products are stored in separated and sequenced pallet loading parameter based zones, the order picking operators visit the zones in logical sequence and collect the items using general routing algorithms within the zone, then non-evolutionary algorithms can handle the problem and support the operator with right picking sequence. While it was a simplified case of the mentioned research, this current paper examines a more detailed methodology and more complex cases.

The aim of this paper is to define and describe the Order Picking Routing Problem based on Pallet Loading Feature (OPRP-PLF). It describes, clarifies, and applies methodologies, measurement- and evaluation techniques for highlighting the relevance of OPRP-PLF at the examined warehouse to support tactical decisions before algorithm development. The paper explains and applies classification and PLF based order picking decision matrix modelling solutions. The previous versions of these solutions were developed by the authors [5]. Besides the detailed problem description and clarified explanation of methodologies, the novelty of this paper is to apply methodologies for more complex and industrially more relevant cases. While examining the complexity of the problem is necessary before optimisation in order to find the right methodology, the paper proves the complexity of PLF based order picking

routing optimisation in the case of one and more UL required orders. The complexity evaluation highlights the importance of meta-heuristic optimisation. As a novelty, the paper examines analytic examples for simple order picking tasks and introduces a Simulated Annealing (SA) algorithm for solving complex PLF based order picking sequencing tasks. The aim of the SA algorithm is to introduce one possible algorithm for the problem to support the operators with quick pallet loading feature related order picking routing decisions.

2 Order Picking Routing Problem based on Pallet Loading Feature (OPRP-PLF)

The proposed research defines the Pallet Loading Feature (PLF) and the Order Picking Routing Problem based on Pallet Loading Feature (OPRP-PLF). PLF is defined as logistics system property, which requires the right picking sequence and pallet loading method to build stable ULs and to avoid product damages. The challenges of OPRP-PLF are to minimise the order picking lead time, build stable transport units and avoid product damages, when industrially relevant but rarely discussed PLF based order picking sequencing is necessary.

The PLF and the OPRP-PLF depends on product properties, order picking list characteristics, and the order picking system, which has several factors:

- Product properties
  - Weight,
  - Shape,
  - Size,
  - Stock Keeping Unit,
  - Stacking properties.
- Order picking list characteristics
  - Ordered items,
  - Ordered quantity,
  - Length of picking list,
  - Number of product types on the list, with different stacking properties,
  - Special customer rules for pallet loading.

- Order picking system
  - Previously picked units and the sequence,
  - Product assignment in the warehouse.

Each product has several parameters, which define their physical stacking property and the required picking sequence (i.e., weight, shape, size). SKUs are not only boxes but bags, cans or any amorphous units, which also have huge impact on stacking property.

The characteristics of orders also influence the OPRP-PLF and the right picking sequence. Its evaluation is essential during warehouse development. The ordered items' property and the ordered quantity influence the stacking properties. For example a high quantity from a small and weakly packaged product can behave together like one simple box and after its picking, the picker might be able to pick further boxes. Some different items with different types of SKU or packaging also can behave stronger together, rather than separately. The length of picking list highlights the necessity of the routing optimisation. A short list usually does not require complex and maybe time consuming optimisation. However, it is necessary to optimise the longer and more complex picking lists, which will save time for the warehouse operation. The number of different product stacking types also influences the requirements of PLF based optimisation. More and more different product types on the same picking list increase the complexity of the order picking sequence, which requires applying optimisation algorithms for OPRP-PLF. The customers usually define the expected pallet loading rules, which usually limit the possible picking sequence. For example the customer sometimes expects "sandwich ULs", which needs to pick a pallet after every picked record to separate items in the same UL. This UL type has an impact on the OPRP-PLF and sometimes changes the stacking properties of items.

The order picking processes themselves have an impact on the OPRP-PLF. During order picking, the previously picked items and its quantity usually influence the possible further picked items and sequence. The picking positions of the products (SLA) have a high impact on picking distances, which influence the necessity of the routing optimisation to reach the shortest lead time and follow the stacking rules.

2.1 OPRP-PLF related decisions

The OPRP-PLF based development is connected to strategical, tactical, and operative decisions. Related to our research the following main challenges and

questions arise regarding the different decision levels.

On a strategic level the warehouse management has to determine the long term business strategy, the main services offered, the main industry (Fast-Moving Consumer Goods (FMCG), automotive, pharmaceutical ...), the infrastructure requirements, and development goals.

On a tactical level several decisions should be made regarding policies and algorithms. It is important to define an ordering policy, as it is allowed for the customers to purchase order (minimum quantity, SKU, ordering time window, etc.). The handled product types and those possible SKUs should also be determined. The product properties will be one input to define the relevance of OPRP-PLF, which must be examined on tactical level as an initial step of algorithm development. Warehousing algorithms (storing in, storing out, replenishment, routing) should be developed, which hopefully will support the operational decisions. If the OPRP-PLF is relevant, the warehousing algorithms should take into consideration the OPRP-PLF with the right weighting. The SLA should also be determined on a tactical level and continuous re-engineering is necessary based on seasonal or periods of changing demand.

On the operational level the warehouse management has to make several decisions hopefully supported by algorithms. When the orders arrive at the warehouse and the order picking tasks are defined, it is necessary to determine: does the ordered quantity fit into one UL or how many UL will be necessary? It is a complex and important question of how the ordered items will be separated to ULs. The optimised order picking routing of the rightly defined UL picking lists should result in the shortest order satisfaction lead time and result in stable ULs. The routing optimisation is strongly connected with the SLA and with the PLF. Due to complex OPRP-PLF, in the case of a well designed SLA and routing algorithm, the shortest picking distance might not result in the shortest lead time, because the picker might have to spend time on UL reconstruction during order picking. The necessity of reconstruction can be caused by higher or lower ordered quantity, as it is assumed during SLA, because different amounts of product can behave differently on the UL. In this case a longer distance might result in shorter lead time because of less pallet loading time. On an operational level the routing algorithm should decide, how to reach the shortest lead time. The possible solutions are to collect items in the right PLF based sequence and walk more or pick with shorter routing and spend time on redesigning the contents of the UL when it is necessary. The best choice depends on the SLA, the time requirement of movements, and the length and contents of the picking list. The increasing frequency and time

requirement of pallet loading and reconstruction can highlight the decreasing efficiency of SLA and the necessity of its re-engineering. The well-defined Key Performance Indicators (KPIs) can highlight the necessity of tactical decisions or re-engineering.

3 Defining the necessity of optimisation for OPRP-PLF

PLFs are unique characteristics of each warehouse. It is an important factor mainly at distribution warehouses where order picking has a high importance, and the handled products have a huge number of variants. The unique nature of warehousing systems requires that methodology be defined for determining warehouse by warehouse the relevance of OPRP-PLF and the importance of applying optimisation algorithms for it. The proposed research defines the relevance of OPRP-PLF with time measurements of warehousing processes and with the evaluation of the modelled pallet loading sequencing possibilities.

3.1 Defining the relevance of OPRP-PLF with measurement

Measuring the warehousing processes is essential to understand the real nature of the developed warehouse and collect information about the most time consuming movements, relation of causes and effects. The warehousing processes are separable for elementary movements (i.e., travel, administration, pick, search, setup), which can highlight the relevance of PLF at a given warehouse. It is necessary to examine the processes step by step to overlook the sequence, the frequency, the time distribution, and the casual relations of elementary movements. The PLF dependent movements and those that are relevant are different warehouse to warehouse. Some typical steps are the UL reconstruction, travel time, and wrapping [4].

3.1.1 UL Reconstruction

The picker spends time on UL reconstruction when rebuilding the UL structure during order picking. Frequent UL reconstruction movement highlights the importance of PLF and the necessity of SLA re-engineering [4].

3.1.2 Travelling speed

When the PLF is relevant at the measured warehouse and re-engineering of the OPP is necessary, the order picking operators usually move longer distances and have longer routes for similar picking lists and lower the travelling speed during picking to avoid the products from falling down [4].

3.1.3 Wrapping

Wrapping is sometimes necessary to strengthen the picked ULs during the long and complex picking tasks. The wrapped ULs are much more stable, which results in less product damage and higher travel speed during order picking. The frequency and the time requirements of wrapping during OPP are measurable. The necessity of wrapping during picking can highlight the relevance of PLF [4].

3.2 Modelling the pallet loading possibilities

The OPRP-PLF depends on product properties, order picking list characteristics, and order picking systems. The possible PLF factors' combination and importance are different warehouse to warehouse, which makes the PLF examination and implementation into the order picking algorithms complex. In the previous part of the research [5] a methodology has been defined to model the possible sequencing rules. The resulted PLF based Decision Matrix (PLFDM) allows us to examine the nature and the complexity of proposed problem. It defines the pallet loading possibilities and the order picking sequencing rules. It will be the basis of the loading algorithm during order picking routing optimisation based on PLF. One, two, and three dimensional loading algorithm can control the possible picking sequence based on the PLFDM. In the case of the 1D problem, a full layer of products is picked into one column and only the vertical sequencing is important. The 2D problem handles neighbouring pallet wide columns on the pallet. The 3D problem is the most complex, because it allows to build any columns on the pallet. This paper will apply the PLFDM to the 1D order picking problem. [5]

4 Methodology to define the PLFDM

The warehouses usually do not have enough and appropriate data regarding the handled items (i.e., dimensions, shape, weight) or those that are changing too often to support a complex PLP or Bin Packing algorithm. However, based

on known, easily measurable, and rarely changing information it is possible to classify the items and every order line. Furthermore, the warehouse operator's best practices and experiences are essential and valuable inputs to the classification. The defined Pallet Loading Classes (PLC) have PLF based logical connections (i.e., we should not put heavy goods on fragile products), which will be the basis of the PLF based OPP algorithms [5].

This section summarises and clarifies the mentioned classification process step by step based on previous research of the authors [5].

4.1 Classification of the product parameter based classes

The products are grouped based on physical product parameters, which has different stacking properties (Eq. (1)). The usually considered factors of Product Classes (PC) are the SKUs, the packaging solution, and the item property. Equation (2) shows a possible industrial example [4, 5].

$$\mathrm{PC} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\} \tag{1}$$

$$\mathrm{PC} = \{\mathit{PlasticBin}, \mathit{CartonBox}, \mathit{SmallBox}, \mathit{Fragile}\} \tag{2}$$

4.2 Classification of the product and order parameter based classes

The defined PCs are specified based on order parameters to define the Product and Order Parameter based Classes (POPC). If it is necessary we separate PCs into further classes. The high quantity of the same item usually has different stacking parameters. For example element B (CartonBox) of the PC set is separated into High Quantity (HQ) and Low Quantity (LQ) elements (Eqs. (3) and (4)). In this case, if the ordered Quantity (Q) is equal to or higher than 4 then the order line is classified as $\mathrm{B_{HQ}}$. Four carton boxes – which are stored in a full layer on the pallet – can be more stable on a UL than only one carton box. If the ordered Quantity (Q) is smaller than 4, then the order line (r) is classified as $\mathrm{B_{LQ}}$. Equations (5) and (6) show an example of the CartonBox class. [4, 5]

$$\mathrm{B_{HQ}} \in \mathrm{POPC} \mid (\mathrm{r_{PC}} = \mathrm{B}) \wedge (\mathrm{Q} \geqslant 4) \tag{3}$$

$$\mathrm{B_{LQ}} \in \mathrm{POPC} \mid (\mathrm{r_{PC}} = \mathrm{B}) \wedge (\mathrm{Q} < 4) \tag{4}$$

$$\mathit{CartonBox_{HQ}} \in \mathrm{POPC} \mid (\mathrm{r_{PC}} = \mathit{CartonBox}) \wedge (\mathrm{Q} \geqslant 4) \tag{5}$$

$$\mathit{CartonBox_{LQ}} \in \mathrm{POPC} \mid (\mathrm{r_{PC}} = \mathit{CartonBox}) \wedge (\mathrm{Q} < 4) \tag{6}$$

4.3 Classification of the special product and order parameter based classes

The Special POPCs (SPOPC) are defined with consideration of previously picked units and their sequence (Eq. (7)). The picked items on the pallet form a physical structure, which influences the choosing of subsequent items. For example, it is possible to pick one layer of small boxes, after one layer of small boxes but the third layer of small boxes would destabilize the UL, so in this case it is forbidden to pick one layer of small boxes after two layers of small boxes (Eq. (8)). [4, 5]

$$S \in SPOPC \mid (POPC_{t-1} = Y) \wedge (POPC_{t-2} = Y) \tag{7}$$

$$\begin{aligned} SmallBoxSmallBox \in SPOPC \mid \\ (POPC_{t-1} = SmallBox) \wedge \\ (POPC_{t-2} = SmallBox) \end{aligned} \tag{8}$$

Where t is the actual picking step, $t-1$ is the previously picked POPC, and $t-2$ is the last but two picked POPC.

