BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

HIGH-LEVEL OBJECT ORIENTED GENETIC PROGRAMMING IN LOGISTIC WAREHOUSE OPTIMIZATION

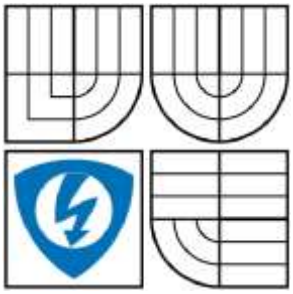
DOCTORAL THESIS
DIZERTAČNÍ PRÁCE

AUTHOR JAN KARÁSEK
AUTOR PRÁCE

BRNO 2014

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

HIGH-LEVEL OBJECT ORIENTED GENETIC PROGRAMMING IN LOGISTIC WAREHOUSE OPTIMIZATION

VYSOKOÚROVŇOVÉ OBJEKTOVĚ ORIENTOVANÉ GENETICKÉ PROGRAMOVÁNÍ PRO OPTIMALIZACI LOGISTICKÝCH SKLADŮ

DOCTORAL THESIS

DIZERTAČNÍ PRÁCE

AUTHOR — Ing. JAN KARÁSEK

AUTOR PRÁCE

SUPERVISOR — Ing. RADIM BURGET, Ph.D.

VEDOUCÍ PRÁCE

BRNO 2014

ABSTRACT

This work is focused on the work-flow optimization in logistic warehouses and distribution centers. The main aim is to optimize process planning, scheduling, and dispatching. The problem is quite accented in recent years. The problem is of NP hard class of problems and where is very computationally demanding to find an optimal solution. The main motivation for solving this problem is to fill the gap between the new optimization methods developed by researchers in academic world and the methods used in business world. The core of the optimization algorithm is built on the genetic programming driven by the context-free grammar. The main contribution of the thesis is a) to propose a new optimization algorithm which respects the makespan, the utilization, and the congestions of aisles which may occur, b) to analyze historical operational data from warehouse and to develop the set of benchmarks which could serve as the reference baseline results for further research, and c) to try outperform the baseline results set by the skilled and trained operational manager of the one of the biggest warehouses in the middle Europe.

KEYWORDS

Artificial Intelligence, Evolutionary Algorithms, Genetic Programming, Logistics, Optimization Techniques, Warehouse Management Systems

ABSTRAKT

Disertační práce je zaměřena na optimalizaci průběhu pracovních operací v logistických skladech a distribučních centrech. Hlavním cílem je optimalizovat procesy plánování, rozvrhování a odbavování. Jelikož jde o problém patřící do třídy složitosti NP-težký, je výpočetně velmi náročné nalézt optimální řešení. Motivací pro řešení této práce je vyplnění pomyslné mezery mezi metodami zkoumanými na vědecké a akademické půdě a metodami používanými v produkčních komerčních prostředích. Jádro optimalizačního algoritmu je založeno na základě genetického programování řízeného bezkontextovou gramatikou. Hlavním přínosem této práce je a) navrhnout nový optimalizační algoritmus, který respektuje následující optimalizační podmínky: celkový čas zpracování, využití zdrojů, a zahlcení skladových uliček, které může nastat během zpracování úkolů, b) analyzovat historická data z provozu skladu a vyvinout sadu testovacích příkladů, které mohou sloužit jako referenční výsledky pro další výzkum, a dále c) pokusit se předčit stanovené referenční výsledky dosažené kvalifikovaným a trénovaným operačním manažerem jednoho z největších skladů ve střední Evropě.

KLÍČOVÁ SLOVA

Evoluční algoritmy, Genetické programování, Logistika, Optimalizační techniky, Systémy řízeného skladu, Umělá inteligence

KARÁSEK, Jan *High-Level Object Oriented Genetic Programming in Logistic Warehouse Optimization*: doctoral thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2014. 182 p. Supervised by Ing. Radim Burget, Ph.D

DECLARATION

I declare that I have written my doctoral thesis on the theme of "High-Level Object Oriented Genetic Programming in Logistic Warehouse Optimization" independently, under the guidance of the doctoral thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the doctoral thesis I furthermore declare that, as regards the creation of this doctoral thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Sb., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No 40/2009 Sb., Section 2, Head VI, Part 4.

Brno ....

(author's signature)

ACKNOWLEDGEMENT

I am deeply grateful to my supervisor Ing. Radim Burget, Ph.D. for the guidance and professional support during the work on this thesis.

I would like to express gratitude to my parents, grand parents, to my dearest girlfriend and to others for their patience, understanding and their infinite support.

Brno ....

(author's signature)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

ACKNOWLEDGEMENT

Výzkum popsaný v této doktorské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
(author's signature)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Optimization, in general, is a process of finding the best feasible solution to a problem. An optimization problem can be viewed as a task, the goal of which is to configure a set of given parameters to reach an optimal solution of the given problem and, simultaneously, meet the predefined criteria. The global optimization [1], [2] is a process where such solution is required with the condition that no better solution exists. Solutions are termed bad and good in terms of an objective to which they are optimized. The field of optimization has grown rapidly in recent decades.

Optimization problems can be classified into two classes according to the time of optimization. The first class represents the *Online Optimization Problems*. These problems have to be solved quickly and in a time interval between milliseconds and a few minutes. The second class represents the *Offline Optimization Problems*. These problems are not time dependent. The user is willing to wait up to hours, days or weeks for a result. Such problems are not carried out so often. Optimization problems are solved by two types of algorithms: *dedicated algorithms* and *optimization algorithms*. Dedicated algorithms are specialized to solve, and exactly solve, a given class of problems in the shortest possible time. These types of algorithms are mostly deterministic and solve a well-known and structured problem. When the problem is too specific or too complex, optimization algorithms are used. The optimization algorithm often needs only a structure of desired solution and a function for a candidate solution quality measure. The optimization algorithms with this information on the input are able to find, or approximately find, a solution. The solution does not usually reach the quality of the solution given by a dedicated algorithm and the process of finding solution is usually slower. The optimization algorithms can be further divided into *deterministic approaches* and *stochastic approaches*.

Deterministic approaches, so called methods of mathematical programming, are for example linear and nonlinear programming, integer programming, dynamic programming and many other mathematical optimization techniques. Deterministic approaches in each execution step have only one way to proceed. If there is no way, the algorithm is terminated. The algorithm for the same input data produces always the same results. These techniques ensure optimality, but they become very complex and unmanageable when the number of parameters exceeds ten.

Stochastic approaches, so called probabilistic, approximation or randomized algorithms, come in useful when the relation between candidate solutions and a fitness function is too complicated and not much obvious. A stochastic algorithm applies at least one instruction based on a randomized action. Generally, stochastic algorithms are not so efficient as deterministic, but when a deterministic algorithm is not applicable, a randomized solution might be advantageous.

1.1 Research Context

The proposed research is focused on the problem of process optimization that can be applied in various fields of industry and anywhere else where it is possible to define activities as standardized processes (hereinafter referred to as *jobs*) consisting of further indivisible operations (hereinafter referred to as *tasks*). The validation of the proposed optimization algorithm will be done in the area of logistics. It is clearly the Online Combinatorial Optimization problem with many parameters. This area was chosen because there is a possibility to consult the work with a qualified expert in this area and the real operational data from the logistic warehouse are available. These facts should ensure the validity and reliability of the proposed work.

The jobs are spread among employees in a common warehouse in a very simple way. The manager has a list of all jobs which is continuously growing according to the new jobs that arrived, e.g. to unload the cargo from an arriving lorry and store it in the warehouse. The operational manager prepares the list of jobs for each employee. The list of jobs assigned to an employee consists of jobs representing picking of one homogeneous pallet or many heterogeneous products. When an employee finishes the list of jobs, a new list of jobs is given to him by operational manager. It is also possible to give to an employee a few lists of jobs in one time, but what will be fulfilled first and in which order is mostly on the employee. These practices are termed as generation one in process planning and scheduling.

A more sophisticated scheduling of jobs among the employees is done as follows. Each employee in the logistic warehouse is equipped with a bar-code reader or any other kind of smart embedded equipment able to do the work of the bar-code reader and more. This equipment is used not only for automatic identification of commodities, but the operational manager also sends the jobs which have to be fulfilled to this equipment. The manager can prioritize the jobs, add new jobs, completely remove the jobs from employee's list, or move the jobs to another employee. All of these functions are done dynamically, which is a very flexible approach to operational management. The operational manager is able to manage the planning and scheduling more precisely also thanks to Warehouse Management System (WMS). The WMS is able to monitor the performance of employees, their workload and many other parameters regarding employees and their jobs. The WMS is also able to monitor jobs, their origin, places for storing and track the commodity which is the objective of the job. The WMS also monitors the equipment and the most important thing is that it is able to communicate with a company Enterprise Resource Planning (ERP) system and change the data bidirectionally. These practices are termed as generation two in process planning and scheduling, and have become the most widespread solutions in warehouses and distribution centers in recent years.

Of course, there are also fully automated logistic warehouses supported by robots, automated hoists, automated guided vehicles, automated storage and retrieval systems and many other equipment, and the benefits of such solution of warehouse are indisputable, e.g. reductions in manpower and labor costs, fork-lift equipment and its maintenance, which implies that the work efficiency is improved. But, on the other hand, such a warehouse requires high capital investment, there is a problem with low tolerance to discrepancies due to mechanization, steep cost of downtime – the warehouse operation comes to a complete halt, reduced flexibility of the warehouse, and much higher maintenance costs. So, logistic companies are not willing in most cases to implement such a solution.

The proposed doctoral thesis is trying to fill the gap between the planning and scheduling processes of the generation two warehouses and the fully automated warehouses, since the key representatives of the logistics and warehousing industry still do not use fully automated scheduling of processes. This task will be done by automation of the system/software part of the scheduling process. The thesis is, of course, also focused on the connection of the latest scientific results in scheduling optimization with the demands of the companies in logistics and warehousing industry. The implementation of such optimization method could rapidly increase the productivity and competitiveness of the companies. Besides, thesis is a part of the project reg. no. FR-TI1/444 "Research and Development of the System for Manufacturing Optimization" which was led by Prof. Ing. Zdeněk Smékal, CSc.

1.2 Research Question

The problem statement, or research question, addressed in this thesis is following: *Is it possible to combine the well-known combinatorial problems solving techniques with the Genetic Programming driven by the Context-free Grammar and solve the logistic warehouse work-flow optimization problem in the way that the current state-of-the-art of the qualified operational manager will be outperformed?.*

1.3 Research Motivation

Although the problem of warehouse optimization is to some degree similar to the well-known scheduling problems (Job-Shop Scheduling Problem (JSS), Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), and other derived combinatorial optimization problems), this paper defines the problem from a different point of view. The optimization task here is focused in particular on the warehouse processes (work-flow) optimization where the JSS, TSP, and VRP overlap.

There are two main motivational factors for this work. The first factor is a significant demand from the industry to solve the problem of work-flow scheduling by an automated method and to help the operational manager with decision making, or even replace the operational manager with an automated optimization algorithm. The second factor is to bring the latest and most effective concepts of optimization in scheduling from the scientific world into the world of logistic industry.

The scheduling is still performed by the operational manager with simple graphical aids (Gantt's charts) in many warehouses. More developed warehouses are using some kind of WMS and the smart embedded equipment. Simple dispatching rules are often used for prompt problem solving. The scheduling in such warehouses often turns to chaotic environment where the completion times cannot be predicted.

The warehouses, commonly used for storing or buffering commodities between the point of origin and the point of consumption, are amongst the most important parts of the logistic chain. The most basic activities in the logistic environment are receiving, transfer and putting away, storing, order picking, cross-docking, and shipping. According to Tompkins et al. [3] the time spent on each of these *operations* can be divided into the following time segments: traveling (50 %), searching (20 %), picking (15 %), setup (10 %), and other unpredictable circumstances (5 %). As it can be seen, traveling, searching, and picking take the most significant part of time and provide a potential best place for optimization, which will be investigated.

1.4 Research Challenges

The most obvious part to optimize from the previous section is the time spent on traveling, searching and picking. This optimization was to this point done by a skilled operational manager and if he was using some optimization techniques, it was just weighting of parameters, which made together a fitness function of desired solution. The main weakness of this approach is the human factor. The operational manager is in stress, most frequently at the pre-Christmas time, and the planning and scheduling of warehouse processes is failing and the whole buffer of jobs is allocated in a hurry and the performance of employees goes rapidly down. Therefore, the automated scheduling method, independent on a human consciousness, is needed.

The first challenge is to design an automated optimization algorithm and support the decision making of operational manager. If the results of the optimization algorithm will surpass the results of the operational manager, the challenge will be to deploy the algorithm as a stand-alone system without any supervision of human.

The second challenge is to develop an optimization algorithm which will be as much flexible as possible. The optimization algorithm should be a general problem

solving algorithm with a possibility to define new criteria how to influence the convergence to the solution. In other words, the optimization algorithm should be easily extended by new innovative ideas, new genetic operators, restrictive conditions from the warehousing knowledge domain.

There are already optimization algorithms for optimization of warehouse processes, mostly focused on picking strategies. They are also focused on searching in form of categorization of commodity by bar-code labeling. Unfortunately, there is a lack in optimization of transportation path. The third challenge will be to design an optimization algorithm for automated collision avoidance.

All of these, together, should guarantee a significant increase in performance and warehouse productivity. Since the fourth challenge is to reach better performance and productivity of warehouse, it is necessary to define methods for performance measurement. So, benchmarking methods and metrics have to be stated.

1.5 Contribution

The main contributions of the proposed doctoral thesis are following:

1. A comprehensive literature review of the warehouse optimization connected to the scheduling problems and the vehicle routing problem was written in chapter 2. The basics found in the literature helped to define the hypothesis and goals (chapter 3), and to extend the mathematical model (chapter 4 for the warehouse work-flow optimization with the help of employees' performance which positively influences the processing time of the scheduling process.
2. The new, extensible, flexible, and multi-platform Evolutionary Framework with the computational core based on Genetic Programming (GP) driven by the Context-Free Grammar (CFG) was developed, implemented, and validated (chapter 6). The framework is based on the existing Grammar Guided Genetic Programming (GGGP) approach presented also in this chapter.
3. The new algorithm for the warehouse work-flow optimization problem (chapter 7) based on the proposed framework (chapter 6) was developed and supported by several new genetic operators, which give the possibility of cooperative job processing. The fitness function can respect multiple criteria, such as the time of processing of the whole job buffer (the makespan), balanced workload of employees, and the number of collisions of trucks (chapter 5).
4. The problem of the warehouse work-flow optimization was described along with the motivation to solve the problem. The set of benchmark tests was created as well as the evaluation process. These together give the reference baseline of the results for the optimization (chapter 8). The results reached by the optimization algorithm are presented in chapter 9.

1.6 Outline

The rest of the thesis is structured as follows. Chapter 2 describes the theoretical background of the proposed doctoral thesis. The main optimization points of the warehouse management and the logistic chain are described (see section 2.1) as well as the most related optimization problems, such as the scheduling problems (see section 2.2) and routing problems (see section 2.3). The second chapter also deals with one of the most frequently used meta-heuristic optimization method, GP, which was used as the core of optimization algorithm in this thesis (see section 2.4).

Chapter 3 describes the hypothesis, which will be the main subject of investigation in this thesis, and the partial goals, which should directly support and confirm the determined hypothesis. Chapter 4 describes the basic mathematical model for scheduling and the mathematical model extended for the purposes of this thesis.

Chapter 5 deals with the design of collision prediction which should represent an approach of how to avoid congestions, blocking, and possible financial losses by predicting potential collisions of trucks in the warehouse. Chapter 6 describes the design of the Evolutionary Framework and the design of the Genetic Programming algorithm driven by the Context-free Grammar which is used as a basic building block for the optimization algorithm proposed in this thesis. Chapter 7 describes how all the expert knowledge and algorithms implemented were put together into one algorithm for the warehouse process optimization. Chapter 8 describes the benchmark definitions and the test sets. Chapter 9 describes the results of benchmarking on both the real data set and the synthetic data set. And the last, but not least, chapter 10 concludes the paper, discusses the results, and proposes some ideas for future work.

2 THEORETICAL BACKGROUND

This chapter is divided into four sections which give a comprehensive survey of theoretical background of the proposed doctoral thesis. Section 2.1 gives an overview of general warehouse optimization problems with focus on applications in real-world environments. Section 2.2 describes the general scheduling problem with focus on shop scheduling, which is the most related to the solution of the problem proposed in this work. Furthermore, methods how the scheduling problems are currently being solved are discussed in this section. Section 2.3 gives an overview of the routing problems which are used to extend the scheduling models in order to solve the problem of logistic warehouse process planning and scheduling connected to dispatching and routing of vehicles. Section 2.4 gives the state-of-the-art in GP which is a promising method that could bring the new possibilities how to connect the mentioned optimization problems together and bring the optimal or near-optimal results.

Optimization problems can also be divided according to the number of optimization criteria. The common classes of problems are stated as single-objective, multi-objective, and constraint optimization problems. The optimization problem can also be a mix of these classes. The advantage is that there are approaches to transform multi-objective problems into single-objective ones, and also to reduce the constraint optimization problems to unconstrained optimization ones. Many new procedures, algorithms, computational methods and techniques of optimization were invented to solve various types of problems. Basically, there are two general types of problems: combinatorial problems and numerical problems.

Combinatorial problems [2], [4] are defined over a finite (or numerable infinite) discrete problem space $\mathbb{X}$ and the candidate solutions structure can be expressed as (1) elements from finite sets, as (2) a finite sequence or permutation of elements x_i chosen from finite sets, i.e., $x \in X \rightarrow x = x_1, x_2, \ldots$, as (3) sets of elements, i.e., $x \in X \rightarrow x = x_1, x_2, \ldots$, as (4) tree or graph structures with node or edge properties stemming from any of the above types, or (5) any form of nesting, combination, partitions, or subsets of the above. Combinatorial problems are e.g. JSS, TSP, VRP, graph coloring, graph partitioning, bin packing, and many others. Algorithms suitable for this group of problems are Genetic Algorithm (GA), GP, Simulated Annealing (SA), Tabu Search (TS), and Extremal Optimization.

Numerical problems [4], [5] defined over $\mathbb{R}^n$ or $\mathbb{C}^n$ (real or complex vectors which are subspaces of numerical space) are also called continuous optimization problems. The typical representative problems of this set are e.g. classification and data mining tasks, and engineering design optimization tasks. These problems can be efficiently solved by Evolution Strategies (ES), Evolutionary Programming (EP), Particle Swarm Optimization (PSO), Differential Evolution, and many others.

2.1 An Overview of Warehouse Optimization

Modern logistic warehouses and distributions centers are designed on the basis of hundreds optimization studies. In consequence of that, WMSs become important and more complex and users find it hard to manage. The software market offers a large variety of solutions with different system requirements and possibilities, and to choose the suitable system for every company is not quite an easy task, because it is influenced by many aspects which must be considered, and one of these aspects are optimization methods based on automated processes.

The WMSs which drive logistic warehouses and distribution centers are core elements of the material and goods flow in a logistic chain and they will be subjected to further investigation in the following text related to optimization of warehousing.

According to [6] the activities of the warehousing optimization can be divided into three groups. First, the basic technical structure of warehouse. Second, the operational and organizational framework, which is in special attention in this work. Third, the coordinating and controlling systems for warehouse operations.

2.1.1 Optimization of Technical Structure

The basic technical structure involves e.g. the layout design of the warehouse or whole distribution center, the choice and dimensioning of conveyors and warehouse equipment, the design of the physical interfaces to neighboring systems and others.