4.4 Defining the PLFDM

PLF based Decision Matrix (PLFDM) models the PLF based sequencing logic. The predecessors (rows) are the elements of the Pallet Loading Class (PLC) set, which are the union of POPC and SPOPC sets. The successors (columns) are the elements of the POPC set (Eqs. (9) and (10)). [4, 5]

$$PLC = POPC \cup SPOPC \tag{9}$$

$$PLFDM : PLC \times POPC \mapsto \{0, 1\} \tag{10}$$

$$PLFDM\,(PLC_i, POPC_j) = \begin{cases} 1, & \text{if } POPC_j \textbf{ can} \text{ be picked after } PLC_i \\ 0, & \text{if } POPC_j \textbf{ can't} \text{ be picked after } PLC_i \end{cases} \tag{11}$$

The PLFDM values are 0 or 1, depending on pallet loading possibilities. If it is possible to pick the examined item (one element of POPC) after the already picked units (PLC element), then the PLFDM value is 1 (true). Otherwise picking is forbidden, so the PLFDM value is 0 (false) (Eq. (11)). Table (1) and Table (2) illustrate an example PLFDM. [4, 5]

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	1	1	1	1	1
PLC_2	0	1	1	1	1
PLC_3	0	0	1	1	1
PLC_4	0	0	0	1	1
PLC_5	0	0	0	0	1
PLC_6	0	0	0	0	1

Table 1: Pallet Loading Feature based Decision Matrix (PLFDM).

	PlasticBin	$CartonBox_{HQ}$	$CartonBox_{LQ}$	SmallBox	Fragile
PlasticBin	1	1	1	1	1
$CartonBox_{HQ}$	0	1	1	1	1
$CartonBox_{LQ}$	0	0	1	1	1
SmallBox	0	0	0	1	1
SmallBoxSmallBox	0	0	0	0	1
Fragile	0	0	0	0	1

Table 2: Pallet Loading Feature based Decision Matrix (PLFDM) example.

4.5 Merging the compatible records in the PLFDM

The resulted PLFDM usually contains records, which have exactly the same values. These PLCs have different properties but they behave in the same way during order picking. This is the reason why those records can be merged, which can simplify the PLFDM. For example in Table (3), where PLC_5 and PLC_6 are merged this results in $PLC_{5,6}$. [5]

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	1	1	1	1	1
PLC_2	0	1	1	1	1
PLC_3	0	0	1	1	1
PLC_4	0	0	0	1	1
$PLC_{5,6}$	0	0	0	0	1

Table 3: Merged pallet loading feature based decision matrix (PLFDM)

5 Defining the importance rate of PLF based order picking process development

After defining the PLFDM of a warehouse, it is possible to evaluate the matrix and define the relevance of PLF based order picking routing optimisation. The Pallet Loading Rate (PLR) is defined based on the amount of pallet loading restrictions; otherwise it is based on the amount of the edges in the PLFDM. Basically, when every POPC can be picked after every PLC, the PLFDM contains 1 in every cell, then the PLF is not relevant at the given warehouse and the Pallet Loading Rate (PLR) is 0. When more and more restrictions (0 value) are defined in our PLFDM then the complexity of OPP and the importance of PLF based routing optimisation is growing. The PLR is calculated by Equation (12), where MaxNum_e is equal to the number of true values in the PLFDM when every POPC element can be picked after every PLC element. Num_e equals to the number of true values in the PLFDM when PLF is modelled [4].

$$\mathsf{PLR} = 1 - \frac{\mathsf{Num}_e}{\mathsf{MaxNum}_e} \tag{12}$$

As part of the research the PLR values have been classified based on intervals, which describe the importance of PLF based routing optimisation at the examined warehouse [4].

- PLF is not relevant, when $\mathsf{PLR} = 0$,
- PLF is weakly relevant, when $0 < \mathsf{PLR} \leqslant 0.2$,
- PLF is relevant, when $0.2 < \mathsf{PLR} \leqslant 0.4$,
- PLF is strongly relevant, when $0.4 < \mathsf{PLR}$.

Tables (4)–(7) show examples for each PLR category.

6 Complexity of PLF based routing optimisation

This paper examines the OPRP-PLF to define the complexity of PLF based order picking routing optimisation. Two main cases have been defined for OPRP-PLF, which might have further sub-problems:

- optimising an order with items, which can be picked into 1 UL and order separation is not necessary,
- optimising an order, which will be picked into more than 1 UL and order separation is necessary.

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	1	1	1	1	1
PLC_2	1	1	1	1	1
PLC_3	1	1	1	1	1
PLC_4	1	1	1	1	1
PLC_5	1	1	1	1	1

Table 4: PLFDM example, when PLF is not relevant

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	1	1	1	1	1
PLC_2	1	1	1	1	1
PLC_3	1	1	1	1	1
PLC_4	0	0	1	1	1
PLC_5	0	0	0	1	1

Table 5: PLFDM example, when PLF is weakly relevant

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	1	1	1	1	1
PLC_2	1	1	1	1	1
PLC_3	0	0	1	1	1
PLC_4	0	0	0	1	1
PLC_5	0	0	0	0	1

Table 6: PLFDM example, when PLF is relevant

	$POPC_1$	$POPC_2$	$POPC_3$	$POPC_4$	$POPC_5$
PLC_1	0	0	1	1	1
PLC_2	0	0	1	1	1
PLC_3	0	0	0	1	1
PLC_4	0	0	0	0	1
PLC_5	0	0	0	0	0

Table 7: PLFDM example, when PLF is strongly relevant

The paper determines the formula for both cases to calculate the possible number of order picking sequencing variations and it examines the behaviour of those in the case of several order picking lists whose length and contents are different.

The aim of this research is to define the complexity and the nature of the search space, which should be handled by the order picking operator. It is not an objective of the paper to classify the investigated problem into a specific complexity class. Its goal is merely to emphasize the supra-exponential growing of the proposed problem.

6.1 Complexity of order picking of one UL without order separation

First, a simpler case is examined when order picking of one UL should be optimised and order separation is not necessary. In this case the customer purchases an order, which will be picked by an operator to one UL. The important questions are:

- How many different picking sequences of the ordered products are possible?
- Is the operator able to find the right sequence of picking herself/himself or is an algorithm necessary?
- What kind of optimisation algorithm is necessary for this kind of problem depending on its complexity?

Each order picking list can contain every PLC every time. While the rules of PLFDM are true, any sequence of the PLCs is possible. The main parameters, which influence the number of sequencing variations of a picking list, are the number of records ($\mathtt{k}$), the number of PLCs ($\mathtt{n}$) and the occurrence of the PLCs ($\mathtt{i}$) in the order picking list.

The PLC occurrence is necessary because every PLC contains several products, which usually have their own picking positions. The possible variations of the picking positions within a PLC have to be considered. $\mathtt{i}$ is defined from the order picking list point of view, to count the occurrence of PLCs, which are on the order picking list (Eq. (13)). When a PLC occurs $\mathtt{i}$ times in a picking list then its possible sequencing variations have to be counted due to the different picking positions ($\mathtt{i}!$) (Eq. (14)). The sum of occurrence values ($\mathtt{i}$) has to be equal to the number of order picking list records ($\mathtt{k}$) (Eq. (15)). The number of records ($\mathtt{k}$) has to be equal to or higher than the number of PLCs

(n) (Eq. (16)). When $k = n$ then the picking list contains 1 product from each PLC and then the variations of occurrence $(i!)$ is not necessary (Eq. (17)).

$$i > 0 \tag{13}$$

$$V = i_1! \cdot i_2! \ldots \cdot i_n! \tag{14}$$

$$k = i_1 + i_2 + \ldots i_n \tag{15}$$

$$k \geq n \tag{16}$$

$$V = i_1 \cdot i_2 \ldots \cdot i_n \tag{17}$$

For example, when the following inputs are given:

- Number of different PLCs on the list equals 3 ($n = 3$), $PLC = \{A, B, C\}$,
- Number of order picking list records equals 12 ($k = 12$),
- A PLC occurs 5 times, $i_1 = 5$,
- B PLC occurs 4 times, $i_2 = 4$,
- C PLC occurs 3 times, $i_3 = 3$.

The number of variations for the above mentioned example is $5! \cdot 4! \cdot 3! = 17280$.

This example is just one case, each n and k pairs have several combinations depending on the occurrence of PLCs. Equation (18) shows the formula, which defines the number of possible combinations.

$$C_{k;n} = \binom{k-1}{n-1} \tag{18}$$

One possible combination is when the occurrence values are balanced ($i_1 \approxeq i_2 \approxeq \ldots \approxeq i_n$) and this reaches the minimum number of variations. In the case of the mentioned example the minimum number of variations equals 13824 when:

- $n = 3$,
- $k = 12$,
- A PLC occurs 4 times,
- B PLC occurs 4 times,
- C PLC occurs 4 times.

Another possible combination is when one of the occurrence values is maximum and the others equal 1, and this reaches the maximum number of variations (Eq. (19)).

$$(k-(n-1))!\cdot(1!)^{n-1}=(k-n+1)! \tag{19}$$

In the case of the mentioned example the maximum number of variations equals 3628800, when:

- $n=3$,
- $k=12$,
- A PLC occurs 10 times,
- B PLC occurs 1 times,
- C PLC occurs 1 times.

During the complexity examination the maximum formula is going to be used (Eq. (19)) because the order picking routing optimisation algorithm has to be able to handle in this case as well, which results in the highest number of variations and causes the highest complexity.

The $(k-n+1)!$ formula (Eq. (19)) takes into consideration the PLF rules in the case of standard triangular and symmetrical PLFDM and it has some important facts:

- Many industrial cases are reducible to standard triangular and symmetrical PLFDM. However, some special industrial cases can result in neither symmetrical nor standard triangular PLFDM, which might modify the real number of variations.
- When the inverse PLFDM is examined, the number of possible variations will be the same. It highlights that the proposed formula is not applicable in PLF based order picking routing optimisation without the PLFDM.
- The proposed formula assumes that each product has only 1 picking position. However, sometimes more than 1 picking position of a product is also possible.

The aim of this formula (Eq. (19)) is to highlight the importance and complexity of PLF based order picking routing optimisation. In this case the specific cases and the inverse solutions are negligible. It could be said that when PLFs are implemented into the order picking routing optimisation, the algorithm should be able to handle a unique (non-symmetrical and non-standard triangular) PLFDM with the exact picking rules.

Generally, the necessary data (n, k, i_1, i_2,... i_n) regarding the order picking list will be available during PLF based order picking routing optimisation to see each possible variation. The algorithm will optimise the picking sequence of several order picking lists whose parameters will be different depending on the customer orders.