The layout design of the warehouse [7] is a key component of further optimization tasks and has a significant impact on order-picking and traveling distances in the warehouse. In [8] it was found out that the layout design has more than 60 % effect on the total travel distance, and three basic types of warehouse layout were presented. In [9] and [10] the application of parallel cross aisles in the warehouse was presented, and it was considered a significant improvement. The layout is usually of rectangular shape and based on pallet manipulation [11]. According to [12] and [13] there are a few factors to be considered in the layout design, such as: number of blocks; length, width, and number of picking aisles; number and shape of cross aisles if they are present; number of levels in the rack; and position of input and output gates in the warehouse. A new *Flying-V* and *Fishbone* design of cross aisles, which offers a 10 % – 20 % reduction of traveling distance were presented in [14]. An analysis of dual-commands was introduced in [15] and *Fishbone* design for dual-commands in [16]. In dual-commands environment the worker loads the goods in pickup and deposit location and travels to storage location and then travels to the second location from which he picks goods and returns back. More developed *Flying-V* design of cross aisles and *Inverted-V* design of cross aisles was proposed in [17].

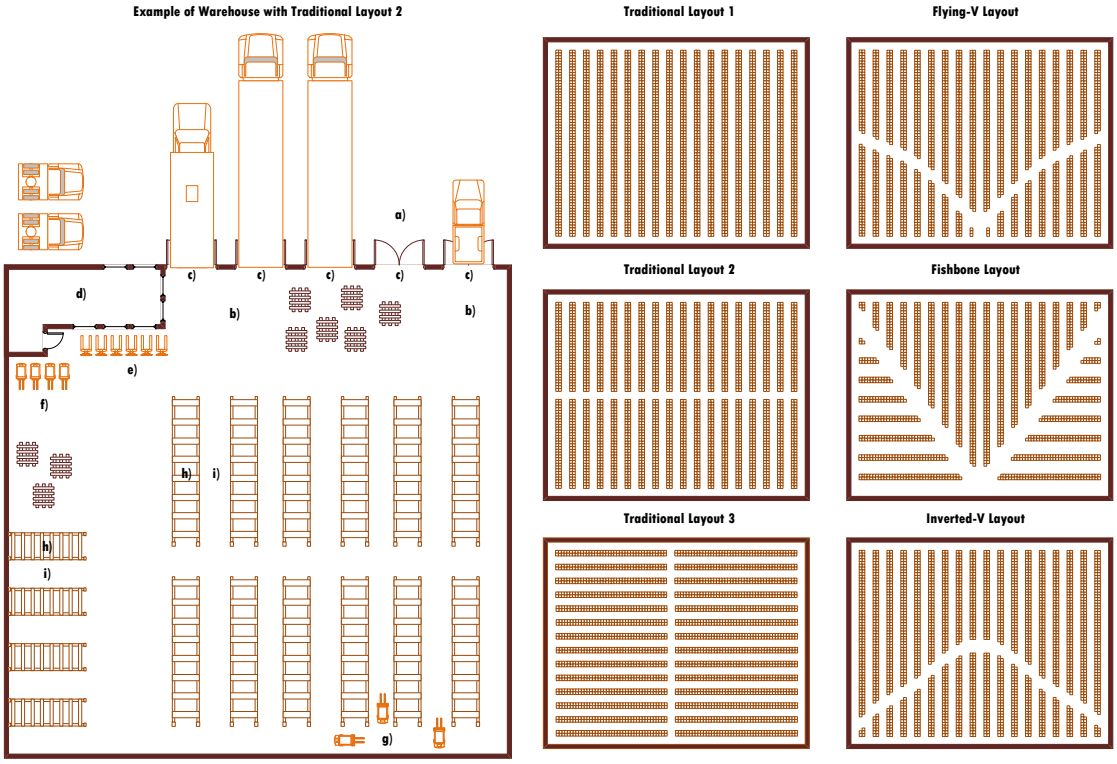


Fig. 2.1: An example of the warehouse environment and the aisles designs

This brings another 3 % improvement of traveling distances. The warehouse layout is also connected with the aisle design, which is discussed in [18] and [19]. The layout is mostly narrow-aisle-like, which increases space utilization with minimal costs, but it can lead to higher operational costs and more congestions among workers.

There are many types of warehouse equipment, especially equipment which should reduce labor cost and increase its utilization. Common storage models cover pallet racks, cartoon flow racks for high-volume picking, and shelving for lower-volume picking. All the equipment is standardized according to the dimensions, but the standardization is mostly only for a specific continent. While industrial trucks for pallet manipulation are demanded in all types of warehouses, the conveyors, cranes, and other positioning equipment is not used everywhere. Conveyors divide the warehouse into zones and move material over a fixed path. Conveyors also restrict the movement of workers and save their energy. With deployment of conveyors also the sortation system is quite often installed for merging, identifying, inducting, and separating products. Sortation is mostly based on some scanning technology of bar-codes, RFID chips, magnetic strips or machine vision. The system works on a few common principles, e.g. a *push sorter* pushes a passing carton to an alternative path from main conveyor, a *tilt-tray sorter* works on the principle of tilting a tray and the object slides into the collecting bin, and others. Cranes are used to

move materials over variable paths in a restricted area, e.g. jib crane, bridge crane, gantry crane, and stacker crane. Positioning equipment, e.g. hoists, balancers, and other manipulators, is used to handle material at a single location. The automation of warehousing and manufacturing is often covered by systems such as carousels, A-frames, and Automated Storage and Retrieval System (ASRS).

Carousel is a shelve rotating in the circle. Instead of picker traveling, the storage location is moving. The simple rotation pattern on how to quickly find the shortest way to pick the order was introduced in [20] and [21]. The large orders in carousel environment have been studied in [22]. The carousels with multiple order-picking have been studied in [23]. Optimal storage locations have been investigated in [24]. *A-frame* is an automated dispensing machine dropping items onto a conveyor. A-frame is used when a product is picked in very high volumes, the labor is expensive and is used only to refill A-frames. The in-aisle cranes, so called ASRSs, replace the humans with trucks by placing simple robotic devices within each aisle moving in horizontal and vertical direction to a full extent of aisle. The design and performance of such models as well as travel time models have been investigated extensively, e.g. in [25], [26], [27], and [28]. Despite of all these inventions, the typical model of warehouse with pickers and various models of trucks are still quite popular.

2.1.2 Optimization of Operational Structure

The operational and organizational framework combine different aspects from many areas, e.g. business management, inventory management, organization management, transportation management and many other areas of management. There are two basic slotting strategies (or storing assignment policies [29]): *random* and *dedicated*. While random strategy allows to store a pallet to an arbitrary empty location with the same probability [30] or to the closest empty location [12], the dedicated strategy allows to store a pallet only to specified locations. The storage locations are often organized somehow, e.g. *class-based storage*, where the goods are clustered according to the frequency how often they appear in orders. This policy assigns the most frequently requested goods to the best (closest) locations from input/output gates. Another possibility is to use *family grouping*, where the goods are clustered according to the relations or similarities between products or orders [31], [32].

Single order-picking is a strategy where the pickers pick only one order at a time and it is one of the most used picking policies. Stock Keeping Unit (SKU) is tightly related to order-picking. SKU represents the smallest physical unit of a product with which the cooperation manipulates, e.g. a box, some kind of case or carton consisting of inner packs and individual pieces of product, but it can also be only a homogeneous pallet in huge distribution centers.

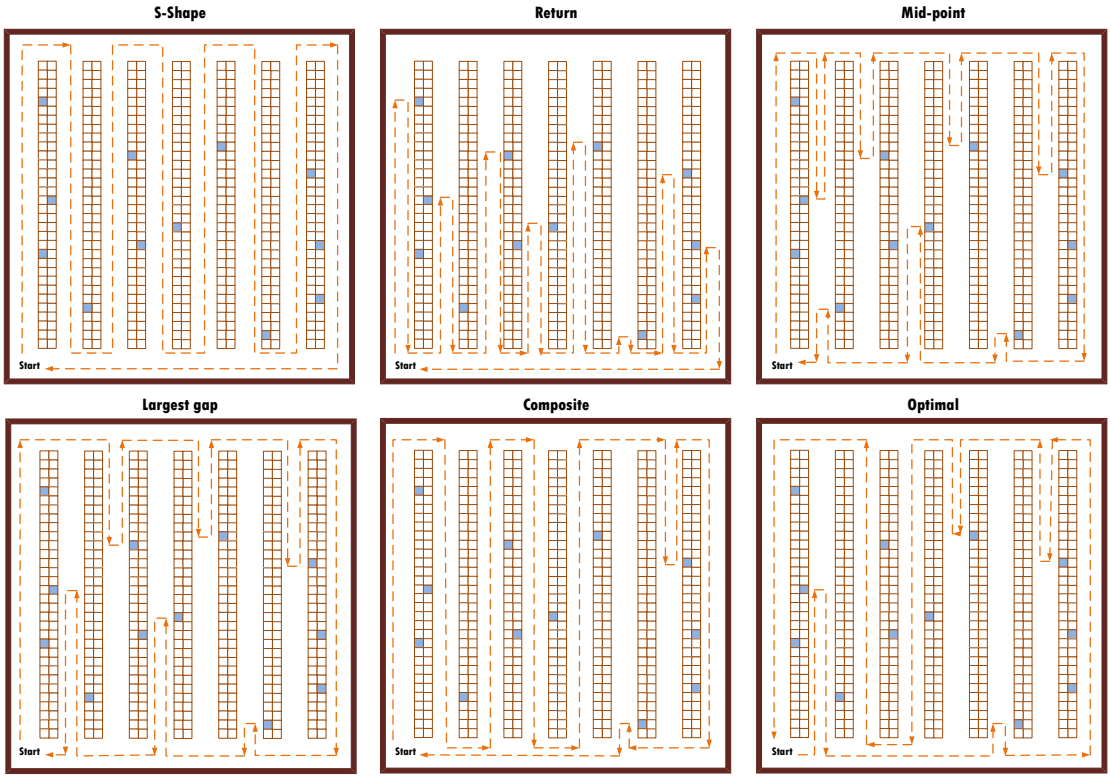


Fig. 2.2: The typical routing methods used in warehouse environments.

The routing policies should ensure an optimal travel path through the warehouse for order-picking. One of the first algorithms for optimal order-picking path design was introduced in [33]. Since the algorithm can be applied only to conventional warehouses, the problem is mostly solved by heuristic methods. The common routing methods, described in [10] are: *S-shape*, *Return policy*, *Mid-point strategy*, *Largest gap strategy*, *Composite heuristic*, *Optimal routing*. All the methods were primarily developed for single-block warehouses. Modified methods for multiple-block warehouses were proposed in [9].

If the order is small and is far from exceeding the picking capacity, it is possible to pick more orders in a single order-picking tour. This is known in the literature as *order batching* or simply *batching*. Since this is a job with sub-tasks (picking tour with several orders) it is considered as an NP-hard problem. It was proven in [34] that *batching* has a significant impact on the performance of order-picking. Therefore, researchers pay attention to the problem of *batching* and the heuristic methods are still under investigation [35], [36]. There is also possible to divide an order-picking process into zones. Goods belonging to the same product group are stored in the same zone. In comparison with batching, *zoning* does not have a significant impact on the performance of the order-picking system [37]. The advantage

of zoning lies in reducing the congestion in the aisles and when the goods are really in one small area, the traveling is also reduced. The main disadvantage is the consolidation of order when it is completed from more pickers from different zones.

2.1.3 Optimization of Warehouse Management

The coordinating and controlling systems are of special importance. The WMSs are used to control the warehouse and optimize all typical warehousing operations (see section 2.1.4), to know every detail about goods and the actual storage location all the time, the utilization of workforce, orders, and they also orchestrate the flow of labor, machines, and goods. Such systems have many interfaces to adjacent systems in the company, e.g. information and merchandizing systems, production and enterprise resource planning systems, material flow and warehouse controlling systems related to business-to-business or business-to-consumer applications.

Why is all the optimization being applied? Everything is basically given by customer demands. The main reasons to optimize are to increase the performance of the company regarding the demand-driven production (pull system), to ensure the productivity (based on just-in-time delivery) and minimize the stocks along the supply chain, provide additional services, and reduce the transportation cost.

2.1.4 Typical Warehousing Operations

The basic processes in the warehouse are receiving, storing, putting-away, picking/retrieving and shipping goods. The shipping operation can also consist of many sub-tasks such as consolidation of goods if the batching, grouping or zoning is applied, checking the order according to its completeness, packing and, of course, shipping itself. The literature also mentions cross-docking as a special warehouse operation. The cross-docking is described at the end of this sub-section.

Receiving is the first operation in the warehouse. This process starts by notification of the arrival of goods. Then the process of unloading, counting, identifying, quality control, and goods acceptance begins (which is together known as incoming inspection), according to the company rules. If the goods are accepted the receipt is issued. The acceptance depends on the delivery status – the delivery date, the quality of delivery, the planned schedule which should also minimize a truck waiting time. The product is then accepted, marked e.g. by bar-code and registered in the information system, and staged for put-away. The receiving process takes about 10 % of operating costs [11]. The paper [38] discusses the role of goods receiving and shipping in warehouse environments. A formal notation of schedules is proposed and the specific analytical examples are shown in this paper.

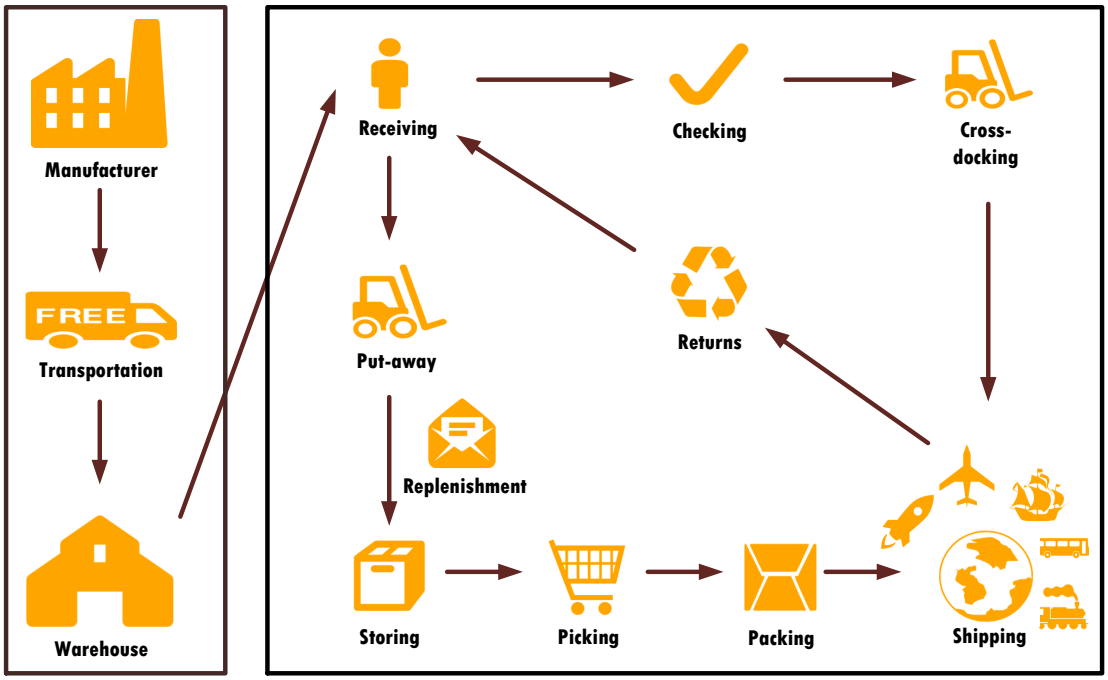


Fig. 2.3: An example of the logistic flow from the manufacturer to the warehouse.

Storing operations consist of distribution of goods to storage areas (transportation to a storage place or cross-docking, which is transportation directly to the shipping department), identification (if it was not done during acceptance), assignment of the storage bin and putting-away which is a simple determination of storage bin concerning the physical dimensions and weight of goods, storage monitoring is also a part of management systems – to know which goods are available and where [6]. **Putting-away** is a process which requires strictly determined storage location. This is very important, because the information system has to be aware all the time what storage locations are available, what is the location of a specific type of goods and where a particular pallet is stored. This information is also used for an efficient design of pick-list. This process requires about 15 % [11] of operating costs, because this covers lot of transfers from the gate to a storage place.

Picking (also called **Retrieval**) is a process which covers a lot of issues. Firstly, the pick lists are given to the employee. The picking takes about 55 % of warehouse operating costs [11] (according to [12] it is 50 % – 70 %) and consist of: traveling (55 %), searching (15 %), extracting (10 %), and paperwork (20 %). Picking can be of two types, homogeneous and heterogeneous. Homogeneous picking is quite simple, the picker operates simply with a whole pallet. In heterogeneous picking the picker is told where and what to pick, in what quantity and units. Due to customer's needs, the heterogeneous picking is logically more frequent. The disadvantage of heterogeneous picking is that the smaller unit means the higher cost. The pick-list

is still quite often a sheet of paper, but in some warehouses the pickers use smart embedded devices such as bar-code readers, personal digital assistants etc. The advantage of using such equipment is reviewed in [39], [40].

The planning of picking process is based on orders and supported by picker routing methods. Basic routing methods are described in [10]. Picking of goods can be done by many ways. The special case of picking is an order-picking which is a consolidation of a customized quantity of one or more articles related to a specific order. Sharing of order-picking is also quite often way how to pick. Sharing of an order is very related to batching, grouping, and zoning. The order-picking was also designed by an algorithm based on TSP heuristic introduced in [41] which performs better for multiple picking than routing methods. It was discovered in [42] that appropriate sequencing of picking is one of the crucial factors to achieve high efficiency of picking. Since traveling is the most time-consuming part of the process, the scientists paid attention mostly to this part of the problem.

The travel time is an increasing function of the travel distance, which was investigated in many papers and considered as one of the primary optimization conditions. Three analytical models of expected travel distance for return policy, traversal policy, and midpoint policy were developed in [43]. To allay blocking and/or congestion, the order-picking strategies can be used, or the layout of the warehouse can be adjusted in the meaning of wide-aisles, or zoning can be applied [44]. The congestion has been also investigated in [12] by waiting time of a picker queuing to enter the warehouse. Analytical and simulation models of order-picking systems to discover the system behavior with different activity levels were developed in [18]. The result was that the congestion among workers can be a significant issue if the space is highly utilized. An analytical approach for the expected system throughput time approximation was provided in [36]. The relationship between pick density and throughput, which has demonstrated the significance of blocking, was determined in [45] and [46]. Inasmuch as the models of picking are mostly investigated only as single-picker operations, and are consequently suitable to evaluate order-picking efficiency by travel distance, an aisle congestion never takes place in such models. In real-world situations the congestion is a normal everyday situation in systems with multiple order-picking and dozens of workers. The throughput analysis for order-picking with multiple pickers and aisle congestion is investigated in [47] and [48]. Heuristic methods were investigated in [49] and it was proved that the storage assignment policies in a multiple picker warehouse environment are outperformed by the proposed heuristics policies in this paper.

The batching in the narrow-aisle order-picking system with picker blocking consideration was investigated in [50]. The paper propose strategies to control picker blocking with 5 % – 15 % reduction in the total retrieval time. Ant Colony Op-

timization (ACO) routing algorithm for two order pickers with consideration of congestions was proposed in [51]. The paper analyzes the warehouse layout and its impact on order-picking system performance and proves a good performance in dealing with the congestions if two pickers are used simultaneously. Two new batch construction heuristics called K-means Batching and Self-Organization Map Batching, which minimize the total travel distance and average picking vehicle utility, were proposed in [52].