The $(k-n+1)!$ formula (Eq. (19)) is compared to the e^k formula to examine the complexity of the OPRP-PLF. Equation (20) can prove the exponential growth of variations, where n is an optional constant and k goes to infinity.

$$\lim_{k\to\infty} \frac{e^k}{(k-n+1)!} = \lim_{k\to\infty} \frac{e \cdot e \cdot e \cdot \ldots \cdot e}{(k-n+1)\cdot(k-n)\cdot(k-n-1)\cdot\ldots\cdot 3\cdot 2\cdot 1\cdot\ldots\cdot 1} = 0 \tag{20}$$

Equation (20) goes to 0 because each $\frac{e}{(k-n+1)}$, $\frac{e}{(k-n)}$, $\frac{e}{(k-n-1)}$ ... quotient goes to 0 and the further quotients $\left(\frac{e}{2}, \frac{e}{1}\ldots\right)$ are constants. This result means that the number of variations ($(k-n+1)!$) has a stronger growth than the e^k has. It proves that the proposed formula has at least exponential growth. It could be said that a heuristics optimisation method would be necessary for PLF based routing optimisation.

Each warehouse handles shorter and longer picking lists. Table (8) represents an example when $n = 3$ and k is between 1 and 15. In this case when $k = 12$ and $n = 3$, which is definitely possible in real life, the possible sequencing variations equal 3628800. It could be said that it is impossible for the order picking operator to be able to define a nearly optimal sequence by herself/himself without any support. Naturally, there are possible cases when k and n are smaller but in this case complex heuristic optimisation might not be relevant (i.e., in Table (8), when $n = 3$ and $k = 6$ there are 24 possible variations). It is necessary to examine the nature of picking lists on a tactical level and determine whether the order picking lists require complex PLF based optimisation or not. When there is a possibility for several complex order picking lists then implementation of PLF based heuristic routing optimisation is necessary. However, the Warehouse Management System should be able to decide on an operational level which list will be sequenced by complex and maybe time consuming algorithms and which will be handled by simple algorithms or by the order picking operator herself/himself.

k	MaxVar	e^k
1	0	2.72
2	0	7.39
3	1	20.09
4	2	54.60
5	6	148.41
6	24	403.43
7	120	1 096.63
8	720	2 980.96
9	5 040	8 103.08
10	40 320	22 026.47
11	362 880	59 874.14
12	3 628 800	162 754.79
13	39 916 800	442 413.39
14	479 001 600	1 202 604.28
15	6 227 020 800	3 269 017.37

Table 8: Number of order picking sequencing variations, when $n = 3$ and k is growing.

6.2 Complexity of order picking when order is separated to several ULs

This section examines the case when the purchased order should be separated to ULs because the ordered amount of products is higher than 1 UL's capacity. The defined ULs have the same parameters and behave in the same way as the previously discussed picking lists. The important questions are:

- How many different variations are possible for separating an order and sequencing the picking items of each UL?
- Is the operator able to find the right separation of an order and picking sequence of each UL herself/himself or is an algorithm necessary?
- What kind of optimisation algorithm is necessary for these kinds of problems depending on its complexity?

The main parameters, which influence the number of separating and sequencing variations of an order, are the number of records (Order:K, UL:k), the number of PLCs (Order:N, UL:n) and the occurrence of the PLCs (Order:I, UL:i).

It is assumed that the number of possible ULs could be equal to or lower than the number of ordered records (K) (Eq. (21)). The sum of each UL's length (k_1, k_2 ... $k_{UL_{num}}$) equals K(Eq. (22)). The order picking sequencing variations of each possible UL are counted by the previously defined formula, $V = i_1! \cdot i_2! \ldots \cdot i_n!$ (Eq. (14)). Based on the combination with repetition formula, Eq. (23) defines the possible order separation combinations, where $UL_{num} = K$, because of Eq. (21).

$$UL_{num} \leqslant K \tag{21}$$

$$K = k_1 + k_2 + \ldots + k_{UL_{num}} \tag{22}$$

$$\binom{UL_{num} + K - 1}{UL_{num} - 1} = \binom{2 \cdot K - 1}{K} \tag{23}$$

Equation (24) sums up the possible separating combinations and sequencing variations of each UL.

$$\begin{aligned} \sum_{k_1,k_2,\ldots,k_K=0}^{k_1+k_2+\ldots+k_K=K} & \binom{K}{k_1} \cdot V_{k_1} + \binom{K-k_1}{k_2} \cdot V_{k_2} + \ldots + \\ & \binom{K-k_1-k_2-\ldots-k_{K-3}-k_{K-2}}{k_{K-1}} \cdot V_{k_{K-1}} + \\ & \binom{K-k_1-k_2-\ldots-k_{K-2}-k_{K-1}}{k_K} \cdot V_{k_K} \end{aligned} \tag{24}$$

The model counts using the maximum number of variations of each UL, when $i >= 0$, thus $V_{k_j} = k_j!$. Differently from Eq. (13), $i = 0$ is allowed because in this case the order picking lists are combined during examination, thus the exact occurrence of each n is unknown. It is a specific case, which results in the highest number of variations, although during optimisation the exact occurrence of each n might be known. In this case the proposed formula can be simplified as Eq. (25) shows. Some further simplifications result in Eq. (26), which defines the possible variations for separating an order and sequencing the picking items of each UL.

$$\begin{aligned} \sum_{k_1,k_2,\ldots,k_K=0}^{k_1+k_2+\ldots+k_K=K} & \frac{K!}{(K-k_1)!} + \frac{(K-k_1)!}{(K-k_1-k_2)!} + \ldots + \\ & \frac{(K-k_1-k_2\ldots k_{K-2})!}{(K-k_1-k_2\ldots k_{K-1})!} + \frac{(K-k_1-k_2\ldots k_{K-1})!}{(K-k_1-k_2\ldots k_K)!} \end{aligned} \tag{25}$$

$$\sum_{k_1,k_2,\ldots,k_K=0}^{k_1+k_2+\ldots+k_K=K} \frac{K!}{(K-k_1)!} + \frac{(K-k_1)!}{(K-k_1-k_2)!} + \ldots + \frac{(k_{K-1}+k_K)!}{k_K!} + \frac{k_K!}{0!} \tag{26}$$

Equation (20) proves, that sequencing 1 UL is at least an exponential problem and requires heuristic optimisation. When an algorithm separates orders to ULs and sequences each UL it will be at least an exponential problem as well, which requires heuristic optimisation. It could be said that supporting the order picking operator with a separating and sequencing algorithm is even more important in this case because this problem is even more complex.

7 PLF based order picking optimisation

Depending on the length and complexity of one UL's picking list, there are 3 different levels for handling the possible picking sequence.

- When the picking lists are short (k is low, like 2-6 records) and/or the lists are simple (contain a low number of POPC, n is low) then the order picking operator is usually able to define the optimal sequence for picking, which results in the shortest picking lead time.
- When the lists are longer but still not longer than about 10 records, an enumeration based algorithm can find the best sequence using a computer. In the case of a 10 records long list the number of possible solutions are $10! = 3628800$ (when reconstruction is allowed), which can be handled by a computer without an intelligent algorithm.
- When the lists are longer than about 10 records ($k > 10$), a quick algorithm is necessary. The growing number of records results in an exponentially growing complexity, which is unmanageable within a short time window by the operators or simple heuristics. Meta-heuristic based solutions usually can be an appropriate solution to get a nearly optimal picking sequence quickly.

The following subsections introduce analytic examples for simple cases and a Simulated Annealing (SA) algorithm for more complex picking lists. The examples and the SA algorithm are defined for a 1D loading during the order picking.

The aim of this paper is to describe one possible algorithm for the complex cases to introduce the nature of the problem. While several further methodolo-

gies are possible (e.g. genetic algorithms, branch and bound methods, multi-restart hill climbers), it is not an objective of this paper to find the best methodology. We leave it for future research. Further aim is to confirm, that supporting the order picking operators with OPRP-PLF algorithms is essential to get a nearly optimal picking sequence quickly. The reasons why the SA algorithm is applied, are its potential application in an evolutionary based search, its possibility to avoid local optimum, and the good experiences about its effectiveness and simplicity.

First of all, the next subsection discusses the objective function, which evaluates the order picking sequencing solutions.

7.1 Evaluation of order picking sequence - objective function

The proposed research evaluates the possible order picking sequencing solutions based on time requirements. Counting the lead time of each picking task begins when the picking operator starts the list and picks up an empty pallet at the start-end position. The lead time measurement is finished when the picking operator has transported the ready UL to the start-end point. During order picking the picker visits each picking position on her/his list, picks the ordered items and reconstructs the UL structure, if it is necessary. The objective function (T) summarises the picking time, the UL reconstruction time, the travelling time and other time requirements (Eq. (27)). The aim of the OPRP-PLF is to minimise the order picking lead time of stable unit loads (Eq. (29)).

$$T = T_P + T_R + T_T + T_O \tag{27}$$

$$\min\left(T_P + T_R + T_T + T_O\right) \tag{28}$$

$$\min\left(T\right) \tag{29}$$

The picking time (T_P) depends on several parameters, which are usually unique for each warehouse, for example ordered quantity, weight, shape and SKU of the ordered product. In the later described test examples a constant is going to be used for the picking time (10 sec/ordered record) for simplification (Eq. (30)).

$$T_P = \sum_{i=1}^{k} t_{P_i} \tag{30}$$

The reconstruction time (T_R) depends also on several unique factors. During the order picking the picker collects the items based on the defined sequence,

when he/she reaches a record that can't be picked after the already picked items, the picker has to unload items from the UL until the actual picking record can be picked. When the actual picked items are allocated to the UL the unloaded items should be loaded back onto the UL in the right sequence. In this case it is assumed that the picker unloads the items separately in order to load everything back in the right sequence to avoid the Tower of Hanoi problem. Equation (31) shows our formula for reconstruction time, where r_p is the number of problematic records where reconstruction will be necessary, and r_r is the number of records with stronger POPC after the problematic record, and this shows the number of unpicked records. In the later described test examples a constant is going to be used for reconstruction time (t_R equals 7.5 sec/ordered record) for simplification.

$$T_R = \sum_{i=0}^{r_p} r_{r_i} \cdot 2 \cdot t_{R_i} \tag{31}$$

Equation (32) defines the travelling time based on the picker's moving speed ($v = 1,6 m/s$) and the distances between positions (S). S_{r_{i-1},r_i} is the travelling distance from the position of the previous record to the position of the actual record. S_{r_0} and S_{SE} define the start-end point where the picking starts and ends.

$$T_T = \frac{S_{r_k,SE}}{v} + \sum_{i=1}^{k} \frac{S_{r_{i-1},r_i}}{v} \tag{32}$$

Further time parameters (like for example extra administration, correction, and searching) are definable with the other time T_O parameter. In the later described test examples T_O equals 0 seconds.

The overall task is to minimise T (Eq. (29)) subject to k (number of records on the order picking list), r_p, r_r, and the distances between positions S.

7.2 Analytic solution for simple cases

When the picking lists are short (k is low, like 2-6 records) and/or the list is simple (contains low number of POPC – n is low) then the order picking operator is usually able to define the optimal sequence for picking, which results in the shortest picking lead time. This subsection describes some simple picking lists and their calculated objective functions. The lead times are calculated using the previously described formulas based on a simple distance matrix (Table (9)), where the values are defined in meters. The possible UL building rules are described in Table (10). Obviously in the explained cases the

warehouse operator defines the right sequence without any calculation, based merely on experience and best practices.

	Pos 1	Pos 2	Pos 3	Start-End
Pos 1	0	50	20	100
Pos 2	50	0	30	80
Pos 3	20	30	0	10
Start-End	100	80	10	0

Table 9: Distance matrix for the simple cases.

	A	B	C
A	1	1	1
B	0	1	1
C	0	0	1

Table 10: PLFDM for the simple cases.