Also the online version of the multiple picking agents for warehouse management has been studied in [53]. The rescheduling of buffer of order to which the new orders arrive randomly while old orders are being picked was investigated and two real time algorithms were proposed. The solution performs good when dynamism is low or moderate, when it is high the solution tends to fail. The dynamic order-picking and cost reduction generated by optimal policies is discussed in [54]. In this research the Markov Decision Process based heuristics, which is compared to some naive heuristics, has been used and the improvement in the range of 7 % – 99 % is depicted. Another heuristic approach for online batching based on offline batching is introduced in [55] and [56]. The algorithms are evaluated in a series of experiments and it is shown that the choice of an appropriate batching method can lead to significant minimization of a maximum completion time. A*-algorithm for routing and SA-algorithm for batching were proposed in [57]. For batches of tree customers, the proposed algorithm produces results with an error of less than 1.2 % compared to the optimal solution.

Shipping ensures that the packed consignment is provided by transport destinations, assigned to the truck and optimally loaded on the truck. The shipping process is ensured by shipping department which can also secure following operations, such as: consolidation, checking, and packing. **Consolidation** of an order is a process of completion a customer's order in case that it was picked by more than one picker. The paper [58] proposed the design and operation of an order-consolidation warehouse. The paper proposes a simulation model and shows its application. When order is consolidated, the process of checking follows. **Checking** of an order is a process that checks if the order is complete and accurate. **Packing** ensures that the picked and consolidated goods, also checked for completeness of an order, are packed for transportation and given to the shipping department. Packing can also be ensured by an autonomous packing department in the warehouse, then the consolidation and checking is usually part of this department.

Cross-docking is a process which minimizes the storage and order-picking time while the receiving and shipping operations are still allowed to the full scope. The basic idea is to transfer goods directly from incoming to outgoing departments without any other warehouse operations in between. In [59] the problem was handled as

a VRP. A solution for multiple delivery centers by many sub-optimizations of single centers based on Neighborhood Search (NS) and TS algorithms was developed in [60]. The scheduling of trucks in the cross-docking system with five meta-heuristic algorithms (GA, SA, TS, Electromagnetism-like Algorithm, Variable NS) with respect to minimization of total operation time was described in [61]. The analytical models for pre-distribution cross-docking (on the side of manufacturing company) and post-distribution cross-docking (on the side of warehousing company) proposed in [62] were compared with a traditional distribution center system. Analytical results showed a pre-distribution cross-docking as a preferred solution for centers with a shorter supply lead time and lower uncertainty of demand, but in general the preference depends on the business environment [62].

2.1.5 Discussion

The optimal operation of a warehouse is achieved when each customer is satisfied completely according to his order, in due time and when all warehouse and logistic processes are done in the shortest possible time with minimal cost and optimal utilization of resources under dynamically changing conditions.

The literature presented in this section gives great ideas of warehousing optimization possibilities, but only some of them are really applied in real-world warehouses. The problem of warehouse layout lies mainly in the effective use of space so that the typical rectangular warehouses with narrow-aisles are most utilized. There is also a critical pressure on the effective utilization of equipment and labor and its minimal quantity in the warehouse, which can also significantly save the costs.

The dedicated assignment based on the frequency of manipulation with goods is broadly used, but some big and well-known companies, e.g. *Amazon*, use the chaotic assignment system and its seems to be a good solution as well. The routing methods supporting order-picking and picking itself has been investigated for single picking tours, but batching seems to be a standard for many companies. Moreover, the most of the scientific papers do not take into account the real conditions as the blocking and congestion are, but there are dozens of workers working simultaneously in real-world warehouses or multiple-block warehouses and the blocking, congestions or even collisions must be taken into account in everyday operation.

The result of this review of warehouse optimization is that the real-world conditions should be applied instead of their relaxation for the good of application. The work can be shared by many more employees than two, which was mostly investigated in the papers. The idea how to apply this optimization is to utilize the shop scheduling techniques combined with vehicle routing problems solving techniques.

The shop scheduling techniques can be employed when the work is scheduled in the warehouse, even when the work must by scheduled dynamically. The machines in the shop scheduling problem are represented in the warehouse by any equipment needed for each job, such as trucks driven by workers (fork-lift hand pallet truck, fork-lift low truck, fork-lift high truck), checking units (workers), packing units (workers with special equipment) and others. The operations in the warehouse, called jobs in scheduling, represent the single assignment given to the worker by operational manager, e.g. the employee has to unload the pallet from a lorry, go through the warehouse and store it in the shelf. The job is composed of sub-operations, called tasks. The task represents each single operation of job e.g. receiving, unloading, putting-away, moving and storing etc. The tasks can be done by several workers. So, the job is spread in few machines working in sequence in the language of shop scheduling problems. Transports, moving and routing of trucks in the warehouse could be inspired by Automated Guided Vehicles (AGV) techniques transformed from open space VRP techniques to the warehouse environment. The application of these methods could further reduce the blocking and congestions as well as collisions of trucks.

2.2 An Overview of Scheduling Problems

The increasing popularity of the Total Quality Management and the consequent on-time delivery of jobs become one of the crucial factors and a great demand for customer satisfaction. The process of scheduling is very important for achieving this goal. In fact, the planning is going hand in hand with scheduling and they are among the most important issues of operational management in manufacturing companies. The intractability of the scheduling problem makes it so much popular even if there are hundreds of papers published on this topic.

The difference between planning and scheduling is following. **Planning** is defined as the process of identifying all activities necessary to complete the project. The important questions in planning are *"What?"* and *"How?"* regarding the material, machines, employees, equipment, and due dates, or deadlines. **Scheduling** is defined as the process of determining the sequential order of jobs, assigning planned duration and determining the start and end of each job on a specific machine. The important questions in scheduling are *"Where?"* regarding the machines and equipment, and *"When?"* regarding the start and end of operation. The scheduling is quite a general process which can be considered in lots of environments, not only in manufacturing. The theoretical background described in this section deals mainly with machine scheduling models and focuses on the relevance of theory to the real-world application in logistics. This section should give an obvious picture that the scientific theory in scheduling has only a limited impact on real-world scheduling problems in logistics.

In terminology there is a distinction between three types of scheduling [63]. The first is **sequencing**, which is a permutation of n jobs or the order in which they are processed on a given machine. The second is **scheduling**, which corresponds to allocation of n jobs within m machines with more complicated settings in conformity with Tab. 2.7 and Tab. 2.8. The third are **scheduling policies**, which describe actions to do for any of the states the system may be in according to Tab. 2.6. Now, let us look at the three common types of schedules:

Definition 1 – Semi-Active Schedule – A feasible non-preemptive schedule is called semi-active if no job can be finished earlier without changing the order of processing on any machine.

Definition 2 – Active Schedule – A feasible non-preemptive schedule is called active if it is not possible to make schedule by changing the order of processing on the machines and have at least one job finished earlier and no job finished later.

Definition 3 – Non-Delay Schedule A feasible schedule is called non-delay if no machine is kept idle while an operation is waiting for processing.

2.2.1 Historical Perspective of Scheduling

The process of scheduling plays an important role in optimization area from the beginning of the previous century with the work of *Henry Gantt*. In spite of that the problem of scheduling is known for such a long time, the first publications were released in the early fifties of the previous century. At first, static models were investigated. Static model is a model where the basis of operations ordering does not change during scheduling, all operations can be sorted only once. This was followed by investigation of more advanced dynamic models, where the order of remaining operations changed, the operations had to be resorted in queue or rescheduled after every decision. In 1955 *Jackson's rule* [64] was proposed, which described how to schedule jobs according to non-decreasing due dates. *Smith's rule* [65] from 1956 schedules jobs according to non-increasing ratios w_i/p_i.

During the sixties, a significant progress in this area was done on formulations of the problems and deterministic algorithms. Mathematical programming methods for the solution of scheduling problems was studied, mainly integer programming, dynamic programming, and branch-and-bound algorithm [66]. *Richard Karp* was working on the complexity theory, he published a significant paper [67] which had a large impact on many researchers in the next decades.

In the seventies, the research in scheduling area was focused mainly on the complexity theory and hierarchy of scheduling problems. In 1979, *Ronald Lewis Graham et al.* [68] published the paper where Graham's notation was introduced for the description of shop scheduling problems. *Gittins index*, introduced in [69], is the largest value that is obtainable by dividing the total discounted expected reward over a given time period (by stopping time) by the discounted time itself. This was further studied by many researchers in stochastic scheduling [70]. In this decade, a shop scheduling problems started to be investigated extensively [71], [72].

In the eighties, expansion of the stochastic scheduling algorithms began. With the boom of personal computers, the scheduling problems were more and more applied to various fields of interest, such as computer science, operations research, industrial engineering, manufacturing, economics and others. The paper [73] introduced a very efficient algorithm for linear programming relaxation of Knapsack problem, which can also be considered as a scheduling problem. The bottleneck identification and the local re-optimization procedure are proposed in [74]. The importance of approximation methods for combinatorial problems and the linkage between operations research and artificial intelligence were suggested in [75].

With the nineties the scheduling theory started to play an important role also in management and services. Problems regarding the complexity studies of various scheduling problems, mainly *1-job-on-r-machines* were analyzed in [76]. The paper

briefly reviews some of the recent extensions of the scheduling theory and recent developments in local search techniques and their use in the scheduling practice. Other reviews of scheduling problems are present in [77], [78] and approximation methods, such as GA, SA, TS are presented in [79], [80], [81], respectively.

In the last years, from the year of 2000, the methods for solving one- and multiple-machine problems as well as shop scheduling problems are further developed with focus on meta-heuristic algorithms. The SA [82] and GA [83] algorithms were studied extensively also with the support and combination of local search techniques [84]. Furthermore, the multi-objective approaches were investigated with focus on flexible shop scheduling problems [85], [86]. In recent years, also many new algorithms were developed, such as Beam-ACO [87], Immune Algorithm Approach [88], Biogeography-Based Optimization (BBO) algorithm [89] and many others. Even though all of them are somehow connected to evolutionary computation techniques, it cannot be concluded that one significantly outperforms another.

2.2.2 Deterministic and Stochastic Models

This subsection describes deterministic and stochastic models. In deterministic models all sets of variables are uniquely determined by parameters and by sets of previous states of these variables. Hence, these models give the same output for the same input data and initial conditions. On the contrary, the stochastic models are based on a certain degree of randomness. Variables are not given deterministically by a unique definition, but by probability distribution. The common notation for both models is described in Tab. 2.1.

Tab. 2.1: The common scheduling notation.

n	The number of jobs.
m	The number of machines.
J_j	The job j; $j = 1, \ldots, n$.
M_i	The machine i; $i = 1, \ldots, m$.
w_j	The weight of j represents a priority factor, i.e. how the job is important in comparison to the other jobs in the job buffer.

In deterministic version of the scheduling problem all the jobs and machines are of a finite number. The description of deterministic model is presented in Tab. 2.2. It is described by the notation defined in [63], which is based on Graham's notation [68]. In the scope of stochastic models the notation has been changed a little bit. The description of stochastic model is presented in Tab. 2.3. Random variables are capitalized, but other symbols remain the same as in the deterministic models.

Tab. 2.2: Deterministic scheduling problem notation.

p_j	The processing time of job j, p_{ij} refers also to machine i. The i can be omitted if the processing time of job j is not relevant to the machine i or if the job j is only to be processed on one specific machine.
r_j	The release date or ready date, the earliest time when the job j is allowed to be processed, this attribute is not related to the machines.
d_j	The due date of job j represents the *committed completion date* of the job, after this date the job is penalized. It can also be referred to as a deadline when the due date *must* be met, deadline is denoted by $\bar{d}_j$.

Tab. 2.3: Stochastic scheduling problem notation.

X_j	The random processing time of the job j, X_{ij} refers also to machine i, The i can be omitted if the processing time of job j is not relevant to the machine i or if the job j is only to be processed on one specific machine. The expected value of the variable X_{ij} is denoted by $1/\lambda_{ij}$.
R_j	The random release date of job j.
D_j	The random due date of job j.

Graham's notation [63], [68] is a triplet $\alpha|\beta|\gamma$. The γ sign represents the objective to which the scheduling problem is optimized. The most common and basic objective which is used for quality measurement of the solution is the completion time and the lateness in its both variants (earliness/tardiness). For more basic objectives, see Tab. 2.4. The regular performance measures used in algorithms consist of these mentioned basic objectives, e.g. the makespan, which is the maximum completion time, or maximum of any other basic objective. The common variant of regular objective is also the weighted form of the objective, which represents a priority factor or importance of each job. For more regular objectives, see Tab. 2.5.

The typical dispatching priority rules, also called sequencing rules or basic heuristics stating the basic algorithms of dispatching, which should be considered in every scheduling algorithm, are described in Tab. 2.6. Dispatching rule is a rule that prioritizes all jobs waiting in a buffer for processing on a machine. Every single time when a machine has been freed, a dispatching rule inspects the buffer of waiting jobs and selects the job with the highest priority. The advantages of dispatching rules are very simple implementation, fast processing, a possibility of reasonably good solution in a relatively short time and optimality in special cases. Unfortunately not for all cases, which limits the dispatching rules in practice and that is why the dispatching rules can also find an unpredictably bad solution in some cases which are not considered as special for a concrete dispatching rule. The combination of rules is called a composite dispatching rule and can perform significantly better.

The β sign denotes details of processing characteristics and constraints (see Tab. 2.7 and Tab. 2.8); other entries in β are self explanatory, e.g. $p_j = p$ means that all processing times are equal, $d = 3s$ implies that all due dates are equal to 3 seconds etc. The α sign stands for a machine environment (see Tab. 2.9 and Tab. 2.10) which is described in more details in the following sections.

Tab. 2.4: The basic objective functions for the time measuring - γ.

C_j	The completion time for job J_j, the $\sum C_j$ is also referred to as a flow time. The C_{ij} refers to the completion time for J_j on machine M_i.
S_j	The starting time for job J_j, if it is less then $r_i j$ the job started earlier, if it is greater the job started late.
L_j	The lateness of job J_j is $L_j = C_j - d_j$, lateness is any deviation from the due date, a positive value means that the job is completed late, a negative value means that the job is completed earlier.
E_j	The earliness of job J_j is $E_j = max(d_j - C_j, 0)$, earliness represents a negative lateness, which means how much is the job completed earlier, otherwise the value is zero.
T_j	The tardiness of J_j is $T_j = max(C_j - d_j, 0)$, tardiness represents a positive lateness, which means how much is the job completed late.
U_j	The unit penalty of J_j is $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.

Tab. 2.5: The regular performance measures of scheduling problems - γ.

C_{max}	The makespan; $C_{max} = max(C_1, \ldots, C_n)$ is the maximum completion time. The completion time of the last job in a system, the minimal value usually denotes the good utilization of machines.
L_{max}	The maximum lateness; $L_{max} = max(L_1, \ldots, L_n)$, measures the worst violation of the due dates.
T_{max}	The maximum tardiness; $T_{max} = max(0, C_j - d_j)$.
$\sum w_j C_j$	The total weighted completion time, often referred to as a weighted flow time. Discounted total weighted completion time $\sum w_j(1 - e^{-rC_j})$, refers to a more general cost function, where costs are discounted at a rate of r, $0 < r < 1$, per unit of time. Therefore, if the job j is not completed at time t the cost $w_j r e^{-rt} dt$ is added for the period of time $[t, t + dt]$. If job j is completed at time t the cost is equal to $w_j - e^{-rt}$. The r is usually close to 0, e.g. 0.1.
$\sum w_j T_j$	The total weighted tardiness time.
$\sum w_j U_j$	The weighted number of tardy jobs.

Tab. 2.6: Typical dispatching priority rules.

CR	Critical Ratio, the ratio between the processing time and the time remaining until due date, the smallest critical ratio goes first.
$FCFS$	First Come First Served, the jobs are processed in the order in which they come. It is an analogy with the First In First Out (FIFO) queue.
ECT	Earliest Completion Time first, the jobs are scheduled according to their completion times.
EDD	Earliest Due Date first, the jobs are scheduled according to their due dates, the objective is the total maximum lateness L_{max}.
ERD	Earliest Release Date first, this is a variance of throughput times, this rule works with various criteria.
LPT	Longest Processing Time first, the rule balances the load on parallel machines. The rule is used with the objective *maximal processing time* C_{max} – the makespan objective.
LSL	the smallest slack time on the last machine
OSL	the smallest overall slack time
SPT	Shortest Processing Time first, the jobs with the shortest processing time are scheduled first. It is used with the objective *sum of completion times* $\sum C_j$.
STR	Slack Time Remaining, the time remaining before due date and remaining processing time, the smallest remaining slack time goes first (maximal lateness), $max(d_j - p_j - t, 0)$.
WI	With Biggest Weight first, the objective used is the *total weighted completion time* $\sum w_j C_j$.
$WSPT$	Weighted Shortest Processing Path first (SPT) plans jobs in descending order according to w_j/p_j, the rule minimizes $\sum w_j C_j$, when the priorities are equal, the SPT rule should be used.

In the following subsection the Single and Parallel Machine Models are described according to the objective functions. Also the preemptive and non-preemptive models are mentioned as well as precedent rules models. The models represent problems where each job is processed on one machine. These simple models are followed by more advanced models in which each job requires execution on more than one machine. Namely it is an *Open Shop* Open Shop (OS) – where the order of machines through which the job passes is immaterial, *Flow Shop* Flow Shop (FS) – each job has the same machine ordering, and *Job Shop* Job Shop (JS) – different machine ordering is possible for various jobs.

Tab. 2.7: The processing characteristics and constraints I – β.

batch	Batch processing, the machine is able to process a number of jobs, let us say b jobs, simultaneously. The jobs can have a different processing time a batch is finished when the last process is finished. [63]
block	Blocking, only in Fm and FFc with a limited buffer between two successive machines, when the buffer is full, the upstream machine is not allowed to release a finished job. [63]
brkdwn	Breakdowns mean that the machine may not be available all the time, e.g. due to shifts, scheduled maintenance etc. A survey subjected to this constraint is given in [90].
$fmls$	Job families means that the jobs are spread to F different job families. Jobs in one family can have different processing times, but they can be processed on same machines without any machine setup. The setup time is considered only when one family g is switched to another h on one machine, and it is denoted as s_{gh}. [63]
nwt	No-wait, only in Fm and FFc, means that jobs are not allowed to wait between two successive machines, this slows the start of processing on the upstream machine until the process is not sure that successive machines can process the job without any waiting. [63]
prec	Precedence constraints, one or more jobs can be completed before another job is allowed to start. There are several forms of these constraints: *chains* – each job has at most one predecessor and at most one successor), *intree* (at most one successor), *outtree* (at most one predecessor). [68], [63]
prmp	Preemptions mean that the scheduler is allowed to interrupt the processing of any job in any point of time and switch to another. When the stopped job is rescheduled to be processed again, it can be done on any machine identical to its start of processing. [68], [63]
prmu	Permutations appear only in Fm and FFc with FIFO, which denotes that the order in which jobs are processed by the first machine is given by the scheduler. [63]
r_j	Release dates mean that the job j is not allowed to be processed before r_j. If r_j is not present the job j can start any time. [68], [63]
s_{jk}	Sequence dependent setup times represent the situation that is incurred between the processing of the job j and job k. If no s_{jk} is defined there are no setup times considered. [63]

Tab. 2.8: The processing characteristics and constraints II – β.

M_j	Machine eligibility restrictions, only if Pm, and denotes that only M_j machines can process the job j [63]
res	The presence of s limited resources R_h where $(h = 1, \ldots, s)$. The job J_j requires r_{hj} units of R_h at all times during the processing. [68]
$rcrc$	Recirculation, one machine more times, only in Jm and FJm. [63]

2.2.3 Single and Parallel Machine Models

This subsection describes very briefly Single and Parallel Machine Models. The variants of models are described in Tab. 2.9. This section is present only for the completeness of the scheduling problem, but the relevance of the Single and Parallel Machine Models to this work is minimal, so these models would not be further discussed in more details.

Tab. 2.9: Possible machine environments I – α.

1	Single machine, the simplest form of a scheduling problem, this is a case of one single machine. Sometimes denoted as an empty set.
Pm	Identical machines in parallel, Pm represents m identical machines in parallel. Job j can be processed on any one of the m machines.
Qm	Uniform machines in parallel with different speeds, Qm represents m machines in parallel with a different speed of processing. The speed of a machine i is denoted by v_i. The time of processing p_{ij} (which job j spends on machine i) is equal to p_j/v_i. It is a generalization of Pm.
Rm	Unrelated machines in parallel, Rm represents m completely different machines in parallel. The machines are different in speed and also in the speed of processing of particular jobs. So, the job j can be processed by machine i with speed v_{ij}. Then, the time of processing p_{ij} is equal to p_j/v_{ij}. This environment further generalizes Qm.

Single Machine Model (SMM) environment comprises basic and simple models which started to appear in publications in the middle fifties [65], [91]. SMMs are also often used in complex environments, where the scheduling problem is decomposed to sub-problems and applied to single machines. A majority of papers focuses on the ability to solve the problem in maximally polynomial time. All these approaches are influenced by Karp's complexity study [67]. In SMM the processing characteristics and constraints such as precedence constraints, processing times, release dates and due dates are mostly investigated. Also the preemptions were investigated, but it was proved that there is no significant advantage of such constraint in SMM.

Parallel Machine Model (PMM) is of great importance to real world application, because the machines often work in parallel groups. PMM can be considered as a two step process. The first step is to allocate the jobs to the machines. The second step is the determination of the job sequence on a specific machine. Three objectives are commonly used in PMM: minimization of the makespan, the total completion time, and the maximum lateness. Most of the problems in the parallel environment are considered as offline scheduling problems. It means that all problem regarding data (processing times, release dates, due dates etc.) are known before the optimization process starts. In the online scheduling problems data are not known a priori. The processing time of the job is known at the moment of job completion. The online scheduling is under investigation more than in single machines. It is very significant for real-world applications, because the information about the job is very limited when the decision has to be made and it fills the gap between the deterministic and stochastic scheduling. In the PMM it is also more important to decide if the preemptions are good to use, e.g. when all jobs are released at the same time.

2.2.4 Shop Scheduling Problems

This subsection depicts three main problems, *Open Shop* OS – the order of machines which job passes through is immaterial, *Flow Shop* FS – each job has the same machine ordering, and *Job Shop* JS – different machine ordering for various jobs.

Open Shop Scheduling is a scheduling problem where n jobs and m machines are given. Job has t tasks and $t = m$. Each task has to be processed on different machine, so each job has to be processed on each machine at least once, even if some processing tasks can be of the zero time value. The order of processing the job by machine is not given and in fact it is a part of final solution. In another words, the routing of each job is fully up to the scheduler. This kind of problem is proven to be solvable in polynomial time, but only for two machines. Two machine problem is often a subject of investigation in literature. When there are more than two machines, the problem is classified as NP-hard. Only if there are more machines than two and the operations are of the same length, the problem can be solved in polynomial time thanks to the edge coloring problem for bipartite graphs. The typical application of Open Shop Scheduling (OSS) is a product configuration which does not depend on the order of operations. In fact, the problem is applicable everywhere as the Flow Shop Scheduling (FSS), irrespective of the order of operations.

One of the first publication which appeared on this topic [71] deals with the minimization of the finish time. The paper deals with a non-preemptive $O2||C_{max}$ ($O(n^x)$) and $O3||C_{max}$ (NP-hard) as well as preemptive $O2|prmp|C_{max}$ ($O(n)$) and $O|prmp|C_{max}$ ($O(n^x)$) cases of the OSS problem. In the OSS problem the non-

Tab. 2.10: Possible machine environments II – α.

Om	The OS represents the problem where each job has to be processed again on each one of the m machines, there are no restrictions of route, and processing times can be of zero value.
Fm	The FS represents m machines in series. Each job has to be processed on all machines, all jobs have to flow in the same route, the FIFO queue is usually used and if it is used the problem is known as a permutation flow shop, β is equal to *prmu*.
FFc	Flexible FS (or Hybrid FS, Multi-processor FS) represents the problem with c stages in series, and each stage has a number of Pm machines, every job has to be processed on all stages on any machine.
Jm	The JS represents the problem where each job has a predefined route to flow. There are two types of this problem according to that how many times the job can visit the same machine: once or more. If more times, the β value is equal to *rcrc* and it is referred to as recirculation.
FJm	Flexible JS extends the Jm by parallel machine environments. There are c work centers with a number of identical machines in parallel, each job has its own route through the job and each job is processed in each work center only once.

preemptive and preemptive cases are often discussed. The mostly used optimization criteria are the makespan [92], the release and due dates [93], [94], and the maximum completion times [95]. Also the complexity of the problem was examined [96]. The online scheduling with minimization of makespan was investigated and an optimal online algorithm was proposed [97]. One of the most frequent topic is the Two Machine OSS problem. The problem was investigated with and without preemptions, with controllable machine speeds [98], with transportation times [99], availability constraints [100], batching [101], online scheduling [102] and many other constraints and optimization objectives.

The most important in this literature research for the purpose of this work are the methods on how the scheduling problems are being solved. A linear programming formulation and a fast polynomial time algorithm for the OSS problem were studied in [93]. The GA algorithm was introduced in [79] and compared to the existing conventional search-based methods such as the Branch and Bound (BB) algorithm with a positive result. The GA algorithm and case-based reasoning to find optimal solution faster than simple GA were successfully combined [103]. At the turn of the century, several significant papers on approximation methods were published. The first work [104] deals with the TS algorithm with support of neighborhood structure

objected to the minimization of makespan. The algorithm proved to be able to find high-quality solutions for hundreds randomly generated problems in a reasonable time and was further developed for Two Machine OSS problem [105]. The second paper [106] applied the SA algorithm supported with the NS method objected again to the minimization of makespan. The papers declare that some of benchmark problems were solved to optimality for the first time by this algorithm. The paper [84] proposed a hybrid GA algorithm which incorporates local improvement procedures based on the TS and simple GAs. This step ensures the algorithm to perform search over the subspace of local optima. The results significantly outperform previous two algorithms in terms of the quality of solution. On the other hand, not all benchmark tests were solved to optimality. The competitive GA was proposed in [107].

In fact, there are only few exact solution methods available for the OSS problem. The paper [108] describes the BB algorithm which performs better than the other algorithms existing until then. A dynamic programming algorithm in two machine environment with availability constraints was proposed in [109]. A few years later, hybridized ACO with beam search was applied which became a state-of-the-art in OSS [87]. Another SA algorithm [102] was proposed, the solution is based on bottleneck and objected to minimization of $\sum w_j T_j$. An immune mechanism was incorporated to GA [110], also based on bottleneck and objected to minimization of $\sum w_j T_j$. A completely new approach based on PSO was introduced in [111]. The solution was modified according to the standard PSO, and also the Beam Search Method was incorporated. The approach was further developed and modified to Multi-objective OSS [112]. The special Network Flow based algorithm was presented in [113]. The problem is formulated as a Mixed Integer Programming (MIP) model with minimization of C_{max}. The multi agent approach with combination of SA and Fuzzy Logic was proposed in [114]. The algorithm emphasizes the flexibility of solution rather than optimality. A different hybrid GA algorithm was proposed in [115]. The paper proposes the advanced techniques to search space reduction such as special crossover operator and memory implemented to mutation which prevents a redundant solution generation. This solution outperforms other GAs. The paper [116] introduces an unusual Parallel Kangaroo algorithm based on random jumping method, the results are comparable to all well-known optimization methods. Another approximation algorithms are proposed in [117], [118], and [119].

The OSS problem can also be generalized. Concurrent OSS problems and Order Scheduling models are also considered in [120], [121], [63]. The problems $Om||\sum C_j$ and $Om|prmp|\sum C_j$ are NP-hard for $m \geq 2$ and $m \geq 3$, respectively. In the recent years also the OSS problem can also be generalized to Flexible OSS problem with job overlaps, which is under investigation. In this problem the processing times of any given job on different machines are allowed to overlap.

Flow Shop Scheduling is represented by n jobs and m machines. Each of n jobs comprises a set of tasks t which must be done on different machines. The processing operation order is the same for all jobs as they go through the machines. The tasks cannot be interrupted. Each machine can process only one task in a time. The problem is to find the configuration (job sequences) which minimizes the C_{max}. The problem is NP-hard and that is why it is usually solved by meta-heuristic methods. FSS represents the production lines and assembly lines in mass production where the machines are arranged in serial order. The representative problem can be e.g. food industry, fabric industry, automotive, aerospace and many others. This problem is often simulated with unlimited intermediate storage. This is actually the capacity between successive machines. The unlimited intermediate storage can be used when the products processed are very small. When the products are large the intermediate storage must have a limited capacity, which may cause *blocking*.

The majority of the papers deal with the minimization of C_{max} objective, because it is the easiest objective. But, in the FSS, this statement is valid only up to 3 machines. Several MIP algorithms were introduced as well as various BB methods. First implementations of the BB algorithm for FSS were introduced in [66] and [122]. The problem was also investigated in environments where the jobs are scheduled and, at any stage, there may exist multiple machines [77]. This type of problem is called Hybrid FS problem. Two Machine FSS problem was also investigated with many variations. The sequence dependent setup times solved by dynamic programming are described in [123] and [124]. The problem with jobs grouping is described in [125] and equal sized transfer batches on 2 and 3 machines [126] and also a lot streaming and sizing are quite discussed [127]. The objective function was also quite often under investigation. Researchers were focused mostly on scheduling with time lags, earliness and tardiness, waiting times and setup effects [128], also fuzzy processing times [129], problems with preemptions and non-preemptions, and availability constraints [130]. The problem focused on complexity and approximations was discussed in [72] and the problem with transportation constraints, which is quite related to this work, was studied in [131].

A fundamental algorithm for solving $F||C_{max}$ was proposed in the famous paper [91] for creation of optimal schedule. Special cases of the problem with *start lags* l_{1j} and *stop lags* l_{2j} which represent the minimal time between starting times on M_1 and M_2, and completion times, respectively, were tested in [132] and [133]. Some $F3||C_{max}$ problems are solvable by Johnson's algorithm. The algorithm solves also the $F2|prmp|C_{max}$ problem, but this problem with $F3$ is NP-hard. An important variation of the FSS problem is *no-wait* problem $F2|nowait|C_{max}$. The process has to be completed without interruption, e.g. "hot ingot" problem in which the metal has to be processed at a continuously high temperature [134], solved by the BB

algorithm [135]. The review of blocking and no-wait models is presented in [136].

As well as the OSS problem, also the FSS problem was solved by many different algorithms and their combinations and many approximability studies were published [137]. The problem with fuzzy due-dates solved by local search algorithms was described in [138]. A comparison of local search methods is presented in [139]. One of the first use of heuristic approaches is introduced in [140]. Other new heuristic approaches were proposed in [141]. This paper was dealing with a two stage problem with parallel machines at one stage and the analysis of classes of heuristics objected to minimization of C_{max}. Also the TS algorithms with support of NS techniques were developed for the FSS problem [81] and GA algorithms [142] were often improved by search space reduction, parallelization, or with multiple objective approaches.

Random Permutation FSS problems with the study of search space topology and algorithm performance are described in [143]. The performance is tested on random problems and the main topic of the paper is the validity of such testing. A Hybrid GA was proposed for a lot streaming [144]. The job (lot) in this paper is split into few smaller tasks (sub-lots) so that the successive operations can be overlapped. Another Hybrid GA was proposed in [83]. The algorithm is investigated to the intent of the effect of initialization methods and genetic operators on the performance of GA. A multi-objective evolutionary search algorithm using TSP based GA algorithm is proposed in [145]. The initialization is done by TSP with the help of a random insertion perturbation scheme. Most of the algorithms use the minimization of the C_{max} as their objective function. The recirculation problem was also solved by GA [146]. The SA algorithm proposed in [147] was focused on a reasonable running time of an algorithm which is done by a well-designed initial solution generator. The learning effects were studied in [148] and furthermore it was shown that the classical Johnson's rule is not the an optimal solution for Two Machine FSS to minimize the C_{max} with a learning effect. The differential evolution was also used for scheduling of FS [149]. The paper described a novel optimization method handling discrete variables as boundary constraints, which appeared to be widely applicable in engineering problems. The Hybrid GA [150] was also used for scheduling with limited buffers, which is a problem with strong industrial background. Simulation results and comparison benchmarks are demonstrated.

Most of the researchers ignore setup times or assume that setup times on each machine are independent of the job sequence. An immune algorithm approach with sequence dependent setup times, which can reasonably schedule complex problems in acceptable time, is proposed in [88]. Also the PSO algorithm with some hybridization [151] was used for no-wait FSS problem. This problem requires jobs to be processed without interruption between consecutive machines and shows how important this fact is in production scheduling. Then the PSO algorithm was devel-

oped [152] for the Permutation FSS problem with the investigation of some special local searching operators to balance the exploration and exploitation abilities. The hybridization and combinations of algorithms are quite often the case how to solve the FSS problem, also the use of multi-objective criteria is in focus in the recent years. A multi-objective approach based on Hybrid Quantum-Inspired GA was proposed in [153] and Hybrid Multi-Objective Immune algorithm in combination with Bacterial Optimization to find Pareto optimal solutions that minimizes both the weighted mean completion time over bar and weighted mean tardiness over bar was proposed in [154]. The effective Hybrid PSO algorithm for problems with limited buffers to minimize the C_{max} was proposed in [155].

A novel differential evolution algorithm for solving no-wait FSS problem with C_{max} and maximum tardiness is proposed in [156]. A novel approach of differential evolution described in [157] focuses on blocking problem which is objected to minimization of C_{max}. New crossover and mutation operators are developed and introduced in this paper. The SA algorithm for Hybrid FSS with multiprocessor tasks to minimize C_{max} was introduced in [158]. The proposed algorithm showed its efficiency in solving Hybrid FSS problem with multiple processors for very large problems. A completely new approach, called Discrete Artificial Bee Colony algorithm, proposed in [159], is focused on lot streaming with the criterion of $\sum w_j E_j$ and $\sum w_j T_j$ penalties under both the wait and no-wait cases. Also the efficient initialization method is used, which is based on various dispatching rules.

A Hybrid FSS problem has been examined recently, an extensive review and classification of such problem is published in [160], [161]. The GA algorithm for robust Hybrid FSS problem was introduced in [162]. The benefit of this algorithm is that it takes an uncertainty of processing time of each job into account. The algorithm is bi-objective and minimizes simultaneously the makespan and the deviation between the makespan of all the disrupted scenarios and the makespan of the initial scenario. The proposed results depict that the algorithm can generate trade off for effectiveness and robustness. The PSO algorithm to minimize the C_{max} was introduced in [163]. This algorithm with the help of the Cocktail Decoding Method significantly outperforms the majority of existing algorithms in terms of the quality of solutions, particularly for large problems. The combination of GA and SA was introduced in [164]. This paper is focused on the minimization of energy consumptions and environmental impacts. The paper proposed the mathematical model and the algorithm to solve multi-objective optimization problems. From the previous text it can be concluded that mostly the makespan and total (weighted) completion times were used in both the scientific and the engineering applications.

Job Shop Scheduling can be described as follows: the set of n jobs and m machines is given as input. Each job j comprises a set of tasks t. The tasks cannot be interrupted and each machine can process only one task in a time. The successive tasks are mostly processed on different machines. The problem is to find an optimal configuration to a given objective. JSS representatives are custom-made products or piece production, in other words, it is a problem for production and assembly lines in piece production with a large number of different products. This represents e.g. the production of machine tools or the custom-made production of cars and motorbikes. The NP-complete problems are not guaranteed to give an optimal solution in polynomial time, which means that no deterministic optimization methods can be used for a more complex problem of this type. This problem is more than seventy years old and it is still very actual. For better understanding and a more precise description of the problem, let us introduce the following notation coming from [76].

One of the first reviews of the JSS problem was published in [165]. The problems of One-Machine JSS regarding the theory of scheduling and computations were reviewed in [166] and in [167], respectively. The work dealing with the general JSS was proposed in [168]. The survey of Dynamic JSS was described in [169]. A broader view of the JSS from the point of view of robustness measures and robust scheduling was presented in [170]. A review of Deterministic JSS was provided in [78].

A simple extension of algorithm for $F2||C_{max}$ allows to solve $J2||C_{max}$ in $O(n \cdot log(n))$ time [171]. The general JS is very hard to solve optimally, the 10-job 10-machine problem formulated in 1963 [172] was solved for the first time in [173]. The solution in that year took 4 hours (17982 s) and the solution was only for a problem with no new jobs and no machine breakdowns. The most convenient problem representation of JS is possible by disjunctive graph models. In the context of JSS problem, the same or very similar problems which were mentioned for the OSS and the FSS problems are solved. The special emphasis is given on machine blocking and no-wait constraints, which are described in [174].

The JSS was often solved by linear programming, constrained programming, disjunctive programming, shifting bottleneck approaches, and BB algorithms. In the early sixties the problem [175] that involved sequencing restrictions and also non-interference constrains for individual pieces of equipment was introduced. The algorithm was based on discrete linear programming. One of the best solution was gained by the BB algorithm [176] which combined $1|r_j|L_{max}$ bound with the enumeration of active schedules. The paper [177] was focused on JSS by implicit enumeration objected to minimization of C_j. The preemptions are also discussed quite often, as well as a many other objectives. The survey of dispatching rules for various objectives were published in [178] and [179]. Another important issue, i.e. the frequency of scheduling, is discussed in [180]. The amazing work [74], dealing with the shifting

bottleneck procedure, which had an extensive impact on other publications, is objected to the C_{max}. The work deals with sequencing and the local optimization on each machine in successive order. This approach gets better results than any other solution up to this publication. The paper [181] discusses what is relevant to the JSS theory. Breakdown predictions are discussed in [182].

JSS with multi-purpose machines was described in [183], and JSS with alternative machines was described in [184]. The paper [185] proposed a new search space for sequencing problems applied to the JSS, the effectiveness was demonstrated on the problem with minimization of the C_{max}. Methodologies based on Lagrangian relaxation applied to the JSS with multiple machine types, generic precedence constraints, and simple routing considerations also proved to be computationally efficient [186]. An overview of BB algorithms based on active schedule generations was presented in [187]. An adaptation of the BB algorithm called Beam Search was also used [188] and the BB algorithm for fast JSS solving was developed in [189]. The approach solved the 10 x 10 problem which was opened for more than 20 years. The paper proposed a complete description of algorithms and presented computational results. A new variant of depth search procedure called Guided Local Search was presented in [190]. The method is based on an interchange scheme and neighborhood trees, then the procedure was embedded into the shifting bottleneck framework. Also fuzzy constraints were applied to the JSS [191]. Another approach with fuzzy processing times in combination with GA was proposed in [192].