The first simple case has 2 records ($\mathsf{k} = 2$) and contains 2 POPCs ($\mathfrak{n} = 2$). Table (11) shows a possible picking sequence when UL reconstruction is necessary based on the PLFDM because the record number 1 (POPC property is "B") is picked before the record number 2 (POPC property is "A"). Table (12) describes a better picking sequence when reconstruction is not necessary and the travel time is equal to the previously discussed solution.

The second simple case has 3 records ($\mathsf{k} = 3$) and contains 3 POPCs ($\mathfrak{n} = 3$). Table (13) and Table (14) evaluate 2 possible picking sequences when reconstruction is necessary and when it is not required, respectively. The results show that the second solution's lead time is lower when reconstruction is not necessary. It is highlighted that in this case the picker has to travel a longer route to avoid reconstruction and reach a lower lead time. It could be said that the picker has to take into consideration the PLF and not to minimise the route length. When the number of records is higher and/or the picking list is more complex the picking operator won't be able to make the right decision without any IT support, which defines the nearly optimal sequence.

Record ID	Position	POPC	T_P	r_{r_i}	T_R	S_{r_{i-1},r_i}	T_T	Lead Time
1	Position 2	B	00:10	0	00:00	80	00:50	
2	Position 1	A	00:10	1	00:15	50	00:31	
	Start-End					100	01:02	
Sum			00:20		00:15	230	02:23	**02:58**

Table 11: Simple case 1 ($k = 2$ and $n = 2$) **with** reconstruction.

Record ID	Position	POPC	T_P	r_{r_i}	T_R	S_{r_{i-1},r_i}	T_T	Lead Time
2	Position 1	A	00:10	0	00:00	100	01:02	
1	Position 2	B	00:10	0	00:00	50	00:31	
	Start-End					80	00:50	
Sum			00:20		00:00	230	02:23	**02:43**

Table 12: Simple case 1 ($k = 2$ and $n = 2$) **without** reconstruction.

7.3 Simulated annealing algorithm for PLF based order picking optimisation

This paper applies a simple optimisation algorithm for OPRP-PLF to find nearly optimal picking sequences in an effective way by using an enumeration based algorithm. The basis of the applied Simulated Annealing (SA) algorithm is defined by Kirkpatrick et al. for Travelling Salesman Problems (TSP) [16]. This research generalised the method for a population based Simulated Annealing algorithm. Each individual defines a possible picking sequence of the order picking list as part of a population. Each individual was developed independently based on SA methodology.

The initial population has N_{ind} individuals, which contains 1 previously sequenced individual (eugenic individual) and $N_{ind} - 1$ randomly defined permutations of order picking list records. The eugenic individual is sequenced by POPC properties to define a reliable starting solution, and its role is to verify that our SA algorithm can provide a better solution than simple sequencing.

Each individual is evolved for N_{iter} iterations without any information exchange between the individuals. The individuals do not consider the PLFDM, the PLF aspects are considered in the objective function. In iteration $Iter$ every individual is perturbed randomly. The number of perturbed records (Num_{PR}) are randomly defined between 2 and Max_{PR}, where Max_{PR} is defined by Eq. (33) with rounding, where k is the number of picking records. If Max_{PR} is lower than Min_{PR} ($Min_{PR} = 4$) then $Max_{PR} = Min_{PR}$. The randomly selected Num_{PR} records are perturbed and the other records are fixed

Record ID	Position	POPC	T_P	r_{r_i}	T_R	S_{r_{i-1},r_i}	T_T	Lead Time
3	Position 3	B	00:10	0	00:00	10	00:06	
1	Position 2	C	00:10	1	00:15	30	00:19	
2	Position 1	A	00:10	2	00:30	50	00:31	
	Start-End					100	01:02	
Sum			00:30		00:45	190	01:58	**03:13**

Table 13: Simple case 2 ($k = 3$ and $n = 3$) **with** reconstruction.

Record ID	Position	POPC	T_P	r_{r_i}	T_R	S_{r_{i-1},r_i}	T_T	Lead Time
2	Position 1	A	00:10	0	00:00	100	01:02	
3	Position 3	B	00:10	0	00:00	20	00:13	
1	Position 2	C	00:10	0	00:00	30	00:19	
	Start-End					80	00:50	
Sum			00:30		00:00	230	02:24	**02:54**

Table 14: Simple case 2 ($k = 3$ and $n = 3$) **without** reconstruction.

on the list. Equation (33) is used iteration by iteration to decrease the impact of the perturbation.

$$Max_{PR} = \left(1 - \left(\frac{Iter}{N_{iter}}\right)\right) \cdot k \tag{33}$$

The perturbed individual is kept if it gives a better solution than the unperturbed (origin) individual or if Eq. (34) holds, where $r \in [0, 1]$ is a uniform random number and τ is a positive scaling factor. The eugenic individual is overwritten only when the lead time of the perturbed individual is lower than that of the unperturbed individual. Algorithm (1) summarises the individual overwriting procedure.

$$r < e^{((-Iter \cdot \tau)/N_{iter})} \tag{34}$$

The SA algorithms are tested with one complex picking list, which is picked in a warehouse where 480 items are randomly allocated on 480 picking positions. The test picking list contains 20 records ($k = 20$) and 6 POPCs ($n = 6$).

Table (15) shows the parameters of the SA algorithm.

Because of the stochastic nature of the SA algorithm, this paper tested the algorithm on 10 different random generator seed values. Table (16) shows the objective function results of the 10 runs in ascending order by NotEugenicLeadTime(T). It highlights, that NotEugenicLeadTime individuals generally

if *Individual = Eugenic individual* **then**
  Overwriting the original individual
else
  if $\mathsf{T}_{\text{unperturbed}} < \mathsf{T}_{\text{perturbed}}$ *or Eq. (34) holds* **then**
    Overwriting the original individual
  end
end

Algorithm 1: Individual overwriting procedure

reached lower lead time T. The deviation highlights the stochastic nature of the algorithm, but every result is acceptable.

Figure (1) shows iteration by iteration the best eugenic and not eugenic individuals of the SA algorithm (Seed=1: SA+1) for PLF based routing optimisation. Axes X visualise the iterations and axes Y visualise the objective function values (T). The red (initially lower) points show the decreasing lead time of 1 eugenic individual. The green (initially upper) triangles show the changing of the not eugenic individuals' lead time. The decreasing number of accepted weaker solutions shows the nature of the SA algorithm. It could be said that the best not eugenic individuals exceeded the eugenic individual. The graph proves, that the eugenic individual was never overwritten by a weaker individual. Furthermore, the not eugenic individual was overwritten by a weaker individual with a decreasing probability.

Table (17) shows the top 10 individuals after SA optimisation. The best solutions needed some reconstructions for the defined picking task in the case of the given SLA. It proves the importance of PLF based order picking optimisation to reach a lower picking lead time. The SA algorithm found significantly better solutions than the eugenic individual ("I1"). The results highlighted that the SA algorithm is able to define a reliable solution for the PLF based routing problem.

Parameter	Value
N_{ind}	50
N_{iter}	1200
τ	10

Table 15: Parameters of the SA algorithm

Scenario	EugenicLeadTime	NotEugenicLeadTime
SA+1	0:12:46	0:12:15
SA+2	0:12:38	0:12:17
SA+3	0:13:34	0:12:17
SA+4	0:12:57	0:12:26
SA+5	0:12:24	0:12:57
SA+6	0:13:17	0:12:17
SA+7	0:12:34	0:12:31
SA+8	0:12:34	0:12:45
SA+9	0:12:28	0:12:28
SA+10	0:12:31	0:12:24
Min	0:12:24	0:12:15
AVG	0:12:46	0:12:28
Deviation	0:00:23	0:00:14

Table 16: SA results of the 10 seeds

8 Conclusion

Several warehouses handle products that require the right picking sequence and stacking method to build stable Unit Loads and to avoid product damages during order picking. The proposed research defined the Pallet Loading Features (PLF), the Order Picking Routing Problem based on Pallet Loading Feature (OPRP-PLF), and those influencing factors.

The warehouse management has to make several tactical and operative decisions to operate cost and time effective PLF, as it will influence the order picking processes to perform proper ULs. While PLF is not relevant at every warehouse, the examination of its importance is essential on tactical level. If PLF is relevant, the operating algorithms should take into consideration the PLFs with the right weighting.

Several operational decisions have an impact on order performance lead time and costs, whose minimisation is a general goal. The main PLF relevant decisions are: how to separate the customer orders to UL order picking lists? How many UL and which UL picking list contents will result in the lowest order picking lead time? If the actual SLA won't let us follow the pallet loading rules and collect the ordered items with the shortest route distance, the picking operator or the routing algorithm should decide whether to collect items in the right stacking sequence and move a longer distance or to pick with a

Ind. ID	T	T_P	T_R	T_T
I43	0:12:15	0:03:20	0:00:45	0:08:10
I1	0:12:46	0:03:20	0:00:15	0:09:11
I14	0:12:47	0:03:20	0:01:30	0:07:57
I6	0:12:58	0:03:20	0:01:00	0:08:38
I31	0:13:08	0:03:20	0:01:15	0:08:33
I3	0:13:12	0:03:20	0:01:00	0:08:52
I16	0:13:23	0:03:20	0:00:45	0:09:18
I33	0:13:24	0:03:20	0:01:45	0:08:19
I32	0:13:26	0:03:20	0:02:00	0:08:06
I25	0:13:26	0:03:20	0:00:15	0:09:51

Table 17: Top 10 individuals of SA+1

shorter routing and spend more time redesigning the contents of the UL during picking. The decision should result in the shortest lead time.

This paper described the methodology for defining the importance of PLF on a tactical level. It introduced how the measurement of warehousing processes can highlight OPRP-PLF. It described the methodology for modelling and evaluating pallet loading rules. The defined PLFDM is the basis of PLF based order picking optimisation, which defines a possible picking sequence to avoid product damages and increase the OPP effectiveness.

Two main cases of customer order fulfilment were explained. The first is when the customer order can be picked into 1 UL and order separation is not necessary. The second is when the customer order should be picked into more than 1 UL and order separation is necessary. The examined cases have some alternative sub-cases, which is necessary to examine in further research. This paper determined a formula for both cases to calculate the possible number of order picking sequencing variations and examined its behaviour in the case of several order picking lists whose length and contents are different. The results proved, where PLF is relevant, the possible sequencing combinations of order picking lists that have an exponential growth. This fact proves the necessity of the heuristic optimisation method for OPRP-PLF, for example Foldesi et al. defined for the road transport travelling salesman problem or Theyset al. and Chen et al. defined for warehouse routing problem [13, 23, 9]. The order picking routing optimisation algorithm has to support the operational decisions, like customer order separation for ULs and the longer distance versus more UL reconstruction time trade-off.

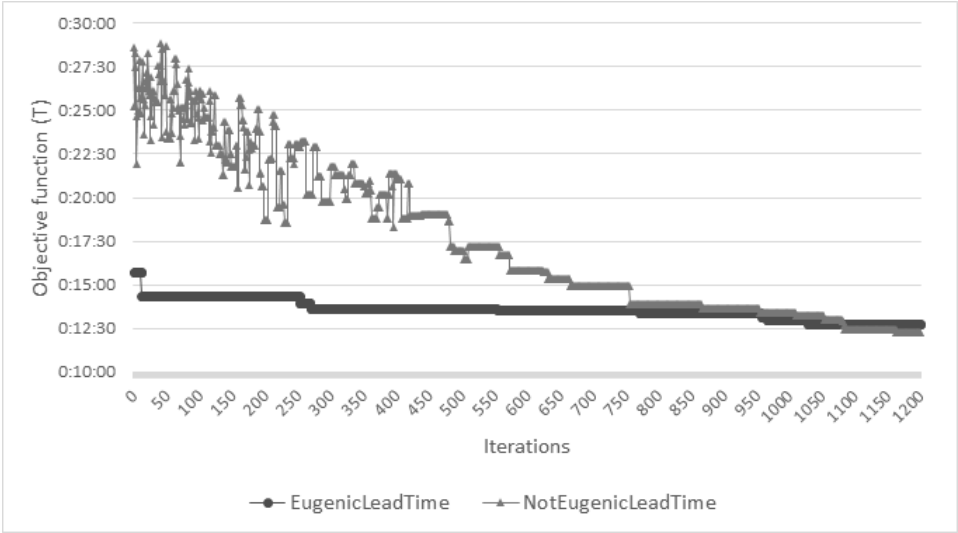


Figure 1: Iterations of the SA algorithm

The paper described some analytic examples for simple cases to introduce operational decisions, which can be made by operators themselves. A Simulated Annealing (SA) algorithm was defined for optimising complex and long picking lists. It resulted in better solutions than a simply sequenced and deterministic optimised eugenic individual. However, as a further research, more effective and quicker algorithms should be developed.