The application of GAs was described in [193]. The GA was applied as a method generating manufacturing process plans. Another promising GA with focus on rescheduling in the JSS and the OSS was proposed in [79]. An evolution learning based on sequences of dispatching rules for job assignment was published in [194]. The algorithm performs better than any shifting bottleneck heuristic or a SA based approach. The permutation with repetition approach with GA was proposed in [195], the new crossover operator was introduced. A great tutorial survey of GA in JSS was described in [196] and hybrid genetic search strategies were described in [197]. The paper [85] proposed a multi-objective evolutionary optimization approach focused on the assignment and scheduling of jobs. The pareto-optimality approach with multiple-criteria was described in [86], [198]. The integration of process planning and scheduling was important for efficient utilization of manufacturing resources and was studied in [199]. The parallelization of GAs with focus on the Island model was proposed in [200], results were compared to traditional GAs, a significant improvement is shown. The optimization focuses mostly on one objective which is C_{max}, but also others were used, such as T_j [201], U_j [202]. The GA proposed in [203] uses global selection and local selection for initialization procedure, computational results were proved to be effective and efficient.

The Neural Network (NN) approach described in [204] is based on a linear programming network based on the Hopfield network. The key contribution of the paper was to show how to map a difficult constraint satisfaction problem onto a simple NN. A modified back-error propagation model of NN was introduced in [205]. The paper [80] describes a SA based algorithm for finding the minimum C_{max} in the JSS problem, this algorithm became the basic building block or state-of-the-art in optimization of JSS. Another approach of SA was described in [206]. An effective hybrid optimization based on SA and GA was introduced in [82], the GA was applied because of implicit parallelism mechanisms and SA was applied as the GA cannot easily regulate the convergence because of disruptive effect of genetic operators. A TS method guided by shifting bottleneck was presented in [207], [208]. The TS algorithm was also applied to the Flexible JSS problem objected to minimize C_{max} [209] – the first part of the algorithm searches for the best sequence of job operations and the second part finds the best choice of machine alternatives. The problem was that most of the papers were tested on randomly generated test problems. A TS algorithm with a new neighborhood structure was proposed in [210]. This approach is one of the most effective algorithms for JSS. Efficient initial strategies for GAs were described in [211] and quite comparable results to the best known TS algorithms were obtained.

Many other algorithms were applied to solving the JSS problem. ACO combined with TS was also used for schedule optimization [212]. Flexible JSS with the Parallel Variable Neighborhood Search algorithm was proposed in [213]. PSO was used for Multi-objective Flexible JSS [214]. The Artificial Immune Algorithm was proposed in [215]. The Hybrid Shifting Bottleneck Procedure proposed in [216] is based on a disjunctive graph model in combination with TS algorithm. And a completely new BBO algorithm for flexible JSS is proposed in [89]. The BBO algorithm is based on migration strategy of animals and is developed for searching a solution area and finding the optimum or near-optimum solution. The solution was successively compared with GAs which are the most similar group of algorithms.

2.2.5 Discussion

The SMM could be the one of the possible solutions of this work. In this case, the trucks in the warehouse would represent a single machine and the optimization of jobs could be done by many ways known from literature according to various optimization criteria. But still, there would be a problem of how to allocate the jobs to single machines. So, this is not considered as the best way of how to solve the proposed problem and the SMMs could be used later for additional optimization of single trucks. Because of the mentioned problem of allocation, the PMM appears

to be better for the purposes of this work. The PMM is investigated more as an offline scheduling problem. Since the information about the job in the case of this work is very limited when the decision of scheduling has to be made, the problem is considered as the online scheduling problem. So, the data is not known a priori and the algorithm has to handle this situation. In the case of this work, also preemptions have to be considered, because many jobs can be released in the same time. Furthermore, the preemptions can help in the situation when one worker will finish the work on the job and gives it to another worker for completion, because e.g. another worker can finish the job in a shorter time or with another equipment.

In real-world warehouses and distribution centers, the most common methodology used for process planning is material resource planning, which is more a planning tool than a solution for detailed schedule creation. In this work the material resource planning methodology is not used at all. The inspiration of how to solve the warehouse job planning problem comes from shop scheduling problems, mostly from the JSS problem. It follows from the preceding literature research that JSS could be the most convenient and proper way, because it is applicable to piece production or any other custom-made products. The most promising technique for solution appears to be an approach based on evolutionary computation, such as GA with some degree of hybridization. Since the hypothesis is stated, but the structure of chromosome is not strictly given, the GP seems like a better solution then GAs. The GP is not wide spread for this type of problems, but there are some papers.

One of the first integration of GP into JSS was introduced in [217] (2003). Unfortunately, more details are not known about this implementation, because the paper is not available for download and it was not possible to contact the authors. Another application of GP on C_{max} optimization of JSS problem was described in [218] (2009). This paper briefly reviews the JSS problem and various algorithms applied to this field of interest. The implementation of GP to JSS was described and many benchmarks were successfully tested. Also, reasonable parameters settings for the GP algorithm were discussed. The problem was objected to minimization of C_{max}. In the recent years, the GP were applied in two known cases. The first case [219] (2012) described the method for scheduling policies evolving for the Dynamic Multi-Objective JSS problem. The new hyper-heuristic method based on GP was proposed for an automatic design of scheduling policies including dispatching rules and due-date assignment rules. The evolved policies showed a promising performance on various types of scheduling configurations. The proposed algorithm was successfully compared to NSGA-II and SPEA2 algorithms. The second paper [220] (2013) also discussed dispatching rules, in particular the iterative learning.

2.3 An Overview of Routing Problems

At this point, the most interesting question regarding this work, *"How to plan and schedule jobs and allocate them to trucks?"*, should be answered by the literature research on the scheduling topic, described in the previous section. The topic of this section comprises two questions: *"How to route trucks in a warehouse?"* and of course, *"How to avoid situations, such as blocking, congestions, and collisions of trucks?"*. The beginning of this section describes the historical perspective of routing which comes directly from scheduling problems. Moreover, the fully automated routing is described in details as well as the basic methods of routing inspired by or coming directly from Vehicle Routing Problems.

2.3.1 Historical Perspective of Routing

The complete beginning of AGVs is connected to the beginning of scheduling, which is dated to the early fifties of the previous century. First, the man-aboard tow truck had been used in factories for dozens of years. The first invention in this field of automation consists of imbedded wire in the factory floor which was followed by truck. This was not actually the real AGV system, it was called a driver-less system. It worked on the principle of magnetic fields. The sensors on the bottom of a truck looked for a magnetic field. The magnetic field was crated by the current running through the wire imbedded in the floor. The stops consisted of an array of magnets, so the truck got a signal that should stop the truck at a certain place. The AGV systems at this level of technology were used until the mid-seventies. The entering of new materials and technologies evolves the AGV systems much more further. Of course, the wire in the floor is still an available and used solution. However, more sophisticated sensors or computers on board to communicate, control and manage systems bring more possibilities, e.g. laser measurement, radio frequency etc.

2.3.2 Fully Automated Routing

Material handling is one of the most expensive processes in any manufacturing or warehousing system. According to [3] it is almost 80 % of the total cost. Therefore, the attempts to reduce this cost extremely increased the need of new methodology for planning and scheduling of material handling operations. To meet this need, there are three criteria which must be met simultaneously [76]:

Sequencing which specifies the order in which jobs are processed at machines;

Scheduling makes time-phased routing and dispatching for pick-up and delivery;

Facility layout and flow-path design make efficient operations possible.

Unfortunately, most papers published in this area take into account only one or two of the three criteria. Moreover, the most of the papers consider the problem as static – all jobs are ready at the time zero, and there are no dynamic arrivals. Furthermore, all tasks of the jobs are non-instantaneous and non-preemptive. Neither a truck nor any other machine can hold more than one job at any time. The problem is then to find a feasible schedule for all employees and trucks in the warehouse so that a given objective is optimized. The recent work related to this problem is divided into three groups: (a) the Robotic Cell Scheduling, (b) the Hoist Scheduling, and (c) the AGVs. The general versions of these problems are NP-hard in the strong sense. Besides, the problems especially in real-time versions or with more constraints are so complicated that they preclude a formal mathematical formulation.

The Robotic Cell Scheduling – This problem has the fewest constraints. The cell is of a flow-line type with several flexible machines and a single material handling robot. The size of in-process buffers is either zero or finite. The main concern is to identify the optimal job input sequence and the robot operation sequence with respect to certain objectives. Two basic example solutions are depicted in Fig. 2.4.

This is a typical problem of cellular manufacturing systems, where each cell is equipped with a single material handling robot and several flexible machines [221]. Buffers between machines are very limited or zero, which makes the cell performance dependent on the sequence of robot moves. Therefore, the problem of sequencing and scheduling is quite discussed [222], [223]. The 2-machine or 3-machine cases are the most frequent cases of this problem [224], [225]. The problem is quite specific and the solution is not applicable for the purpose of this work.

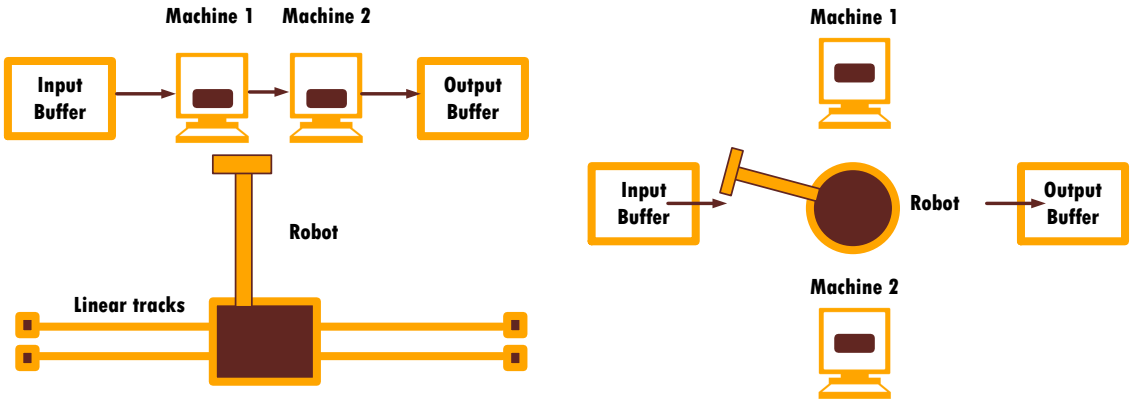


Fig. 2.4: An example of the robotic cell scheduling system.

The Hoist Scheduling Problem – This type of problem can be viewed as a special case of the FSS problem with a certain number of machines and *wait* and $non - wait$ constraints [226]. This is mostly used in electroplating and chemical industries. A typical line in this industry consists of a large number of chemical

tanks considered as machines and hoists transporting the products between tanks. Each job is a barrel carrying identical parts to be plated. A different job type may require a different treatment. A tank and a hoist can hold only one job at a time.

The cyclic scheduling of hoist with time-windows constraints is the most restrictive problem. This problem typically deals with multiple hoists in a flexible flow shop. The most distinct feature is that the job processing time at each machine is not fixed, but is strictly limited by a lower and an upper bound (i.e. the time window constraint) and collision-free constraint. For this type of problem deterministic [227] as well as stochastic algorithms [228] were investigated and developed. Also the collisions were considered [229], but still, this is not the best solution for inspiration since the work is done by track fixed hoists, see Fig. 2.5.

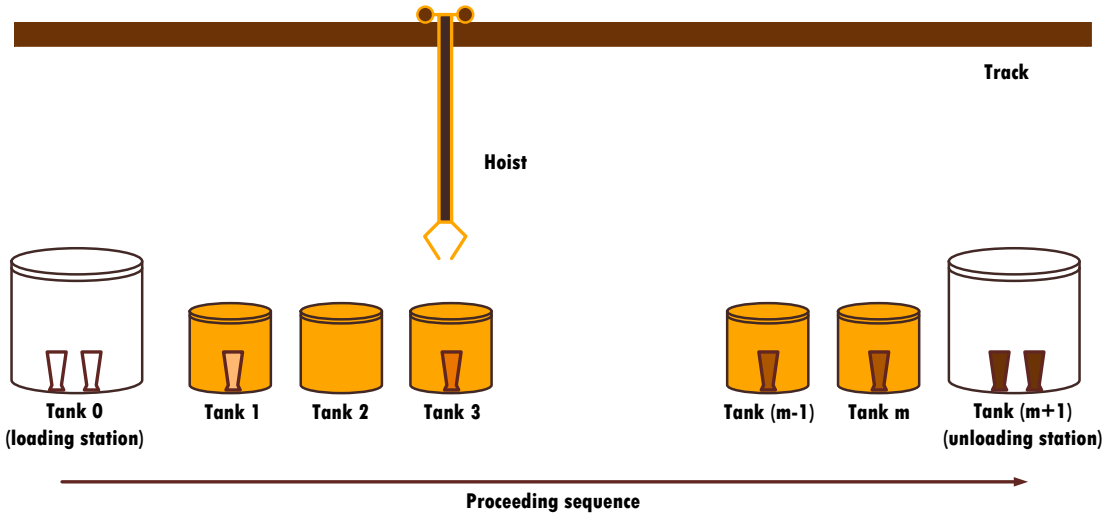


Fig. 2.5: An example of the hoist scheduling system.

The Automated Guided Vehicles – The AGV typically occurs in the process of flexible manufacturing [230], [231]. Such process typically consists of software controlled machines. Each machine has limited input and output buffers, interconnected by a material flow network. The AGV scheduling deals with an automated JSS with non-instantaneous material delivery, non-zero buffers at machining centers and multiple AGVs traveling on a shared network. The main constraint which must be satisfied by schedule is to avoid collisions of AGVs during their processing. The main problem is how to schedule the moves of vehicles so that traffic collisions are eliminated. This problem is discussed in the papers [232], [233].

During the manufacturing process, AGVs circulate on a guided paths connecting machine centers and transport materials and goods among these centers. Any improper dispatching of AGVs will immediately lead to congestions, collisions, long delays, and financial losses [234], [235]. The paths in practice are uni-directional or

bi-directional. Of course, bi-directional paths lead to better productivity, but the requirements on their implementation and control are much more expensive. The networks are commonly in the configuration of a single-loop or multi-loop network. In the single-loop network, the machines follow one loop and it is very easy to avoid all collisions there. In the multi-loop network, the network blocking is the topic of scheduling, especially in the bi-directional network. Since the paths are given by the layout design of the environment, because the vehicles are guided by the wire, laser or spot infrastructure which is given (see Fig. 2.6), all these properties have to be taken into account during the scheduling. Most analytical approaches that guarantee the optimal schedule with respect to certain objectives are limited to special cases and collisions are not much considered or solved by machine waiting than by collision prediction.

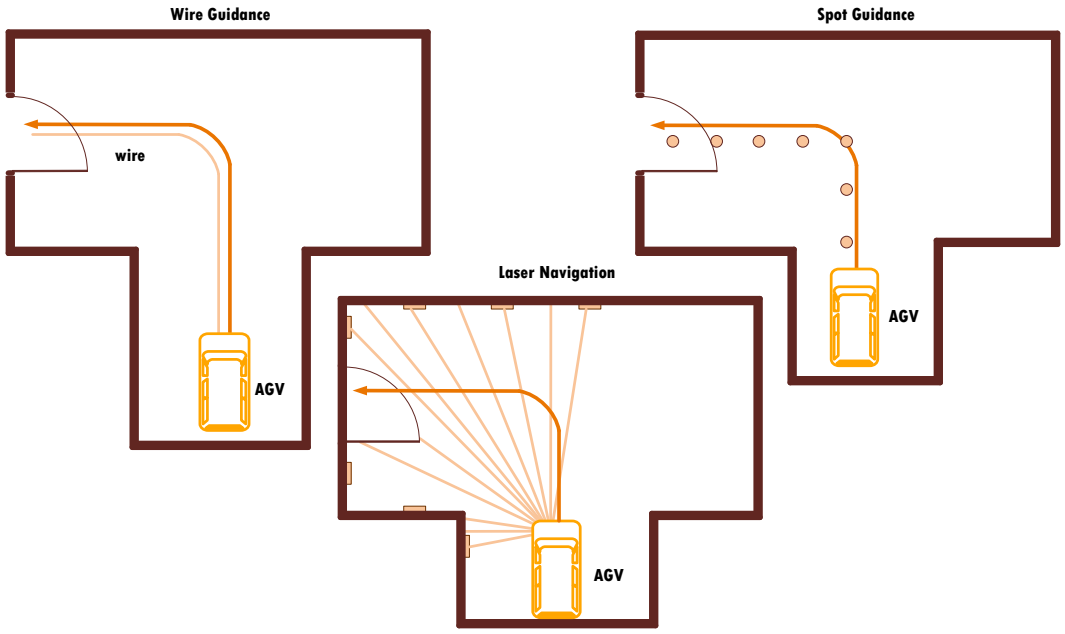


Fig. 2.6: An example of the Automated Guided Vehicle system.

There are three categories of papers on the topic how to predict and avoid collisions [230]. The first approach is to design guide-paths in a way that collisions and deadlocks are avoided. The second approach is to divide the environment into non-overlapping areas or zones, which prevents the collisions. The third approach is to use a routing strategy. The third approach could be the solution of the problem discussed in this work. The routing strategies can be divided into two groups, static and dynamic. Static routing is connected to VRP, especially concerning the subproblem with time windows, which will be discussed in the following sub-section. Dynamic routing is also somehow connected to VRP where multiple demands for customer service in real-time have to be satisfied. Analogies between the AGV and

VRP are quite clear, also the algorithms for one problem can be applied to another problem, because the main goal is to minimize waiting times of a job. The problem is that the VRP models do not take into account congestions, because the traffic network is bi-directional and if it is not, the traffic rules are given, so there is no need to predict collisions. Conflict free routes in a bi-directional network based on Dijkstra's algorithm were proposed in [236] and the solution for conflicting routes based on Petri Nets was proposed in [237]. Generally, the approaches are based on *complete route planning* or *incremental route planning*. The complete route planning plans the entire route from the point of origin to the final destination. The incremental route planning plans segment by segment until the destination is reached. The disadvantage of complete routing is that this solution is not flexible for unpredictable events which can occur all the time. On the other hand, the optimality of routes based on the incremental approach is quite disregarded.

The literature discusses the route planning and its optimization according to breakdowns and collisions, but it does not say much about the prediction of these situations itself. Also various types of interruptions might occur, e.g. vehicle breakdowns, maintenance breakdowns, objects in path etc. As a result of these circumstances the blocking can occur and jobs could not be finished on time.

2.3.3 Vehicle Routing Problem

Pickup and Delivery Problem (PDP) is tightly connected with the logistic optimization, especially with the problem of how to determine how many warehouses a company should have, where to locate them, and which customers allocate to which warehouses. In the general version of this problem a set of routes has to be constructed so that all transportation requests are satisfied. The routing process is influenced by parameters of each transportation vehicle, such as e.g. a given capacity, size, speed, start location, and end location. An extensive research related to the aspects of modeling and optimization of VRP was done in the past decades. This was done due to the need of efficiency improvement, because the traffic increases much faster than the traffic network grows. Therefore, the research is lately focused mainly on how to prevent the breakdown of the system and preserve the productivity of transportation.