The routing optimisation is usually running on given Storage Location Assignment, whose optimisation and harmonisation with routing are essential. As part of the proposed research SLA algorithms will be developed which take into consideration PLFs and be harmonised with the nature of the orders. The generated SLA alternatives will be evaluated with the developed PLFDM based routing algorithms. The nature of customer demands and the product lines are usually changing, which requires continuous re-engineering of OPP. When the items are dedicated to a position and the SLA is not optimised dynamically, the picking distances will possibly be growing and the routing optimisation will be more important. The increasing order picking lead time usually highlights the necessity of SLA re-engineering. A well-defined Performance Measurement system is the basis of OPP optimisation. It helps realise the changed nature of orders and it highlights the necessity of OPP re-engineering.

The outputs of distribution warehouses are transportation ULs, which are homogeneous or inhomogeneous order picked ULs. These ULs' structure,

strength, transportation requirements, and behaviour during transport have a huge impact on supply chain management, influence its effectiveness, and have impact on the possible risks during transportation. The risk management, which is an examined field by many researchers, like [7, 15] also has to take into consideration the PLF to examine the possibilities of product damages during transport.

Understanding the nature of OPP is the first step in harmonising the warehousing processes. The realised warehouse dependent factors have to be implemented into OPP algorithms with the right weighting. The OPP development should be continuous based on well-defined Performance Measurement indicators to harmonise the warehousing system with the changing environment.

Acknowledgements

The author acknowledges the financial support of this research by the Project EFOP-3.6.1-16-2016-00017. Internationalisation, initiatives to establish a new source of researchers and graduates, and development of knowledge and technological transfer as instruments of intelligent specialisations at Széchenyi István University.

References

[1] R. Alvarez-Valdes, F. Parreño, J. M. Tamarit, A tabu search algorithm for the pallet loading problem, *OR Spectrum*, **27,** (2005) 43–61. ⇒164

[2] J. Ashayeri, M. Goetschalckx, Classification and design of order picking, *Logistics Inf Manage*, **2,** 2 (1989) 99–106. ⇒163

[3] E.E. Bischoff, Three-dimensional packing of items with limited load bearing strength, *European Journal of Operational Research*, **168,** 3 (2006) 952–966. ⇒164

[4] T. Bódis and J. Botzheim, Modelling Order Picking Sequencing Variations of Pallet Setup Clusters, *Proc. of The International Conference on Logistics and Sustainable Transport 2015*, Celje, Slovenia, 2015, pp. 86–94. ⇒165, 169, 170, 171, 172, 174

[5] T. Bódis, J. Botzheim, P. Földesi A Simple Case of Pallet Setup Features Based Order Picking Routing Optimization, *Acta Technica Jaurinensis*, **9(3),** (2016) 204-215. ⇒165, 170, 171, 172, 173

[6] A. Bortfeldt, G. Wäscher, Constraints in container loading A state-of-the-art review, *European Journal of Operational Research*, **229,** 1 (2013) 1–20. ⇒164

[7] W. Chang and A. E. Ellinger and J. Blackhurst, A contextual approach to supply chain risk mitigation, *The International Journal of Logistics Management*, **26,** 3 (2015) 642–656. ⇒192

[8] F. T. Chan and H. Chan, Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage, *Expert Systems with Applications*, **38,** (2011) 2686–2700. ⇒164

[9] Tzu-Li Chen and Chen-Yang Cheng and Yin-Yann Chen and Li-Kai Chan, An efficient hybrid algorithm for integrated order batching, sequencing and routing problem, *International Journal of Production Economics*, **159,** (2015) 158–167. ⇒164, 190

[10] K. L. Choy and H. Y. Lam and L. Canhong and C. K. H. Lee, A Hybrid Decision Support System for Storage Location Assignment in the Fast-Fashion Industry, *Technology Management in the IT-Driven Services (PICMET), 2013 Proceedings of PICMET '13*, San Jose, California, USA, 2013, pp. 468–473. ⇒163

[11] Yi-Ping Cui and Yaodong Cui and Tianbing Tang, Sequential heuristic for the two-dimensional bin-packing problem, *European Journal of Operational Research*, **240,** 1 (2015) 43–53. ⇒164

[12] R. de Koster and T. Le-Duc and K. J. Roodbergen, Design and control of warehouse order picking: A literature review, *European Journal of Operational Research*, **182,** (2007) 481–501. ⇒163

[13] Péter Földesi and János Botzheim and László T. Kóczy, Eugenic bacterial memetic algorithm for fuzzy road transport traveling salesman problem, *International Journal of Innovative Computing, Information and Control*, **7,** 5b (2011) 2775–2798. ⇒190

[14] R. Gamberini, B. Rimini, M. Dell'Amico, F. Lolli, M. Bianchi, Design and Optimization of Picking in the Case of Multi-Item Multi-Location Multi-Pallet Customer Orders, *Warehousing in the Global Supply Chain, Springer London*, 2012, 397–424. ⇒163

[15] U. Jüttner, Supply chain risk management: Understanding the business requirements from a practitioner perspective, *The International Journal of Logistics Management*, **16,** 1 (2005) 120–141. ⇒192

[16] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, *Science*, **220,** 4598 (1983) 671–680. ⇒186

[17] H.C.W. Lau and T.M. Chan and W.T. Tsui and G.T.S. Ho and K.L. Choy, An AI approach for optimizing multi-pallet loading operations, *Expert Systems with Applications*, **36,** 3 (2009) 4296–4312. ⇒164

[18] G. H. A. Martins and R. F. Dell, Solving the pallet loading problem, *European Journal of Operational Research*, **184,** 2 (2008) 429–440. ⇒164

[19] K. Moeller, Increasing warehouse order picking performance by sequence optimization, *Procedia Social and Behavioral Sciences*, **20,** (2011) 177–185. ⇒164

[20] B. Molnár and Gy. Lipovszki, Multi-objective routing and scheduling of order pickers in a warehouse, *International Journal of SIMULATION*, **6,** 5 (2005) 22–33. ⇒165

[21] R. Saraiva, A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company, *IFAC-PapersOnLine*, **48,** 3 (2015) 490–495. ⇒164

[22] J. Y. Shiau and M. C. Lee, A warehouse management system with sequential picking for multi-container deliveries, *Computers and Industrial Engineering*, **58,** (2010) 382–392. ⇒165

[23] C. Theys and O. Bräysy and W. Dullaert and B. Raa, Using a TSP heuristic for routing order pickers in warehouses, *European Journal of Operational Research*, **200,** 3 (2010) 755–763. ⇒164, 190

[24] S. Webster and R. A. Ruben and Kum-Khiong Yang, Impact of Storage Assignment Decisions on a Bucket Brigade Order Picking Line, *Production and Operations Management*, **21,** 2 (2012) 276–290. ⇒164

Received: October 13, 2017 • Revised: December 16, 2017

DE GRUYTER OPEN Acta Univ. Sapientiae, Informatica **9,** 2 (2017) 195–213

DOI: 10.1515/ausi-2017-0012

Hierarchical k_t jet clustering for parallel architectures

Richárd FORSTER
Eötvös University
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös University
email: fulop@caesar.elte.hu

Abstract. The reconstruction and analyze of measured data play important role in the research of high energy particle physics. This leads to new results in both experimental and theoretical physics. This requires algorithm improvements and high computer capacity. Clustering algorithm makes it possible to get to know the jet structure more accurately.

More granular parallelization of the k_t cluster algorithms was explored by combining it with the hierarchical clustering methods used in network evaluations. The k_t method allows to know the development of particles due to the collision of high-energy nucleus-nucleus.

The hierarchical clustering algorithms works on graphs, so the particle information used by the standard k_t algorithm was first transformed into an appropriate graph, representing the network of particles. Testing was done using data samples from the Alice offline library, which contains the required modules to simulate the ALICE detector that is a dedicated Pb-Pb detector. The proposed algorithm was compared to the FastJet toolkit's standard longitudinal invariant k_t implementation. Parallelizing the standard non-optimized version of this algorithm utilizing the available CPU architecture proved to be 1.6 times faster, than the standard implementation, while the proposed solution in this paper was able to achieve a 12 times faster computing performance, also being scalable enough to efficiently run on GPUs.

Computing Classification System 1998: I.1.4
Mathematics Subject Classification 2010: 58A20
Key words and phrases: jet, cluster algorithm, hierarchical clustering, database of experimenthal particle physics parallel computing, multi-core, C++11

1 Introduction

We researched the behaviour of the jet in the high energy physics [20, 25, 30] using the many-core architecture of the modern CPUs [18, 19]. We applied the most important principles of physics and we review the main concepts in this article.

The basic conception is the parton model to study the high energy hadron collisions. Due to the hadronisation process we can measure the final state object. The theory of strong interaction between the quark and gluon is described by Quantum Chromodynamics (QCD) [26]. In this process colored objects are created i.e. quarks and gluons. The scale of this procedure is few fermi 10^{-15} m. The short and long distance physics are fundamentally different. The colored objects are free to move within short range. On the scale of a few centimeter the colored objects become confined into color singlets. The process of quarks and gluon showering is a hadronization. During this procedure many mesons and baryons emerge they decay and form evolve the final state objects which are measured by detector. The spray of hadron is called jet which is a connection between the short scale physics and a final state measured particles [9, 28].

Different type of data requires different approaches to clusterize the input. For real world networks, hierarchical algorithms are used to compute the clusters. By combining some aspects of hierarchical processes and k_t jet clustering, a more efficient process will be made available for generating jets. This proposed algorithm has lower complexity compared to the k_t solution and has faster computation on the same hardware, while also being more scalable, that further enables jet generation on many-core architectures, such as GPUs. The Louvain method, used as the hierarchical clustering algorithm for the basis of the new process already proved to be scalable on both CPUs and GPUs. Combining the two methods provided a final algorithm, that can process 12 times faster, than the standard k_t clustering.

2 Clustering and declustering mechanism

In this Section we discuss the clustering of jets and it is followed by declustering in order to fine the substructures. There are many articles about this question in the literature [1, 16, 32]. We consider the most important algorithms [3] and discuss the physical meaning of these processes.

2.1 Jet

A jet is a narrow cone of hadrons and other particles which are formed by quark and gluon during the high-energy collisions. From the hard subprocess hadronize the created hadrons become collimated around original parton directions and the higher energy parton takes shapes more collimated. These bunches of hadrons are called jets (Figure 1). They can be interpreted as a link to partons and it can yield to understand a deeper level of the important parton interactions. Jets sketch forms a rather simple process, what happened in an event without taking into calculation the multiparticle dynamics. Therefore we use the jets, rather the directly measured hadrons, these can be constructed as infrared-safe observables. The QCD is theoretical background to calculate the predictions of jets with high precision.