In 2008 two very interesting papers regarding the PDP were published. The first paper [238] refers to transportation problems from/to a depot. This general problem is usually referred to as Vehicle Routing Problem with Backhauls (VRPB). This problem is divided into four sub-categories. In the first two sub-categories customers can represent either delivery stop (line-haul customers) or pickup stop (back-haul customers). The last two sub-categories represent a situation where

each customer requires a stop for both the delivery and pickup. The first sub-category is described by the criterion that the group of delivery customers has to be served before the first pickup customer. The second sub-category does not consider a group criterion, and mixed visiting of customers is explicitly allowed. The third and the fourth sub-categories of customers can be both the line-haul and back-haul customers, but the fourth sub-category allows the vehicle to visit a customer only once, while the third sub-category allows recirculation. More about the sub-categories is described in [238]. The second paper [239] refers to transportation problems between customers. This represents a general problem which is usually referred to as Vehicle Routing Problem with Pickups and Deliveries (VRPPD) and is divided into two sub-categories. The first sub-category refers to the situation where PDP locations are unpaired. Each unit of goods picked up can satisfy demands of any delivery customer, and only homogeneous goods are considered. The second sub-category is a general PDP with a Dial-A-Ride Problem (DARP). Both types consider transportation requests. While PDP deals with the transportation of goods, the DARP considers the transportation of passengers. More about this problems is described in [239]. There is a lot of variants of the general PDPs or VRPs, e.g. stochastic or dynamic versions of a problem, problems with limited capacity of vehicles or split delivery, time windows and other versions of these problems. On the other hand, these problems have a common basis in TSP, so the algorithms to find the solution for these problems also comes from the algorithms for TSP solving and VRP is just another combinatorial optimization problem applied to transportation, distribution and logistics domains classified also as NP-hard. In the following text, the attention is paid to the indoor VRP and the methods for collision prediction in the VRPs which could be possibly used in the solution of this work.

The most extensively studied problem related to this work is a Vehicle Routing Problem with Time Windows (VRPTW). The time windows are associated to customer visits and depots. Waiting time is allowed upon an early arrival and forbidden for late arrival. An extensive review of VRPs and VRPTW is presented in [240] and [241], respectively. The most efficient exact methods can solve problems up to 100 customers, and few instances up to 1000 customers [242], but the problems are very dependent on the instance of problem and the width of time window. Also the heuristic algorithms have been applied successfully, such as evolutionary algorithms combined with local search [243] and Hybrid GAs [244] with various types of crossover and mutation operators. When facing directly traffic network congestions, the Time Dependent Vehicle Routing Problem (TDVRP) is the most related problem which is frequently combined with VRPTW and the FIFO property for the travel times (earlier starting vehicle arrives earlier). For this problem, the frequent approaches are also based on evolutionary computation or TS [245] and

ACO algorithms [246]. Additionally, more time related attributes on routes have been introduced e.g. the velocity of vehicles, waiting-times, multiple-time windows, time-dependent services and many others. The conclusion is that heuristics proved to be efficient for this problem.

2.3.4 Discussion

Automated warehouses use a lot of variants of extensive conveyors, sortation equipment, AGVs, and ASRSs. Automated warehousing is very popular in rich countries where the land cost is very high and labor willing to do the job in the warehouse is small (e.g. Japan and Hong-Kong). The solution of automated warehouse is quite expensive and the payback period of such solution is more than two times longer than in traditional manual warehouses. Of course, the traditional manual warehouses are not completely manual. The labor there works with various types of fork-lift trucks, conveyors and other equipment which helps with transportation of goods and saves the energy of labor. The problem of choice which type of warehouse is the best for a particular company depends on many factors. If the company needs a high-throughput warehouse, the full automation is not the best choice, because the risk is increased by necessity of maintenance, and when one component is under maintenance, the whole operation of such warehouse is affected. On the other hand, the manual warehouse is more flexible when any changes are applied. In contrast to that, the fully automated warehouse has to be reprogrammed by experienced operators, while the manual warehouse in the simple way can be changed by workers (move racking, change warehouse flow, add another service).

Summarizing, in the production environments, static and dynamic algorithms have been developed to solve the routing problem. It can be concluded on the basis of the literature research in the above sections that scheduling and routing issues are studied mostly as two different areas. The integration of these two aspects is a challenging problem. It is concluded in the survey [230] that there is not much research on integration of scheduling, dispatching and routing. There is also written that more attention should be paid to the simultaneous scheduling of different types of material handling equipment incorporating capacity, space and time constraints.

The benefits of the full automated warehouse include reductions in manpower and labor costs, a fork-lift equipment and its maintenance, which implies that the work efficiency is improved. The full automation is also ideal for high density warehouses, where the manipulation with goods is minimal, product damage is reduced. Such warehouses are designed for pallets and boxes as well as other items, but the goods with special size and high weight need special treatment. And the biggest benefit is 24/7 operation without overtime costs.

Disadvantages of the fully automated warehouse are high capital investment, low tolerance to discrepancies due to mechanization, steep cost of downtime – the warehouse operation comes to a complete halt, reduced flexibility of the warehouse, and much higher maintenance costs. These reasons are the answer to the question why the fully automated warehouses are not still leading the market.

2.4 The State of the Art in Genetic Programming

This section presents a brief explanation of genetic programming. First, the basic concepts and ideas of GA and GP are described followed by the description of the conventional GP algorithm. Then, the basic aspects of GP and the research of scientific literature are introduced. Particularly, initialization and selection methods, genetic operators and fitness measurement are discussed. Furthermore, some advanced techniques and problems accompanying GP are mentioned.

2.4.1 Basic Concepts and Ideas

Genetic Programming GP [247] is a systematic, domain-independent method belonging to a group of evolutionary optimization techniques. GP is based on GA using the Darwinian principle of survival and reproduction of the fittest. GA tries to find the best solution of the given problem by genetically breeding the *population* of individuals over a number of *generations* [303]. The process is similar to biological evolution and it is based on occurring genetic operations such as *crossover* and *mutation*. The population represents a set of programs, solutions of simulated problems. Every single program in the population is called *individual*. Individuals are created on the basis of an assumed solution structure called *chromosome*. Chromosome comprises of parameters related to the problem which are known as *genes*. Genes with assigned values are *alleles* and form individuals.

GP is an attempt to deal with one of the most important questions in computer science: "How can computers learn to solve problems without being explicitly programmed?" [248]. Before the GP algorithm can be applied to a problem, there are *five major preparatory steps* for a proper operation:

Identification of terminal set – A set of terminals represents inputs of an algorithm, e.g. images, sounds, no argument functions, variables, constants...

Identification of non-terminal set – Arithmetic and logical operations, standard mathematical and standard programming operations, domain-specific operations etc. The set of terminals and the set of non-terminals are the ingredients from which the solution of problem is constructed. The sets have to satisfy sufficiency requirement and closure requirement. The *sufficiency* requirement tells that the set of terminals and the set of non-terminals are together capable of expressing a solution of the problem. This requirement ensures that it is possible to solve, or approximately solve, the problem. The *closure* requirement tells that each of the functions in the non-terminal set should be able to accept on input any value that may be returned by any

other function and any value that may possibly be in the terminal set. This ensures the validity of the solution.

Fitness measure (Evaluation) – The *fitness function* drives the evolutionary process. The fitness function evaluates how well each individual candidate solution in the population performs in a problem. The function should be *fully defined*, which should ensure that all possible candidate solutions which appear in any generation will be properly evaluated.

Parameters for controlling the run – The *primary parameters* are the population size and the maximum number of generations to be run. The *secondary parameters* are e.g. the maximum size/depth of the individual, the maximum number of nodes of the individual, the ratio of genetic operators which may be fixed or adaptable depending on the diversity of the population etc.

Terminating criterion and result designating – In general, the termination criterion is the running time of algorithm or the reach of the predefined level of accuracy of the solution. Usually, the best individual is stated as the result of the algorithm.

When the preparatory steps are finished, the GP algorithm can be designed. The general GP algorithm (see Fig. 2.7) breeds candidate solutions to solve problems by executing the following three steps [247], [248]:

Population Initialization – The initial population is generated by random composition of terminals and non-terminals with respect to the requirement of sufficiency and the requirement of closure.

Evolutionary Process – The following sub-steps are performed iteratively until the terminating criterion is met and the result is designated.

Fitness measure – Each candidate solution in the population of individuals is measured by fitness function.

New Population – New population of candidate solution is created by genetic operators. Genetic operators are applied to candidate solutions from previous generation selected with a probability based on fitness measure.

The first possibility is *reproduction*, which is simple copying of individual into a new population. This is connected to *elitism*, which means that the most fittest individuals are guaranteed a place in the next generation.

The second possibility is to use genetic operators, such as *crossover* which genetically recombines randomly chosen parts of two parental individuals, which results in two new offspring individuals, or to use *mutation* which mutates a randomly chosen part of one parental individual and creates one offspring individual. Of course, there are also other genetic operators, but these two are fundamental and considered as primary genetic operators.

Result Designating – At this stage, the method that identifies a best-so-far individual as a result of GP algorithm has to be stated. This result represents an optimal solution, or sub-optimal solution, to the problem.

The basic conventional flowchart of GP which implements the steps mentioned above is depicted in Fig. 2.7. The flowchart has been introduced by John R. Koza in [247], [248] and the mutation branch has been added. In Fig. 2.7 the RUN represents the current run number and N denotes the maximal number of runs. The variable GEN refers to the current generation number (evolution step) and M is the size of population. The index i refers to the current individual in the population. The algorithm works on the probability principles, so the probability of reproduction is p_r, the probability of crossover is p_c, and the probability of mutation is p_m.

During the evolutionary process the population is developed by genetic operators. Because the population is involved, GP algorithm is considered as a parallel search algorithm. While the parental individuals are affected by genetic operators, more offspring individuals than in the original population can occur. The number of individuals in the population may have a rising trend, or may have a maximum number of individuals preserved in each population, due to the already high computational complexity of the optimization process. New offspring individuals are selected to a new population based on evolutionary principles, so that the fittest individuals are most likely to participate in the creation of a new population.

When comparing GP to GA the main difference is in the representation of data structures of individuals. GAs usually handle linear data structures of a fixed length, which is usually represented by binary data. In contrast to that, GP operates with a population of hierarchically structured individuals which represent computer algorithms or programs. GP concept was originally designed for tree expressions based on LISP programming language and its S-expressions. Of course, GP is not tied only to one language and can be used in any other language. Fig. 2.8 shows the tree representation of the mathematical expression and its form written in the prefix standard used in LISP and also in the commonly known infix notation.

Although GP, originally designed by John R. Koza [247], [248], [249], works with tree data structures, it can also by adapted to other data structures. It should also be noted that the Tree Genetic Programming (TGP) is based on the general graph data structure which also includes Cartesian Genetic Programming (CGP) designed by Julian Miller in 1998 [250]. Julian Miller originally derived the CGP data representation from electrical circuits, designed a few years earlier with Peter Thomson. Another widespread type is a Linear Genetic Programming (LGP) which works with linear data structures. LGP was comprehensively described by Markus Brameierem and Wolfgang Banzhaf in [251]. Of course, there are many other types of GP enjoyed

in the scientific community, but they are beyond the scope of this thesis. A brief summary of the types of GP may be found in [252]. Since the early beginning of GP good results in various fields of interests were obtained, e.g. in finance prediction, economic modeling, control and optimization of industry, medicine, biology, bio-informatics, etc. The results of GP are comparable to human or even surpass human, which is proven by thousands of papers dealing with GP applications [253].

The evolutionary process is based on blind random search. One of the most discussed topic in algorithms based on randomness is *"How to reduce the search space and preserve only valid individuals?"* and simultaneously satisfy the closure

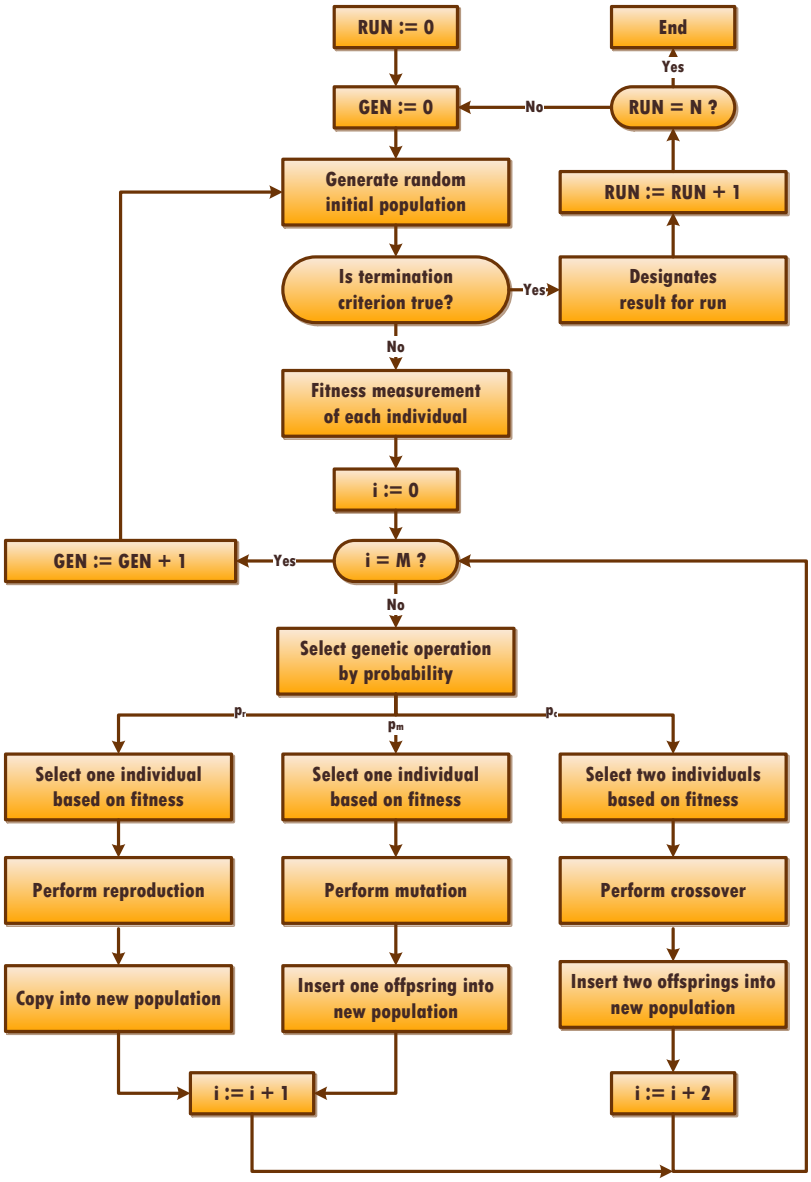


Fig. 2.7: The conventional flowchart of the Genetic Programming algorithm.

requirement. The performance improvement of GP can be done by search space reduction. Limitation of building blocks forming a syntactic trees reduces the search space by elimination of meaningless combinations of blocks [254]. The basic techniques to reach this goal is to use Strongly Typed Genetic Programming (STGP) [255]. STGP is an enhanced version of GP which enforces data-type constraints. The combination of generic functions and generic data types makes GP more powerful among other type-constraint enforcement approaches. Furthermore, there are methods based on grammatical conditions limiting the generated syntax trees representing a candidate solution to a problem. These techniques include Grammatical Evolution [256], [257], [258], Context-Free Grammar [259], and Context-Sensitive Grammar [260], [261]. GGGP is an extension of traditional TGP and solves the closure problem using a Context-Free Grammar rules, which provide a formal definition of the syntax tree rules of the problem. An example of such a definition of rules is shown in Fig. 2.9. Methods using grammar or any of the other techniques reducing the search space usually require special rules which control the initialization methods and genetic operators. Detailed information related to GGGP are described in the following sub-sections.

2.4.2 The Process of Initialization

GP algorithms begin by creating an initial population of individuals. Typically, the individuals are generated randomly across the entire search space. Initialization is an important process, since it can significantly affect the rate of convergence to the result. The initial population must contain a sufficient degree of diversity among the individuals on which basis it will be possible to create following populations with promising individuals in forthcoming generations.

The simplest initialization methods are based on random coupling of terminal and non-terminal symbols. In this case, only the closure requirement is considered,

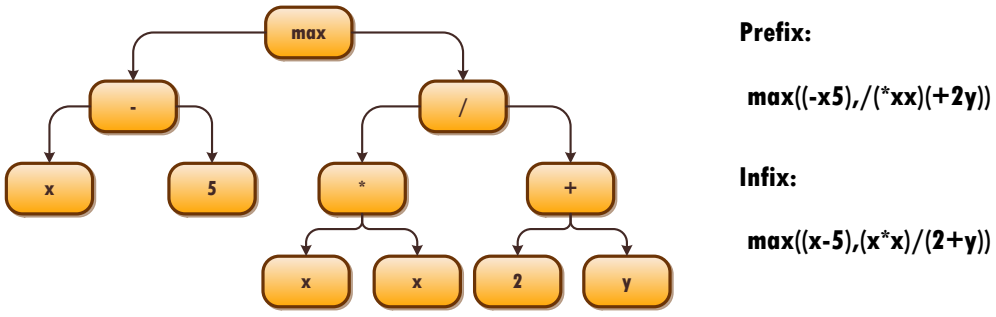


Fig. 2.8: The tree-based chromosome with prefix and infix notation.

so all individuals remain valid. There are two basic methods using randomness, the *full method* and the *grow method*. The full method generates syntax trees of depth n and the grow method generates syntax trees of depth 1 to n, where n is the maximum depth of the syntax tree. Grow method can generate population with much more diversified individuals. The method combining both approaches was created and named the *ramped half-and-half method*. This method generates a half of the individuals using the full method and the other half using the grow method. The maximum depth of the syntax trees generated is ramped, such that individuals are created in a range of sizes. The method is recommended by John R. Koza [247] just because of a rich variety of different shapes of syntax trees.

The restrictive conditions, such as a uniform distribution of individuals across the search space, or criteria related to a defined structure of the problem are used in more sophisticated initialization methods. One of the first algorithms, designed by Hitoshi Iba [262], is called the *RAND-TREE*. This algorithm puts emphasis on the structure of syntactic trees to be uniformly spread across the search space (approximate spreading). In the same year, Bōhm and Geyer-Schulz proposed an algorithm with the exact uniform distribution [263]. In the year 1997 Kumar Chellapila [264] proposed the algorithm *RANDOMBRANCH* with an approximately uniform distribution and the algorithm was focused on the speed maximization of tree creation. Another example might be the Probabilistic Tree Creation (PTC) method designed by Sean Luke [265]. S. Luke introduced in this publication two initialization algorithms, each of them was introduced in a randomized form and also in the form based on STGP. Another interesting approach is the solution based on Context-Free Grammar. First attempts were made by P. A. Whigham in [259], [266] and in 2006 by a team of researchers from the Polytechnic University of Madrid [267]. The last

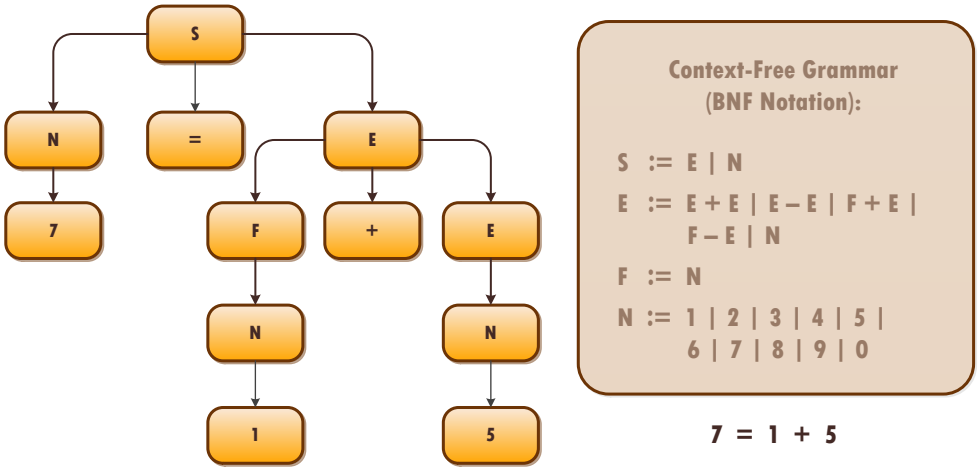


Fig. 2.9: The context-free grammar and generated syntactical tree.

method is compared with the earlier approach in [267] and shows the best results in finding solutions to a symbolic regression, which was found in the smallest number of generations of all compared methods. This method was studied and implemented in practical experiments during the research taking place in the context of the work.