2.2 Clustering of jet

The jet is clustered, when we study the jet momentum due to the final state particles in the calorimeter [5]. The results can be made more accurate to consider the other measured quantities as muon systems. All together it means the clusters. By theoretical research we have to mention two questions. These are the infrared (IR) safety and collinear safety[28].

An observable is infrared safe, if it does not depend on the low energy physics of the theory. We speak about the collinear(C) safety, when a parton is replaced by a collinear pair of partons, then it should not modulate the jet clustering results. The properties of jet does not change, when one of the particles radiates a very soft objects, or breaks up two collinear particles. Therefore the jet can be determined by perturbative methods to compare with the experiment.

In the theoretical physics the infrared divergence means that situation, when an integral of Feynman diagram diverges because of the constituent objects with very small energy goes to zero. It is important, when the model contains massless particles, as photons. One possibility to deal with it is to apply an infrared cutoff and it approaches zero. The divergence is usually remains finite in all measurable quantities. Therefore the infrared safe and collinear safe jet reconstruction algorithm can be used to the evaluation of the measured data responsing to theoretical condition or it is applied to a given order thanks to the algorithm becomes IRC safe.

The determination of the jets mass and energy depend on the size of jet radius. In the case of larger jet radius these quantities can be calculated more

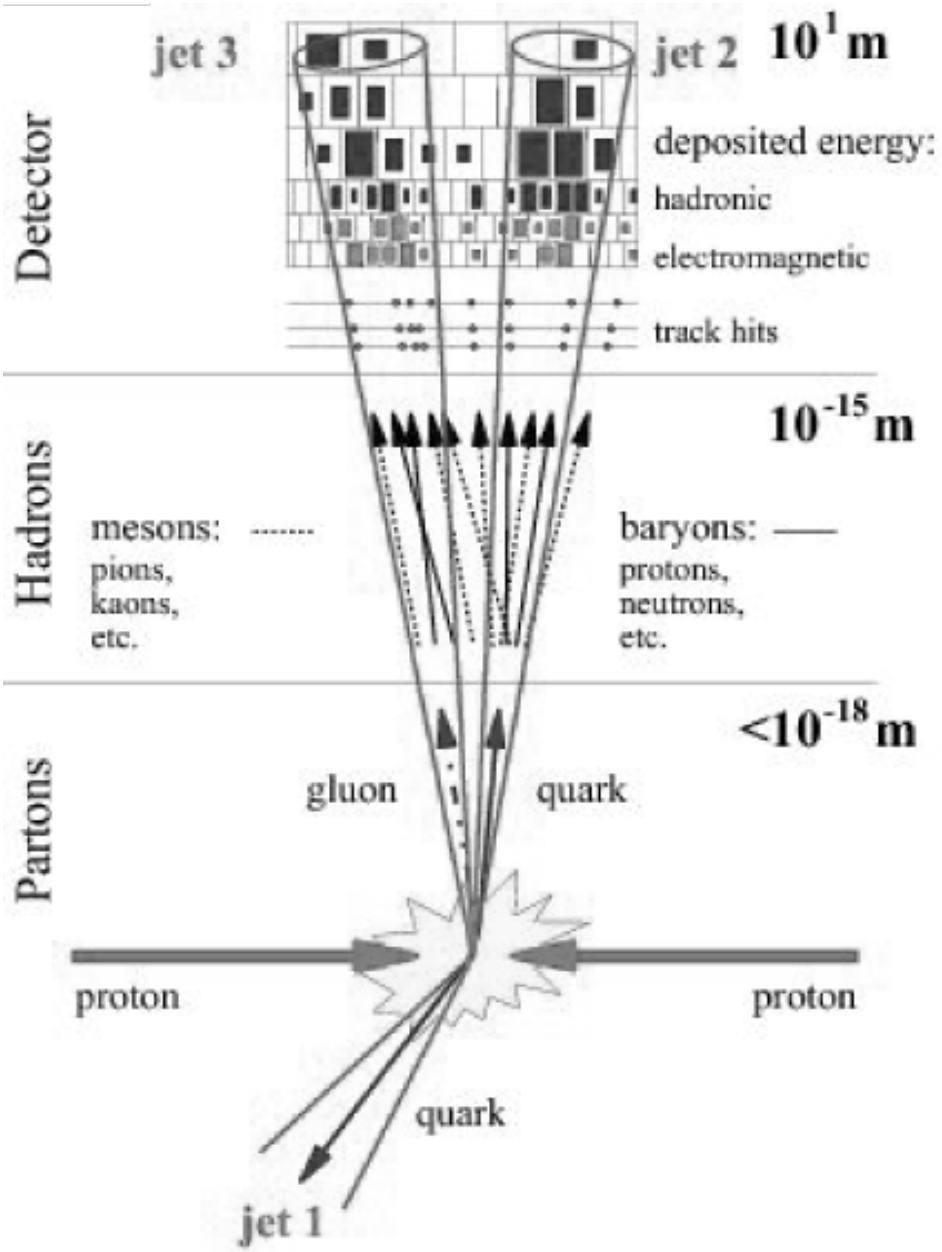


Figure 1: Structure of jet

exactly than smaller distance because the cluster contains more hadronised particles, which includes the underlying event and pile-up.

2.3 Cone and sequential recombination algorithms

We discuss two important classes of jet algorithms. The first is the cone algorithm. This methods are the iterative cone with progressive-removal (IC-PR) [2], The iterative cone with split-merge (IC-SM) [7] and seedless infra-red safe cone (SIScone) [31].

The second part of the algorithm is the sequential clustering algorithms. We will introduce the K_t [14], Anti-K_t [12] and the Cambridge/Aachen [34] algorithms.

2.3.1 Cone algorithms

In the case of cone algorithm a conical region contains the particles of jet, so the cluster takes place in the $(\eta - \phi)$ space. Therefore the jet has rigid boundaries. This algorithm was popular in the experimental physics, because it could be easier implemented, but it was not so preferred in the theoretical physics. The cone algorithms are IRC unsafe.

IC-PR The iterative cone algorithm together progressive removal is a collinear unsafe algorithm.

Look for that cell which has the largest p_t i.e. the hardest box. It becomes a center. Let us generated the radius of cone R around the center. We can determine the trial jet axis to sum up the cells inside the cone by four-vectors. If the trial jet axis corresponds to center axis, the behaviour of cone is stable. Each particles which are situated in the stable cone are deleted from the list of particles. This method is repeated by the next hardest cell. But if the trial jet axis does not correspond to the center of axis, thenn the trial jet axis need to look for the another center axis. This method is repeated until convergence of the axes occurs. This process is repeated until there are center above threshold energy E_{cut}.

IC-SM The iterative cone algorithm together with the split merge method is an infra-red unsafe algorithm. For example JetClu, midpoint cone. This process of the jet is as follows:

The first all cells which are above a threshold energy E_{cut} are center. Look for all stable cones together with those center applying the same method as in the IC-PR algorithm, but we do not delete any particles from the list once, a stable cone is discovered. Each stable cones which were recovered are written

as ptortojets. A split merge method is run on the protojets until we do not discover each of them.

SIScone That infra-red safe cone method which does not have center is a IRC safe cone algorithm. Because the area is relatively small, therefore it works by the UE and PU well. It has better results than the R for hard radiation, therefore has good resolution, but it is wrong for resolving multijets. The process of SIScone of the jet is written schematically:

1. Choose a particle i
2. Look for each particles j within distance $2R$ of i
3. If there is no more particle j
4. then i is a stable cone and write to the list of protojets
5. Else
6. Look after the circles which is generated by i and j. These are lying on their circumference and determine the momenta of the cones.
7. For each circle
8. All four permutations of the two edge points which are situated on or out of the circle. These four circles mean as current cones.
9. Each current cone, which was not previously looked for.
10. It must be decided that the current cones are situated in or out of the edge particles. This is same as the cone determined by the momentum of the particles in the current cone. If not, then the current cone is unstable.
11. Check each current cones which are not unstable and create an explicit stability one. Write each stable cones to the list of protojets.
12. Run a split merge method on the protojets.

2.3.2 Sequential clustering algorithms

The sequential clustering algorithms [3] can be used when the particles are situated in the jets and there are small differences in the transverse momenta. Therefore so called groups particles can be written on the momentum space in jets, which contain fluctuating areas in $(\eta - \phi)$ space. The sequential clustering algorithms were preferred by theorists. This method do not have been favoured by experimentalists, because it has slow implementation. The FastJet program [13] provides such clustering algorithms which are speed enough for experimental research. Sequential clustering algorithms are also IRC safe.

We introduce the basic idea of sequential clustering algorithms. The first we mention the distance variable between two particles $d_{ij} = min(p_{ti}^a, p_{tj}^a) \times \frac{R_{ij}^2}{R}$, where a is an exponent corresponding to a particular clustering algorithm. We define distance $R_{ij}^2 = (\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2$ between two particles in the $(\eta - \phi)$

space and R is the radius parameter which determines the final size of the jet. The value is changing in the range [0.4,0.7]. The second distance variable is $d_{iB} = p_{ti}^{a}$ which is the momentum space distance between the beam axis and the measured particle.

In the sequential clustering algorithms the first step is to look for the minimum of the entire set $\{d_{ij}, d_{iB}\}$. If d_{ij} is the minimum distance then the particle i and j are integrated in one particle (ij) we need to take the summation four-vectors and i and j are deleted from the list of particles.

If d_{iB} is the minimum value, then i becomes a final jet and it is deleted from the list of particles.

We need to continue this method until either each particles are part of jet, where the distance between the jet axes R_{ij} larger than R, this process is the inclusive clustering. Or until a designed the total of jets have been looked for. We call it exclusive clustering.

K_t method In the case of K_t algorithm [12, 14] the a value equals to 2, then the equations follows this form:

$$d_{ij} = \min\left(p_{ti}^2, p_{tj}^2\right) \times \frac{R_{ij}^2}{R} \tag{1}$$

$$d_{iB} = p_{ti}^2 \tag{2}$$

The K_t algorithm [15] is applied mainly to cluster the soft particles, because the particles have low p_t particularly effecting the fluctuates in area appreciably and a method that is susceptible to the UE and PU [23]. Because the method of clustering is efficient process, therefore K_t algorithm works well at resolving subjets.

Anti-K_t method The value a equals to -2, then we speak about Anti-K_t. The form of equation is the following:

$$d_{ij} = \min\left(\frac{1}{p_{ij}^2}, \frac{1}{p_{ij}^2}\right) \times \frac{R_{ij}^2}{R} \tag{3}$$

$$d_{iB} = \frac{1}{p_{ti}^2} \tag{4}$$

The equation (3) is dominated by high p_t, therefore this method can be applied to cluster hard particles mainly. So the area fluctuates slightly and the process is slightly susceptible to the UE and PU. The Anti-K_t's clustering preference

results in a method, namely it has the optimum at resolving jets, but therefore it has poor de-clustering.

Cambridge/Aachen The value a equals to 0 which yields the C/A algorithm, which provides the following equations:

$$d_{ij} = \frac{R_{ij}^2}{R}$$

$$d_{iB} = 1$$

Both of the distance variables do not depend on the momentum, therefore its area fluctuates poorly and slightly susceptible to the UE and PU. Because the spatial property of the distance variable is little, therefore C/A de-clusters is optimum for studying jet substructure, but it is poorly more complicated to de-cluster than the K_t algorithm.

2.3.3 Sequential algorithms: the FastJet package

FastJet [13] is a software package which is used to determine the cluster jets. It is an open source program.

The basic reconstruction algorithm [11] has been further developed to faster software including the array structure for the distance between the objects.

The original program implementation provides the next calculation of the demand:

First we determine the d_{ij} distance between all the particles and the d_{iB} between all the objects. This calculation needs $O(N^2)$.

Second we search the closest particles and that objects which are nearest neighbour, i.e. the minimal value of the distance d_{ij} and d_{iB}. This calculation disposes $O(N^2)$ and it is done N times.