2.4.3 The Process of Selection

During the evolution process, the individuals from the existing population are selected as parents to breed a new population. Usually, the choice is based on a fitness value and the fittest individuals are more likely to participate in the breeding process. In spite of that, the fittest individuals may not necessarily be chosen to participate in the process creating the new population, as well as the weakest individuals may not be excluded from the breeding process. When the selection method is applied, it is necessary to take into account an evolutionary biology. In the evolutionary biology it can also happen that a stronger individual can be excluded from the breeding process due to accidental death and the weakest individual may participate. The main point of success during the selection process is to ensure the diversity of population so that the possibilities of evolution process are not limited due to lack of diversity, also the premature convergence to solution and getting stuck in a local extreme should be avoided. The degree of producing the high quality individuals for breading a new generation is called selective pressure. The value of selective pressure depends on selection methods. If the method with a high selective pressure is used, there is a danger that the process may get stuck in a local optimum. On the other side, when the selection pressure is small, a larger space of possible solutions is scanned, and the algorithm converges to the solution very slowly.

Selection methods are described in almost every book dealing with GAs [268] and GP [247]. There is a large amount of selection methods. The most commonly used methods will be described in the following text. The most used and recommended selection method is the *tournament selection*, also the *roulette wheel selection* and the *rank based selection*, which is only a variation of the classical roulette wheel selection. The tournament selection is inspired by the rivalry of animals. First, the subset of individuals m (m is usually equal to 2) is selected. Second, one of m individuals is declared as the winner of tournament and has the opportunity to participate in the breeding process. The roulette wheel selection represents a selection mechanism in which the selection of individuals is directly proportional to the fitness evaluation of individuals, and it is one of the most used selection mechanism. This mechanism imitates natural selection, thus the fittest individuals most likely survive. The disadvantage of this method lies in the fact that several individuals with a relatively high fitness value take the biggest part on the roulette wheel, which

prevents the method to select the individuals with worse values. At first sight, this does not seem bad, but this approach significantly reduces the diversity, which has a big impact on the whole evolution process. This should be prevented by the rank based selection, in which individuals are selected on the principle of roulette wheel, but instead of fitness values the order of individuals in population is used.

2.4.4 The Process of Breeding

The core of each GP algorithm and driving mechanism of population breeding are genetic operators. Genetic operators drive the creation of each new individual (*offspring*) by modifying individuals (*parents*) selected from the previous population. Originally, the operators were divided by John R. Koza [247] into two groups, *primary operators* and *secondary operators*. Initially, the primary operators were represented only by *reproduction* and *sexual recombination* known as *crossover*, and secondary operators were represented by *mutation*, *permutation*, *editing*, *encapsulation*, and *decimation*. In many publications, the mutation operator is considered as a very powerful operator and is classified as one of the primary operators. In this work, the mutation operator is also considered as a primary operator.

The *reproduction* operator is based on the principle of natural selection of the fittest individual. The operation is asexual, which means that it operates only with one parent and produces only one offspring. The operator works in two steps. In the first step, the parental individual is selected. In the second step, the selected individual is copied, without any alternation, to a new population. The process strongly depends on the used selection method. The process is also called *elitism*, which means that n elite (fittest) individuals are reproduced to guarantee that the level of fitness value of the best individual is not decreasing. Monotony of evolution is ensured right by reproduction operation which simply copies the individuals.

The *crossover* operator represents a sexual recombination, which means that this operator operates with two parents and also produces two offsprings. The parents are in most cases chosen based on the selection method and fitness measure. Then, a randomly selected crossover point in each parent has to be selected. Now, two sub-trees for swapping are selected, the root of each sub-tree is the crossover point in each parental individual. After the sub-tree swapping operation, two new offsprings are inserted to the new population. Because of the closure property, both new individuals created are syntactically valid. The crossover operator is the most commonly used genetic operator. There are many variants of crossover operator and their contribution to the evolutionary process is compared in [269] and [270]. The simplest variant operators do not follow any complicated rules and the only rule is to follow the closure property which ensures generating of only valid individuals.

On the other hand, there are also more advanced techniques of using GP to confine the search space and speed up the convergence to the result. These techniques also require modification of initialization method as well as modification of genetic operators with respect to the validity of individuals.

The standard crossover, designed by John R. Koza [247], is based on sub-tree swapping which randomly selects the crossover points in the first and second parent and swaps sub-trees, which creates two new individuals. It is worth noting that this operator can generally work also as an incest crossover operator or the multi-parent crossover. In incestuous crossover, the process of recombination operates with two identical parents and different crossover points are selected [247], which can cause the creation of two new individuals. However, incest prevention mechanism is often implemented, because the diversity of population is decreasing when one parent is used very often. Multi-parent crossover [271], [272] is based on diagonal crossover points [273]. Crossover points are randomly selected in all n parents, and n sub-trees to swap operation are produced. Selected n sub-trees are swapped along the main diagonal, so that each of the newly created individuals includes a new sub-tree.

The One Point crossover [274] is a more sophisticated method of recombination. Operator works on the same principle as a standard Sub-Tree crossover, but selects the second crossover point depending on syntactical similarities with the first selected sub-tree. This operator was also often studied from the perspective of scheme theory predicting behavior by mathematical model [275]. The Size-Fair crossover [276], based on a similar size of individuals, a sub-tree with the same or near same shape is selected in both parents. Crossing points are chosen in these sub-trees, and the proper sub-trees are swapped. This approach has an advantage in that it is carried by a sufficient amount of genetic information from parents to offspring, which precedes the destruction of the genetic information of an individual based on a defined scheme. The loss of genetic information is discussed in [277]. Another advantage of this approach is that it takes into account the size of individual (the depth of the syntactical tree), which prevents the problems with code bloat.

The more complex variants of crossover operators take into account the semantic meaning of the individuals or the context of the problem. An example of these might be Context Preserving Crossover (CPC), which exists also in STGP version [278], Whigham's Crossover (WX) [259], Grammar-based Crossover (GBX) [279], [280] or Semantics Aware Crossover (SAC) [281], [282]. WX [259] provides enhancements in the form of crossing point selection. These points are selected randomly, but the same non-terminal symbol with the different sub-tree is selected in each individual. This ensures, to some extent, the preservation of genetic information. The last two mentioned methods will be discussed in more details. GBX provides a possibility to define the system, how the new individuals will be composed, which brings a

significant advantage for the search space reduction. The three most valuable features of the GBX crossover are that a) it always generates a valid individual, b) searches all nodes of parental individuals that can generate new rules depending on the grammar and can possibly lead to the desired solution, and c) provides an effective mechanism to control code bloat. GBX is based on a set of rules defined in [279], [280]. The comparison of GBX, WX, *One Point Crossover* and *Size-fair crossover* is described in [279] on the example of symbolic regression. When the GBX was used, the result was found in the eighth generation, using the WX in the tenth generation, using the *One Point Crossover* in the fifteenth generation, and using the Size-Fair Crossover also in the fifteenth generation. The test was repeated a hundred times and measurements were arithmetically averaged. Another interesting approach, published in the recent years, is a crossover based on problem semantics [281], [282]. The paper [281] presents two kinds of crossovers. SAC and Semantic Similarity-based Crossover (SSC). These methods are not based on the principles of grammar which defines the limits between the syntactical structure of trees. These methods are based on the semantic distance calculation between the selected subtrees for swapping operation. It is an interesting approach that allows to maintain a certain degree of genetic information of parents and simultaneously modify its certain parts and create new offsprings. A disadvantage may be the result of slower convergence, because the calculation of semantic distance is quite time demanding.

Mutation operator is mainly used in the case when the population is loosing diversity of individuals and the evolution process tends to converge prematurely. The mutation operator is asexual and works only with one parent. When the individual is selected with a given probability, the mutation point is randomly selected, which gives the sub-tree that will be changed. The operator then removes the sub-tree under the mutation point and the removed code is replaced by a newly generated sub-tree, which gives the new offspring the new population. This mutation is called a *Sub-Tree Mutation* [283] and nicknamed as Headless Chicken Crossover operator. Another mutations used are *Point Mutation*, *Hoist Mutation*, and *Shrink Mutation*. The Point Mutation generates only a new mutation point with the same number of arguments, but the whole sub-tree is completely preserved [284]. The Hoist Mutation replaces the sub-tree of mutation point by a randomly selected branch of sub-tree being currently removed. The Shrink Mutation replaces the sub-tree of mutation point by randomly generated terminal symbol. Another mutations tries to optimize the structure of the sub-tree by using other secondary operators.

The most important types of mutations for this work are Grammar-based Mutation (GBM) [280] and Semantically Driven Mutation (SDM) [285]. GBM is made â€‹â€‹by the same authors as GBX and works on the same principle, using the Context-free Grammar. The SDM algorithm generates a completely different sub-

tree than the original sub-tree, which ensures that the resulting offspring will achieve very different results. This approach drives the required diversification in population if needed. The authors of SDM claim that the operator is not resistant to code bloat, which is the aim of further work related to this operator. Mutation operators, in comparison with the crossover operators, are used only in a small percentage of population breeding, especially when there is a lack of diversification as it was written above. GBM was selected for the experiments in the scope of this work.

2.4.5 The Process of Evaluation

Let us suppose that there is a relatively randomly generated population where fitness for current environment of all individuals is on a pretty low level. However, some individuals show better results than others and so it is necessary to measure how the individuals fit to the observed environment. For these reasons, it is essential to design a special evaluation function which is also called a *fitness function*. In general, the fitness function can be defined as a number of errors between the veritable output and the desired output of the GP algorithm, the amount of time needed to calculate the desired accuracy of the resulting individual, etc. The fitness function can be represented in different ways, which depends on the nature of the solved problem.

The most general form of fitness function is a *raw fitness*. The raw fitness represents the difference between the achieved and expected value of the result. Obviously, it is a minimization problem. The raw fitness is often transformed to the reference value, e.g. *standardized fitness*. So, it is always a numeric value, where the lower value represents a better solution (the best solution is represented by 0). Other basic forms of the fitness function are *adjusted fitness* and *normalized fitness*. Both functions ensure that the resulting value lies in the interval $< 0; 1 >$ and an optimal result is assessed by value 1. Besides of these basic forms of fitness functions which were defined in the early years of evolutionary computation techniques [247], user is able to define own function which best suites to the solved problem.

2.4.6 Advanced Techniques

Among the most used advanced techniques in GP are the already mentioned restrictive conditions of a search space, e.g. context or context-free grammar etc. Other solution is a grammatical evolution which comes from an idea to separate the genotype from the phenotype. The genotype is an ordered list of numbers which code the selected rules from a defined context-free grammar. The phenotype then is a tree-like data structure which is evaluated by fitness function. In the standard GP algorithm the genotype and phenotype are interpreted as the same object. There is

also a lot of variants of grammatical evolution, e.g. the most interesting enhanced version is searching the variants of genotype by GA or PSO. This problem domain is also connected to the *schema theory* [275], [286], [287]. The theory, in general, says that schema (a template) identifies parts of individuals with similarities at certain part of individuals' topology. Based on these findings, a certain part of individuals is marked as optimal and are not changed in further process.

Other advanced techniques used are automatic learning and auto-organizing operations that alter the structure of individuals and are known as Architecture-Altering Operations (AAO). The AAOs simply allow to automatically create and delete functions for altering the structure of individuals and change their number of input parameters. The first representative of such functions are Automatically Defined Functions (ADF)s [247], [288]. The purpose of ADFs is to encapsulate repetitive parts of the source code or sub-structures of individual, which can later be reused many times or may become a part of a terminal symbol set. These techniques also include Automatically Defined Iterations (ADI), Automatically Defined Loops (ADL), Automatically Defined Recursions (ADR), and Automatically Defined Storage (ADS). There is also a lot of other techniques related to architecture organization, but it is beyond the scope of this thesis.

The advanced techniques also cover the parallelization of source code of GP algorithm and distributed models. There are two basic models of parallelization: the Coarse-Grained Model so called *Island Model* and the Fine-Grained Model so called *Cellular or Grid Model*. There are also other possibilities such as parallelization on graphic cards, field-programmable gate arrays and others, but for the purpose of this thesis the first two models are the most important.

The Island Model divides the population into smaller finite sub-populations called *demes*, between which some migration can occur. There is a standard GP algorithm which is responsible for initialization, evaluation and evolution of the entrusted sub-populations above each of these sub-populations. The standard GP algorithms are commonly enhanced by the migration operator which periodically exchanges the individuals across all sub-populations. How many individuals and how often they are migrated depends on the setting of migration operator.

The Cellular or Grid Model assigns each individual to a single cell in a cellular network. The population is essentially a system of active individuals who interact only with the neighbors in the immediate surroundings (4 neighbors in the von Neumann neighborhood and 8 neighbors in the Moore neighborhood). Evaluation, selection and reproduction are ensured by a common coordinating algorithm for all cells. There is also a possibility to use a migration operator which in this case is used only for swapping of the individuals in distant cells.

2.4.7 Known Problems

In spite of the indisputable advantages of GP algorithms, there are also some tricky problems which have to be handled. These problems include, e.g. *introns*, which are basically non-functional parts of the source code. Introns cause *code bloat*, which means that they generate a code with no apparent functionality, e.g. $a+0$, $a*1$, etc. The first problem mentioned is a *code bloat* and another, but very similar problem is a *code growth*. These two terms are often used interchangeably, however, they represent something a little bit different. The code bloat [289], [290] is an issue in which an individual is expressed by many introns or operations that could be simplified into a shorter term with the same meaning, while the code growth [291], [292] is undesirable in terms of time and memory consumption, even though it may bring some positive results in the context of an individual.

Other problems are directly connected to the GP algorithm and its setting. The basic problems are how to choose the correct number of individuals [293] or the number of generations when the evolution process is handled in this way, or how to preserve a reasonable degree of diversity of individuals in population, which is related to the genetic operators used, and to the probability of their usage and many other parameters which can significantly affect the evolutionary process and convergence to the desired solution. The question of diversity is very important. If the diversity becomes very low, the evolutionary process tends to *premature convergence* and the result will get stuck in a local optimum. Otherwise, if the diversity is too high, the algorithm is searching in a very wide search space, which reduces the rate of convergence. This is also somehow related to redundancy of individuals, which can be prevented by a wisely chosen initialization method and selection mechanism.

Other problems are the problems of *over-fitting* and *under-fitting*. If the GP algorithm is trained on a small set of training data, it achieves excellent results (high fitness value), but if the algorithm is applied on new and unknown data, the results are very poor (low fitness value), which means that the generalization of solution generated by GP algorithm is too low and the algorithm memorized the training data. The opposite of this problem is the *under-fitting*, which means that the algorithm is trained and the degree of generalization is too high. Another problem may be the time-consuming calculation of the fitness function, which can be to some extent solved by the fitness cache.

2.4.8 Discussion

The GP algorithms are used in various fields of interest in which they often exceed the solutions designed by human experts. Of course, there is a lot of problems the GP algorithms suffer from, but with the expert design of preparatory steps of

GP algorithm, lots of them can be suppressed or completely avoided. This section mentions the most frequently used GP techniques and approaches, which served as an inspiration for the implementation of the GP algorithm used in this thesis.

Since the design of the GP algorithm is not quite an easy task, it was decided that the Evolutionary Framework will be designed based on modular architecture and the GP algorithm will be the first module. This decision seemed to be a reasonable step, because it can be reused in future and easily extended by other possible modules, such as other artificial intelligence and machine learning algorithms.

The proposed GP module will be built on the CFG and the core of the algorithm will run on tree-like data structures. When the CFG was used, the standard genetic operators were adjusted. According to the literature review done in this section, the best choice, easily implemented and powerful, seems to be the approach based right on the CFG, which was used with the initialization method Grammar-based Initialization Method (GBIM), crossover GBX, and mutation GBM. Also the semantically driven operators are quite interesting, but this approach was obviated because of the computational demands and the problem with similarity measurement, which is quite difficult for some types of problems, and of course, the similarity measurement is also computationally expensive when the algorithm operates with data, such as audio, images, and video.

The proposed GP algorithm as well as the Java Evolutionary Framework is described in Section 6 [304]. The use of the proposed algorithm was then adjusted to the JSS problem applied in logistic warehouses and distribution centers, which is described in Section 7 in details [305]. The proposed GP algorithm was tested also on other examples, such as text processing and the emotion recognition [306], [307], and image processing and image mining [308], [309], [310], and [311].

3 THE OBJECTIVES OF DISSERTATION

This chapter is divided into two sections. Section 3.1 describes the scientific hypothesis which will be subjected to investigation in the following chapters. Section 3.2 describes the partial goals which should directly support and confirm the determined hypothesis and reach the specified goals of contribution.

3.1 Hypothesis

The testing hypothesis of proposed doctoral thesis has been determined as follows:

If an appropriate combination of the Job Shop Scheduling and the Vehicle Routing Problem solving techniques will be implemented by the Genetic Programming algorithm driven by the Context-free Grammar and the Multi-Criteria Fitness Function, then the current state-of-the-art of work-flow scheduling in logistic warehouses and distribution centers used in real-world warehouse environments can be outperformed.

The following section describes the specific goal of this project from the economic point of view and partial goals, which should help support and confirm the determined hypothesis in this section, more particularly as well as the main points of realization.

3.2 Goal & Partial Goals

The main goal, coming from the hypothesis stated in the previous section, is a

"substantial increase of productivity and reduction of operating costs".

The specified goal can be reached when the number of executed manipulations with commodities will be continuously increased over time, the bottlenecks will be analyzed and the material flow in the warehouse will be increased, the transportation paths will be shortened as much as possible, the scheduling of jobs will be done also with respect to employees' performance, trucks' velocity, size and manipulation possibilities, the number of employees and trucks will be reduced as well as the overtime bonuses, which will together lead to reductions in operating costs and significant acceleration of the return on investment in the warehousing environments.

The partial goals of the thesis are defined as follows:

1. **Involvement of the human factor in the optimization parameters.** The involvement of the human factor in the optimization parameters enhances the standard mathematical model (see Chapter 4), which becomes more complex, but could also help to save the processing time. Heretofore, only the parameters related to jobs and machines have been considered. Since the performances of particular employees considerably differ from each other, it is possible to save the processing time when the jobs are scheduled with respect to this parameter. Similarly, when different types of jobs are scheduled, it is possible to prevent inadvertent errors in processing, because some employees are more conscientious than others, and save time which should normally be used for checking and controlling of a job done by the previous employee.

2. **Involvement of the multi-criteria fitness function for optimization.** The involvement of the multi-criteria fitness function in the optimization process can respect more than one customer's requirement to which we are trying to optimize the solution. This approach also includes a subjective weighting factor used in the current software used by warehouse operators. This partial goal basically describes the fitness function working with respect to the optimization of time processing and the number of collision situations, as well as other factors such as the balanced workload of employees. The minimization of mutual crossing of manipulation routes of trucks is necessary to solve for decreasing collision situations. This solution could also minimize the possibility of commodity and equipment damage. Moreover, it is also possible to extend the fitness function by employees' performance, trucks' velocity, size etc.

3. **The variability of time planning and simulation (minutes/hours/shift).** Generally, the jobs are scheduled ad-hoc and when the jobs on work-plan are done, the employee gets a new work-plan. Thanks to the databases of incoming jobs and evolutionary processes the system should be able to schedule the work for a few next minutes as well as for the whole working shift. There is no problem to make the plan presently, but for the sake of missing simulation and visualization, nobody is able to say where the employee is working in the specific time, which leaves a space for further time optimization. So, what is needed is a dynamic simulation of job scheduling in the warehouse environment for highly qualified assessment by operational manager, which could lead to further optimization and changes. All of these will be done by visual simulation of how the scheduled work is done.

4. **The possibility of co-operative jobs.** The co-operative jobs are a great possibility how to decrease the time of job processing. Generally, each employee has his own work-plan and has to fulfill assigned jobs. The co-operative jobs help to solve situations when one employee is able to unload the pallet from the truck, move it to a specific cell in the warehouse, but is not able to store it to the rack due to the limits of used equipment. So, another employee can continue the work and store the pallet to a higher level of the shelf, which could not be carried out by the first employee with the fork-lift hand pallet truck without changing the equipment. Besides this fact, the employees may not have an authorization for all types of equipment.

5. **Design, implementation, and validation of the framework.** Another goal is to design and to implement the evolutionary framework, which should be able to run on genetic programming algorithms driven by context-free grammar. The possibility of an easily extensible design is very welcomed as well as a flexible fitness function definition with further extension possibilities. The GP algorithm or the framework itself have to be validated, and then adapted in the way that they could work with a defined JSS problem applied to the logistic warehouse environments and distribution centers.

6. **Design, implementation, and validation of benchmark tests.** The last, but not least, goal is to design and to implement benchmark tests, which would prove the functionality of the proposed solution and prove that the hypothesis is true. The validation will be done also by benchmark tests extracted from the real-world warehouse environments.

4 THE MATHEMATICAL MODEL

The common mathematical model for the flexible JSS problem used across the research publication (see [294], [295], [296], [297]) is described in this chapter, see Section 4.1. The basic notation – parameters and decision variables are described as well as the most frequent optimization criterion and the common constraints to which the optimization criterion is subjected to. The second part of this chapter, Section 4.2, describes the extension of the Flexible JSS problem model, specifically how it is possible to use the common Flexible JSS model in the warehouse and distribution center environments. This section also describes which parameters are used, which variables are known or not known at the beginning of scheduling and the additional constraints are also described.

4.1 The Job-Shop Scheduling Model

The standard Flexible JSS problem is formulated as follows. The Flexible JSS problem has m machines and n jobs. Each job consists of a sequence of operations $O_{j,h}$, $h = 1, \ldots, l$, where $O_{j,h}$ stands for the h-th operation of job j and l stands for the number of operations required for job j.

The set of machines is noted $M, M = M_1, \ldots, M_m$. The specific machine is indexed by i. The set of jobs is noted $J, J = J_1, \ldots, J_n$. The specific job is indexed by j and the specific operation of job is indexed by h. The specific operation of job is noted as $O_{j,h}$ and requires one machine of suitable machines $M_{j,h}$ out of a machine set M ($M_{j,h} \subset M$), which are together described by processing time $P_{i,j,h}$. The subset $M_{j,h}$ is defined by $a_{i,j,h}$. Then, the index k is defined for each machine which describes the sequence of allocated operations according to the weight of job (a priority factor) w_j. The complete notation is described in Tab. 4.1.

Tab. 4.1: The general notation of the Flexible JSS problem.

h	index of operation, $O_{j,h}$ denotes operation h in job j, $h = 1, \ldots, l$
i	index of machine, M_i denotes machine $i, i = 1, \ldots, m$
j	index of job, J_j denotes job $j, j = 1, \ldots, n$
k	index of allocated operation for specific machine,
k_i	number of operations assigned to machine i
l	number of operations, each job has a different number of operations
m	number of machines which can be used for job processing
n	number of jobs, each job consists of a sequence of operations
w	weight also called a priority factor, w_j weight of job j

In order to describe the Flexible JSS model precisely, the following parameters and decision variables must be introduced (see Tab. 4.2). The mixed integer linear programming model formulation is also described in the following text.

Tab. 4.2: The parameters of the Flexible JSS problem.

$a_{i,j,h}$	describes the capable machine set $M_{j,h}$ assigned to operation $O_{j,h}$
$p_{i,j,h}$	processing time of operation $O_{j,h}$ performed on machine M_i; $p_{i,j,h} > 0$
$Ps_{j,h}$	processing time of operation $O_{j,h}$ after selecting a machine
$t_{j,h}$	start time of the processing of operation $O_{j,h}$
$Tm_{i,k}$	start of working time for machine i in priority k
L	a large number

$$a_{i,j,h} = \begin{cases} 1 & \text{If } O_{j,h} \text{ can be performed on machine } i, \\ 0 & \text{Otherwise,} \end{cases}$$

$$x_{i,j,h,k} = \begin{cases} 1 & \text{If } O_{j,h} \text{ is performed on machine } i \text{ in priority } k, \\ 0 & \text{Otherwise,} \end{cases}$$

$$y_{i,j,h} = \begin{cases} 1 & \text{If machine } i \text{ is selected for operation } O_{j,h}, \\ 0 & \text{Otherwise,} \end{cases}$$

The objective is to minimize the makespan (C_{max}).

Subjected to the following constraints:

$$C_{max} \ \geq \ t_{j,h_j} + Ps_{j,h_j} \quad (\forall\ j); \tag{4.1}$$

$$\sum_i y_{i,j,h} \ \cdot \ p_{i,j,h} \ = \ Ps_{j,h} \quad (\forall\ j,\ h); \tag{4.2}$$

$$t_{j,h} \ + \ Ps_{j,h} \ \leq \ t_{j,h+1} \quad (\forall j,\ h = 1, \ldots, l-1); \tag{4.3}$$

$$Tm_{i,k} \ + \ Ps_{j,h} \ \cdot \ x_{i,j,h,k} \ \leq \ Tm_{i,k+1} \quad (\forall\ i,\ j,\ h,\ k = 1, \ldots, k_i - 1); \tag{4.4}$$

$$Tm_{i,k} \ \leq \ t_{j,h} \ + \ (1 \ - \ x_{i,j,h,k}) \ \cdot \ L \quad (\forall\ i,\ j,\ h,\ k); \tag{4.5}$$

$$Tm_{i,k} + (1 - x_{i,j,h,k}) \cdot L \geq t_{j,h} \quad (\forall\, i,\ j,\ h;\ k); \tag{4.6}$$

$$y_{i,j,h} \leq a_{i,j,h} \quad (\forall\, i,\ j,\ h); \tag{4.7}$$

$$\sum_{j} \sum_{h} x_{i,j,h,k} = 1 \quad (\forall\, i,\ k); \tag{4.8}$$

$$\sum_{i} y_{i,j,h} = 1 \quad (\forall\, j,\ h); \tag{4.9}$$

$$\sum_{k} x_{i,j,h,k} = y_{i,j,h} \quad (\forall\, i,\ j,\ h); \tag{4.10}$$

$$t_{j,h} \geq 0 \quad (\forall\, j,\ h); \tag{4.11}$$

$$Ps_{j,h} \geq 0 \quad (\forall\, j,\ h); \tag{4.12}$$

$$Tm_{i,k} \geq 0 \quad (\forall\, i,\ k); \tag{4.13}$$

$$x_{i,j,h,k} \in \{0,1\} \quad (\forall\, i,\ j,\ h,\ k); \tag{4.14}$$

$$y_{i,j,h} \in \{0,1\} \quad (\forall\, i,\ j,\ h); \tag{4.15}$$

The minimization objective is the makespan (Eq. 4.1). Constraint Eq. 4.2 determines the processing time of operation $O_{j,h}$ done on machine i. Constraint Eq. 4.3 ensures that each job will follow the specified processing order of operations. Constraint Eq. 4.4 ensures that each machine will process only one operation simultaneously. Constraint Eq. 4.5 and Eq. 4.6 ensure that each operation $O_{j,h}$ is allowed to be processed when the previous operation $O_{j,h-1}$ is completed. Constraint Eq. 4.7 declares suitable machines for each operation. Constraint Eq. 4.8 assigns the operations to a machine. Constraints Eq. 4.9 and Eq. 4.10 ensure that all operations will be performed only on one machine with a certain priority. Constraint Eq. 4.11 describes that the start time of all operations is greater or equal to zero, which means that all operations are prepared immediately and can be started immediately. Constraint Eq. 4.12 describes that processing time of all operations is not a negative value. Constraint Eq. 4.13 describes that the operation which is assigned to a machine can start immediately, if the machine is in idle time. Constraint Eq. 4.14 describes that $O_{j,h}$ is performed on machine i with priority k and Eq. 4.15 describes that machine i is selected for this operation.

4.2 The Extended Job-Shop Scheduling Model

The JSS model described in Section 4.1 has been extended for the purposes of this work with respect to the warehousing environments as follows. The problem has e employees, m machines and n jobs. Each job consists of a sequence of operations $O_{j,h}$, $h = 1, \ldots, l$, where $O_{j,h}$ stands for the h-th operation of job j and l stands for the number of operations required for job j.

Since we are not talking about automatic production lines, but the warehouses and distribution centers, the labor in the warehouse environment has the same importance as the warehouse equipment or machines. The set of employees is noted $E, E = E_1, \ldots, E_o$. The specific employee is indexed by e. The set of jobs is noted $J, J = J_1, \ldots, J_n$. The specific job is indexed by j and the specific operation of job is indexed by h. The specific operation of the job noted as $O_{j,h}$ always requires one employee of suitable employees $E_{j,h}$ out of an employee set E ($E_{j,h} \subset E$), which are together described by processing time $P_{e,j,h}$. Some operations require one machine of suitable machines $M_{j,h}$ out of a machine set M ($M_{j,h} \subset M$), which could be described by processing time $P_{i,j,h}$. Since not every employee has authorization for all equipment, the suitable set of machines for employee is stated as $M_{e,j,h}$ out of suitable machines for operation $M_{j,h}$ ($M_{e,j,h} \subset M_{j,h} \subset M$). The subset $M_{j,h}$ is defined by $a_{i,j,h}$ and the subset Ej, h is defined by $b_{e,j,h}$. The operation can be performed on the machine i only if the employee e has an authorization for the specific machine.

Tab. 4.3: The extended notation of the Flexible JSS problem.

e	index of employee, E_e denotes employee $e, e = 1, \ldots, o$
o	number of employees which can be used for job processing

Tab. 4.4: The extended parameters of the Flexible JSS problem.

$b_{e,j,h}$	describes the capable employee set $E_{j,h}$ assigned to operation $O_{j,h}$
$p_{e,i,j,h}$	processing time of operation $O_{j,h}$ performed on machine M_i by employee E_e; $p_{e,i,j,h} > 0$, this replaces the original $p_{i,j,h}$
$Ps_{j,h}$	changes to processing time of operation $O_{j,h}$ after selecting a machine and an employee, this notation definition is also changed
$C_{i,j,h}$	the completion time of operation $O_{j,h}$ is not known a priori
C_j	the completion time of job i is not known a priori
D_j	the due date for job j is given a priori
$\bar{D}_j$	the deadline for job j should also be given a priori

$$b_{e,j,h} = \begin{cases} 1 & \text{If } O_{j,h} \text{ can be performed by employee } e, \\ 0 & \text{Otherwise,} \end{cases}$$

$$c_{e,i} = \begin{cases} 1 & \text{If employee } e \text{ is authorized for machine } i, \\ 0 & \text{Otherwise,} \end{cases}$$

$$z_{e,j,h} = \begin{cases} 1 & \text{If employee } e \text{ is selected for operation } O_{j,h}, \\ 0 & \text{Otherwise,} \end{cases}$$

In the warehouse environment, performing an operation on machine i by authorized employee e has no setup time and also no setup cost. So, that is the advantage of the warehouse environment. The batch processing, which is the performing of the same operations on the set of jobs, is an advantage only when the next job is located in the a distant part of the warehouse and the employee with a truck should relocate to that position.

The constraints are extended as follows:

$$z_{e,j,h} \leq b_{e,j,h} \quad (\forall\, e,\ j,\ h); \tag{4.16}$$

$$c_{e,i} = 1 \quad (\forall\, e,\ i); \tag{4.17}$$

$$y_{i,j,h} \cdot z_{e,j,h} = c_{e,i} \quad (\forall\, e,\ i,\ j,\ h); \tag{4.18}$$

$$\sum_{e} z_{e,j,h} = 1 \quad (\forall\, j,\ h); \tag{4.19}$$

$$D_j \geq C_j \quad (\forall\, j); \tag{4.20}$$

$$\bar{D}_j \geq D_j \quad (\forall\, j); \tag{4.21}$$

$$z_{e,j,h} \in \{0, 1\} \quad (\forall\, e,\ j,\ h); \tag{4.22}$$

Constraint Eq. 4.16 declares a suitable employee for each operation. Constraint Eq. 4.17 ensures that employee e is authorized for machine i. Constraint Eq. 4.18 ensures that the selected employee is authorized for the selected machine. Constraint Eq. 4.19 ensures that all operations will be performed only by one employee. Constraint Eq. 4.20 ensures that the due date is less or equal to the completion time of the specific job. Constraint Eq. 4.21 ensures that the deadline for job j has to be fulfilled, and the Eq. 4.22 describes the parameter interval of the selected employee for operation $O_{h,j}$.

5 THE COLLISION PREDICTION ALGORITHM

This chapter is based on the paper [312] and deals with investigation of collisions of particles and their exploitation in the real-world application which is in the context of this thesis the logistic warehouse environment. The collisions of particles are important physical phenomena met by people every day, e.g. billiards, racket striking ball (ping-pong, tennis, squash, ricochet, and others), golf, car accident etc. Generally, a collision is an isolated process in which two or more moving particles exert their forces on each other over a relatively short period of time.

Collision is a phenomenon, limited in time and space, in which two or more objects mutually affect each other [298], [299]. During the mutual affection of objects the redistribution of *momentum* (p) and *kinetic energy* (E_k) dawn in the system. There are two basic types of collisions: *elastic* and *inelastic* (*plastic*). The collision of two particles is called *binary collision* and the collision of more particles is called *multi-particle collision*. If the particles are physically in contact it is a *near collision*, e.g. billiards, if they are not in physical contact it is a *distant collision* which is represented by a gravitational force, magnetic force, electric force, e.g. the earth circulates around the sun. Since the perfect conditions are impossible to reach in a real-world, the simulations are done in an isolated system. The isolated system respects the law of conservation of momentum and kinetic energy, and also the internal structure of particles. The elastic collisions conserve both, the momentum and the kinetic energy, while inelastic collisions conserve only the momentum but not the kinetic energy. The collisions are categorized based on the conservation of E_k.

There is a plenty of fork-lift trucks in the logistic environments, manipulating simultaneously with homogenous or heterogeneous pallets between shelves. When the trucks are crossing the aisles and paths of other trucks the collisions may occur. In consequence of that, the congestion and blocking of aisles in the warehouse may arise, which spins out the time of job completion of particular workers and decreases the productivity of warehouse or worse, it leads to complete cut-off of the warehouse work-flow. The main idea of this chapter is to propose an approach of how to avoid congestions, blocking, and possible financial losses by predicting potential collisions of trucks and to give a notice to warehouse operator who can prepare an appropriate reaction, e.g. to send one truck by a different path, to make a time-window for one truck and let it to finish its job before the second truck will come etc.

Section 5.1 describes physical basics, which are elastic and inelastic collisions [300]. The applied example of collision of two objects in a 2-dimensional environment is described in section 5.2. Section 5.3 describes types of fork-lift truck collisions which can occur. Section 5.4 describes an example of collision detection and the design of algorithm is described in section 5.5

5.1 Elastic and Inelastic Collisions

Suppose two scenarios of elastic collisions in a 1-dimensional environment. In the first scenario, two objects are moving on a frictionless surface in the same direction at an initial velocity. The velocity of object o_1 is higher then that of object o_2, see Fig. 5.1. In the second scenario only object o_1 is moving towards object o_2 which is at rest.

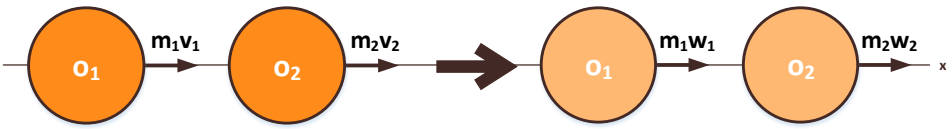


Fig. 5.1: A collision in a 1-dimensional environment.

In Fig. 5.1 m stands for the *mass* of an object, v stands for the *velocity* of an object before collision, w stands for the velocity of an object after collision, and $p = m \times v$ is the *momentum*. The objects are distinguished by subscripts. Since it is an elastic collision, the total kinetic energy of system E_k before and after the collision must be equal, because it is conserved due to the law of conservation of momentum and kinetic energy, which is shown in Eq. 5.1.

$$\begin{aligned} E_{k1} + E_{k2} &= E'_{k1} + E'_{k2} \ , \\ \frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 &= \frac{1}{2}m_1w_1^2 + \frac{1}{2}m_2w_2^2 \ . \end{aligned} \tag{5.1}$$

The p_1 and p_2 of objects can change after collision, but according to the law of conservation of momentum, the momentum of the system is conserved, which is shown in Eq. 5.2:

$$\begin{aligned} p_1 + p_2 &= p'_1 + p'_2 \ , \\ m_1v_1 + m_2v_2 &= m_1w_1 + m_2w_2 \ . \end{aligned} \tag{5.2}$$

A system of Eqs. 5.3 is obtained by adjustment of Eq. 5.1 and Eq. 5.2:

$$\begin{aligned} m_1(v_1^2 - w_1^2) &= m_2(w_2^2 - v_2^2) \ , \\ m_1(v_1 - w_1) &= m_2(w_2 - v_2) \ . \end{aligned} \tag{5.3}$$

The Eq. 5.3 helps to formulate particular velocities of objects after collision:

$$\begin{aligned} v_1 + w_1 &= v_2 + w_2 \ , \\ w_1 &= v_2 + w_2 - v_1 \ , \\ w_2 &= v_1 + w_1 - v_2 \ , \end{aligned} \tag{5.4}$$

and final velocities are determined by combination of Eq. 5.2 and Eq. 5.4:

$$\begin{aligned} w_1 &= \frac{(m_1 - m_2)v_1 + 2m_2v_2}{m_1 + m_2}\,, \\ w_2 &= \frac{(m_2 - m_1)v_2 + 2m_1v_1}{m_1 + m_2}\,. \end{aligned} \tag{5.5}$$

The Eq. 5.5 discovers one interesting fact. If the mass of both objects is of the same value, the collision simply switches the velocities of objects. Likewise, if one of objects is not moving, the moving object passes its velocity to the standing object and stops. The degree of elasticity is quantified by the *coefficient of restitution*, which is the ratio of velocities after and before an impact, see Eq. 5.6:

$$k = \frac{w_2 - w_1}{v_1 - v_2}\,. \tag{5.6}$$

If the collision is a perfect elastic collision, the value of coefficient of restitution is 1, while the value 0 represents a state when a hitting object stops at collision. When the elastic collision is not perfect, the coefficient of restitution differs in dependence on material and it is also dependent on the velocities of colliding objects. When the velocities are high, the deformation of objects is considerable and the coefficient of restitution decreases.

Now, suppose an elastic collision in a 2-dimensional environment. The first