Then we can perform the jet reconstruction to apply the advanced pairs and cluster of measured data set. The calculation of the algorithm in particular the cluster provides $O(N^3)$.

FastJet algorithm employs two arrays to solve the original processes. One of them is applied for the distance between the closest objects, another contains the distance of the beamline. The calculation demand is $O(N^2)$.

Further development of the FastJet software achieves $O(N \ln N)$ calculation to use another applicable metric instead of the array structure for the distance of closest particles.

That clusters and jets, which were produced by FastJet, are saved in pseudojets. It plays important role, because it allows to reconstruct the structure

of the clusters. The pseudojets include the four-momentum and the hierarchical time dependent events inside the cluster, this results all objects in the jet and it plays important role for the declustering mechanism.

2.4 Substructure of jet

Substructure of jets can be one jet containing more than one group of gaussian-distributed clusters. Substructure can also be a non gaussian component, which is corresponds to an offset. It can as well consist of another gaussian group of clusters, namely second hard jet.

Three different types which is able to define:

I: Subjet from uncorrelated sources, overlapping the hard jet considered or clustered together with it. It is soft process, originating from proton-leftovers, initial state radiation, beam-rests and/or scatterings, e.g. pileup (PU) and underlying event (UE).

II: Subjet from correlated sources, clustered together with the hard jet considered, originating from the same primary vertex, but another branch of the Feynman diagram.

III: Subjet from correlated sources, originating from the decay of a single boosted particle, clustered together into a single jet.

3 Problem statement and notation

Let $G(V, E, \omega)$ be an undirected weighted graph, with V representing the set of vertices, E the set of edges and ω a weighting function that assigns a positive weight to every edge from E. If the input graph is unweighted, then the weight of the edges is considered to be 0. The graph is allowed to have loops, so edges like (i, i) are valid, while multiple edges between the same nodes should not be present. The following will be the adjacency list of vertex i: $\Gamma(i) = j|(i, j) \in E$. Let δ_i denote the weighted degree of vertex i, such as $\delta_i = \sum_{j \in \Gamma(i)}\{\omega(i, j)\}$. Let N denote the number of vertices in graph G, M the number of edges, and W the sum of all edge weights, such as $M = \frac{1}{2}\sum_{i \in V} \delta_i$. By computing the communities, the vertex set V will be partitioned into an arbitrary number of disjoint subsets, each with size n, where $0 < n \leq N$. $C(i)$ will denote the community containing vertex i. $E_{i \to C}$ is the set of all edges connecting vertex i into community C. Consequently let $e_{i \to C}$ hold the sum of the edge weights in $E_{i \to C}$.

$$e_{i\to C} = \sum_{(i,j)\in E_{i\to C}} \omega(i,j) \tag{5}$$

The sum of all the vertices in community C shall be denoted by deg_C, which will represent the degree of the whole community.

$$deg_C = \sum_{i\in C} \delta_i \tag{6}$$

3.1 Modularity

Let $S = C_1, C_2, ..., C_k$ be the set of every community in a given partitioning of V, where $\{1 \leq k \leq N\}$. Modularity Q of partitioning S is given by the following [27]:

$$Q = \frac{1}{2W} \sum_{i\in V} e_{i\to C(i)} - \sum_{C\in S} \left(\frac{deg_C}{2W} \cdot \frac{deg_C}{2W} \right) \tag{7}$$

Modularity calculation is a common solution to measure the quality of the process and also to define a termination function. Still it's not without some drawbacks, like the resolution limit [21, 33]. Definition can be given in multiple forms as are described in [33, 4, 6]. In the literature the more widespread version is defined in Eq. 7, also this is used in the Louvain method [8].

3.2 Community detection

Given a $G(V, E, \omega)$ graph as input, the expected result is partitioning S of communities that leads to the greatest modularity. This problem is known to be NP-complete [10]. The major difference compared to other partitioning solutions is the number and size of the clusters, while also in specific cases some initial distribution is given [22].

4 The Louvain algorithm

The Louvain method [8], gives an iterative, greedy algorithm, that produces the communities in multiple phases. Each phase runs for many iterations until the system is converging. As the initialization of the first phase each vertex will belong to a set containing only that node. As the process goes on, through the iterations the gain in modularity becomes lesser and the iterating stops when a predefined threshold is reached. In every round the vertices are checked in an

arbitrary, but predefined order. Visiting vertex i, the neighbors are explored, searching for a new community with the highest modularity gain. Once this calculation is done, the selected neighbors community will be selected and vertex i will be moved there. If this search yields no suitable community, no changes are made to the current node. One iteration lasts until all vertices are examined. Because of this the modularity is a monotonically increasing. By reaching convergence in a given phase, the system is getting reduced, by assigning a single "meta-vertex"[24] in the place of all the nodes belonging to the same community. The new nodes can have loops and the weight of these new edges will be the sum of the weights of all the edges that are connecting the inner nodes of the group. For an edge pointing into another cluster, the weight is calculated by summing the weights of all the edges between the connected sets. The result will be a reduced graph $G'(V', E', \omega')$, which becomes the input for the next phase. At any given iteration, $\Delta Q_{i\to C(j)}$ holds the modularity gain resulting from the reassignment of vertex i from its current community $C(i)$ to a neighboring $C(j)$. This is given by:

$$\Delta Q_{i\to C(j)} = \frac{e_{i\to C(j)}}{W} + \frac{2\cdot\delta_i\cdot mod_{C(i)/i} - 2\cdot\delta_i\cdot mod_{C(j)}}{(2W)^2} \tag{8}$$

For any vertex i the new community will be computed based on the following. For $j \in \Gamma(i)\cup\{i\}$:

$$C(i) = arg\max_{C}(j)\Delta Q_{i\to C(j)} \tag{9}$$

Because the modularity is monotonically increasing it guarantees termination. By running only for a dozen iterations during a few phases, this method can find the clusters in real world datasets.

4.1 Parallel heuristics

The challenges to parallelize the Louvain method were explored in [24]. To solve those issues multiple heuristics were introduced, that can be used to leverage the performance of the parallel systems in a basically sequential algorithm. From the proposed heuristics two is going to be detailed in this paper. Lets assume the communities at any given stage are labeled numerically. The notation $l(C)$ will return the label of community C.

4.1.1 Singlet minimum label heuristic

In the parallel algorithm, if at any given iteration vertex i which is in a community by itself ($C(i) = i$, singlet community [24]), in hope for modularity gain might decide to move into another community, that holds only vertex j. This transition will only be applied if $l(C(j) < l(C(i)))$.

4.1.2 Generalized minimum label heuristic

In the parallel algorithm, if at any given iteration the vertex i has multiple neighboring communities providing modularity gains, the community with the minimum label will be selected. Swap situations might occur, when two vertices are transitioning into the other's community in the same iteration. This can delay the convergence, but can never lead to nontermination as the minimum required modularity gain threshold will guarantee a successful termination.

5 Hierarchical jet clustering

The hierarchical clustering is based on the Louvain clustering (Section 4). To work with it, the algorithm was tweaked to incorporate the specific needs of the jet clustering.

The Louvain method works on a weighted, undirected graph, while on the other hand jet clustering (Subsection 2.2) uses a list of input particles, hence this list needs to be transformed into a graph. Obviously the particles themselves will be appropriate for the nodes. As the k_t clustering uses the distance between the elements, a function assigning an edge to the nodes with the distance between the two adjacent items is a logical solution. The problem in this case is the sheer volume of edges, as this will create a fully connected graph with $n * (n-1)/2$ total links. Instead making the connection between nearest neighbours and second to nearest proves to be sufficient. Also, the edges will now contain directionality. In those cases, when the particle's nearest "neighbour" is the beam, the node in graph will be isolated, and will represent a singular jet. The original hierarchical process is greedy in the sense, that it relies on modularity gain to drive the computation. This is important, because there is no information about the clusters before starting the computation. While using the k_t algorithm it is known, that the processing will end, when all particles are assigned into a jet, thus eliminating the need to compute the modularity for the graph.

The result of a hierarchical clustering is the dendogram. This tree will con-

tain the connection between the different phases of the cluster generation, leading to the final assignments. Originally this is implemented by generating a level of this tree, a new graph is induced by renumbering the nodes based on their actual cluster assignment and recalculate the edges for them. Jet clustering doesn't need the inner weight of the clusters, only the calculated distance from that subjet and that will be incorporated into the graph itself through the edges. The solution proposed here doesn't require the regeneration of each levels graph, but the graph will be dynamically morphed, by applying the changes through the different phases. Also the terminology used for the generation in the original Louvain algorithm is to move the nodes into clusters, where each individual node will check which cluster will be the best in the actual phase. Here the nodes connected with a directed edge will decide among each other in such a way, that the node pointing to another one will draw that to itself.

5.1 Sequential processing

Initially the result of the Louvain clustering depends on the original order of the input value. The same can be said about the k_t clustering (Subsection 2.3.2) as well: always the two closest elements are combined into a new jet, thus the input should be ascending ordered. This way always the first element of this list will be tested. After generating the new item, its distance should be calculated against the remaining elements and finally it needs to be inserted into the input array using a sorted insert based on the calculated distance and a link will be generated between this item and its nearest neighbour, while the links to the original subjets will be removed. The algorithm checks after this recombination if the other elements nearest neighbour is the new recombined jet or a completely different item. If the distance between an element and the recombined jet becomes bigger, than how much for its individual part was, then have to check if there are additional nearer neighbours and the closest of them will be linked too and the two edges will be set this way. The processing continues until any of the original input particles are present.

5.2 Parallel processing

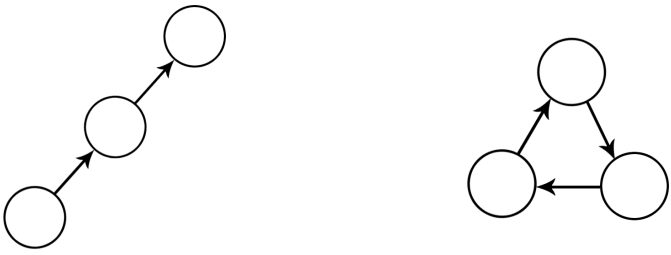
The evaluation of the original hierarchical clustering in parallel required some additional heuristics (Subsection 4.1) to keep the precision and consistency of the base algorithm. By introducing the method to a more complex clustering process, further requirements have to be satisfied to compute the final inclusive

jets. These additional heuristics are closely connected to the structure of the generated graph.

5.2.1 Transitivity effect

Let's call the transitivity effect the case, when multiple particles or subjets are having at least 1 input and 1 output edge. The different problems, that can rise from this are explored in this subsection.

Processing in parallel in one phase multiple nodes will try to merge themselves with the connected nearest neighbour, while also that neighbour might do the same at the same time, building up a chain. Just following through that chain and merging the nodes together, might omit a potential change in the nearest neighbours, that might appear by computing the separate recombined subjets. In a simple case (Figure 2a) this can be solved, by only applying the draw from the node, that is not tried to be recombined by another node, meaning the node doesn't have incoming edges.



(a) Chained nodes in the graph (b) Cycle in the graph

Figure 2

A more complex case might involve cycles (Figure 2b) on this chain, where all the nodes have inbound links and the previous solution can't be applied. While processing the graph, if no cycles are present, then new subjets will be generated. Even if there are cycles, but still have simple chains, the computation can continue. At one point the clustering will not be able to push any node into a new cluster as only cycles are available, effectively halting the processing. If the clusterization is incomplete, there are nodes, that are not assigned to jets and no cluster assignment is taking place in a given phase, it is known to have cycles only, so there is no need for additional cycle checking.

The next step is to eliminate the cycle. For this the two nodes with the shortest distance needs to be found, thus a minimum search is required. This

way the two elements are getting removed from the chain, with all of its connections. In case of the new subjet invocating a new cycle, this search will be repeated. If the graph has multiple unconnected components (Figure 3), the search can be done in parallel among all the components, but not for connected components, as they might connect to the same cycle.

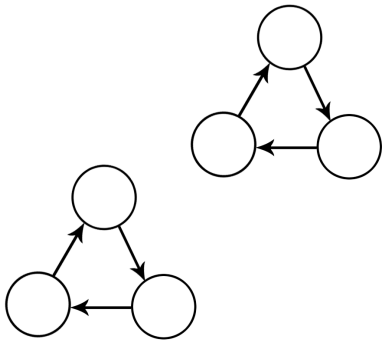


Figure 3: Multiple components in the graph

5.3 Results

The complexity of the k_t jet algorithm is $\Theta(N^2)$, which requires a high amount of computation to be done. It was shown in [18], that by applying parallelization, the runtime of this process can be reduced considerably. Tests running on the system detailed in Table 1, using the raw data from an event (containing 140535 points) simulated with the AliRoot framework's PbPbbench [29] test application.

The system used for development and testing is described in Table 1.

CPU	GPU	OS	Compiler
Intel Core i7 4710HQ	GeForce GTX 980M	Windows 10 Pro	Visual C++ 2013

Table 1: The test system

The previously proposed parallel implementation takes $207,9$ seconds compared to the FastJet [13] (Subsection 2.3.3) framework's sequential k_t implementation, that needs $347,18$ seconds to finish the clustering, giving a 1.67 faster runtime. On the other hand the new hierarchical jet clustering (Section

5) method needs only 81 seconds to conclude the generation, while producing the same output and the parallel implementation takes only 26 seconds to finish the clustering. This leads to a 13 times faster evaluation compared to the sequential k_t algorithm and 8 times faster compared to the parallel implementation of the same jet clustering (Table 2).

Algorithm	Complexity	Runtime
k_t	$\Theta(N^2)$	347,18 s
parallel k_t	$\Theta(N^2)$	207,9 s
hierarchical jet clustering	$O(N)$	81 s
parallel hierarchical jet clustering	$O(N)$	26 s

Table 2: Runtimes of the k_t and parallel k_t algorithm and the hierarchical jet clustering

5.3.1 Complexity

In every step, where a new subjet is generated, it's distance will be computed to the other remaining $n-2$ nodes, where n is the number of nodes in the actual phase. If the computation is running sequentially (Subsection 5.1), the number of nodes decreases by 1 between each phase, while in parallel (Subsection 5.2) it depends on how many subjets will be computed at once. Overall this part will take $(n-2)+(n-4)+\cdots+1$, because the process will continue until all original particles are assigned to a jet. In the worst case it might be, that always subjets will be merged and at the end there will be 1 subjet and 1 particle.

If chains will be present, to break them up can be done in constant time as the node that isn't pulled by someone will merge it's nearest neighbour. In the case of cycles, the complexity comes from finding the unconnected components and doing the minimum search in each of them. To find the minimum edge, the complexity will be $O(N)$. For the component search, if the graph is stored with the list of edges, to find all components it will take $O(N+M)$ steps, where N is the number of nodes and M is the number of edges.

Overall the complexity of the proposed algorithm is $O(N)+O(N)+O(N+M)=O(N)$.

6 Summary

In this paper a new kind of jet clustering algorithm is detailed, that builds on some fundamental characteristics of hierarchical clustering used on real network datasets. Thanks to this approach the $\Theta(N^2)$ complexity of the original k_t jet algorithm was reduced to be linear (Subsection 5.3.1). This and the higher granularity coming from this allows for further parallelization, that greatly helps reducing the time of processing, providing a 13 times faster computation. Also thanks to this GPU based computing for jet clustering becomes implementable and as it was introduced in [17] for the Louvain clustering method, the use of many-core architectures further decreases the runtime even with a factor of 12.

7 Future work

The proposed approach needs more thorough testing, with different data sets to see if the precision always stays the same and produces the same result as the original k_t algorithm. On the other hand the modifications to the hierarchical clustering should be also applied on the full GPU implementation to see the performance benefits of the new solution with extremely parallelizable architectures.

A bigger step will be to see how this solution can be further improved upon, potentially using machine learning in the process and providing a fundamentally different approach to clustering.

References

[1] A. Ali, G. Kramer, Jets and QCD: A historical review of the discovery of the quark and gluon jets and its impact on QCD *Eur. Phys. J. H* **36** (2011) 245–326. [arXiv:1012.2288 [hep-ph]]. ⇒196

[2] G. Arnison et al. [UA1 Collaboration], Hadronic jet production at the CERN proton-antiproton collider, *Phys. Lett. B* **132** (1983) 214. ⇒199

[3] R. Atkin, Review of j-et reconstruction algorithms, *Journ. of Phys.: Conf. Ser.* **645** (2015) 012008. ⇒196, 200

[4] D. Bader, J. McCloskey, Modularity and graph algorithms, *SIAM AN10 Minisymposium on Analyzing Massive Real-World Graphs* (2009) 12–16. ⇒204

[5] F. Beaudette [CMS Collaboration], Performance of the particle flow algorithm in CMS, *PoS ICHEP 2010* (2010) 002. ⇒197

[6] J.W. Berry, B. Hendrickson, R.A. LaViolette, C. A. Phillips, Tolerating the community detection resolution limit with edge weighting, *Phys. Rev. E* **83,** 5 (2011) 056119. ⇒204

[7] G. C. Blazey, J. R. Dittmann, S. D. Ellis, V. D. Elvira, K. Frame, S. Grinstein, R. Hirosky and R. Piegaia et al., *Run II Jet Physics: Proc. of the Run II QCD and Weak Boson Physics Workshop*, [arXiv:hep-ex/0005012] ⇒199

[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, *Journal of Statistical Mechanics: Theory and Experiment* **10** (2008) doi:P10008 ⇒204

[9] M.G. Bowler, *Femptophysics*, Pergamon Press 1990. ⇒196

[10] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, *IEEE Trans. Knowl. Data Eng.* **20,** 2 (2008) 172–188. ⇒204

[11] M. Cacciari, G. P. Salam, *Phys. Rev. Lett* **B641** (2006) 57–61 [hep-ph/0512210] ⇒202

[12] M. Cacciari, G. P. Salam, G. Soyez. The anti-$\mathsf{K_t}$ jet clustering algorithm, *JHEP* **0804** (2008) 063 [arXiv:0802.1189 [hep-ph]]. ⇒199, 201

[13] M. Cacciari, G. P. Salam, G. Soyez, FastJet user manual, *Eur. Phys. J. C.* **72** (2012) 1896, arXiv:1111.6097v1 ⇒200, 202, 209

[14] S. Catani, Y.L. Dokshitzer, M. H. Seymour, B. R. Webber, Longitudinally invariant $\mathsf{K_t}$ clustering algorithms for hadron hadron collisions. *Nucl. Phys. B* **406** (1993) 187224. ⇒199, 201

[15] S. D. Ellis, D. E. Soper, Successive combination jet algorithm for hadron collisions *Phys. Rev. D* **48,** 7 (1993) 3160. ⇒201

[16] S. D. Ellis, J. Huston, K. Hatakeyama, P. Loch, M. Tonnesmann, Jets in hadron-hadron collisions *Prog. Part. Nucl. Phys.* **60** (2008) 484 [arXiv:0712.2447 [hep-ph]]. ⇒196

[17] R. Forster, Louvain community detection with parallel heuristics on GPUs, *20th Jubilee IEEE Int. Conf. on Intelligent Engineering Systems* **20** (2016), ISBN:978-1-5090-1216-9, doi: 10.1109/INES.2016.7555126 ⇒211

[18] R. Forster, A. Fülöp, Parallel $\mathsf{k_t}$ jet clustering algorithm, *Acta Univ. Sapientiae Informatica* **9,** 1 (2017) 49–64. ⇒196, 209

[19] R. Forster, A. Fülöp, Jet browser model accelerated by GPUs, *Acta Univ. Sapientiae Informatica* **8,** 2 (2016) 171–185. ⇒196

[20] R. Forster, A. Fülöp, Yang-Mills lattice on CUDA, *Acta Univ. Sapientiae, Inf.*, **5,** 2 (2013) 184–211. ⇒196

[21] S. Fortunato, Community detection in graphs, *Phys. Rep.* **486,** 35 (2010) 75–174, http://dx.doi.org/10.1016/j.physrep.2009.11.002. ⇒204

[22] B. Hendrickson, T. G. Kolda, Graph partitioning models for parallel computing, *Parallel Comput.* **26,** 12 (2000) 1519–1534. ⇒204

[23] M. Hodgkinson, Missing ET performance in ATLAS, in *Proc. 34th International Conference in High Energy Physics (ICHEP08)*, Philadelphia, 2008, eConf C080730 [arXiv:hep-ex/0810.0181] ⇒201

[24] H. Lu, M. Halappanavar, A. Kalyanaraman, Parallel heuristics for scalable community detection, *Parallel Computing* 47 (2015) 1937 ⇒205, 206
[25] S. Moretti, L. Lonnblad, T. Sjostrand, New and old jet clustering algorithms for electron-positron events JHEP **9808** (1998) 001 [arXiv:hep-ph/9804296]. ⇒196
[26] T. Muta, *Foundation of Quantum Chrodinamics*, World Scientific Press 1986. ⇒196
[27] M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, *Phys. Rev. E* **69,** 2 (2004) 026113. ⇒204
[28] M. E. Peskin, D. V. Schroeder, *Quantum Field Theory*, Westview Press, 1995. ⇒196, 197
[29] D. Rohr, S. Gorbunov, A. Szostak, M. Kretz, T. Kollegger, T. Breitner, T. Alt, ALICE HLT TPC Tracking of Pb-Pb events on GPUs, *Journal of Physics: Conference Series* 396 (2012) doi:10.1088/1742-6596/396/1/012044 ⇒209
[30] G. P. Salam, *Towards Jetography*, *Eur. Phys. J. C***67** (2010) 637–686. [arXiv:0906.1833 [hep-ph]]. ⇒196
[31] G. P. Salam, G. Soyez, A partical seedless infrared-safe cone jet algorithm, *JHEP* **0705** (2007) 086 [arXiv:0704.0292 [hep-ph]]. ⇒199
[32] G. Sterman, S. Weinberg, Jets from quantum chromodynamics, *Phys. Rev. Lett.* **39** (1977) 1436. ⇒196
[33] V. A. Traag, P. Van Dooren, Y. Nesterov, Narrow scope for resolution-limit-free community detection, *Phys. Rev. E* **84,** 1 (2011) 016114. ⇒204
[34] CMS collaboration, A Cambridge-Aachen (C-A) based jet algorithm for boosted top-jet tagging. *CMS PAS JME-09-001*, 2009 ⇒199

Received: November 6, 2017 • Revised: December 6, 2017

Acta Universitatis Sapientiae, Informatica is covered by the following services:

ACM Digital Library
AMS Digital Mathematics Registry
Baidu Scholar
Cabell's Directory
Celdes
Clarivate Analytics - Emerging Sources Citation Index
Clarivate Analytics - Web of Science
CNKI Scholar (China National Knowledge Infrastructure)
CNPIEC
DOAJ
EBSCO (relevant databases)
EBSCO Discovery Service
Genamics JournalSeek
Google Scholar
Inspec
io-port.net
J-Gate
JournalTOCs
KESLI-NDSL (Korean National Discovery for Science Leaders)
Naviga (Softweco)
Primo Central (ExLibris)
ReadCube
ResearchGate
Sherpa/RoMEO
Summon (Serials Solutions/ProQuest)
TDOne (TDNet)
Ulrich's Periodicals Directory/ulrichsweb
WanFang Data
WorldCat (OCLC)
Zentralblatt für Mathematik

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.

Sapientia University

DE GRUYTER OPEN

Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Idea Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro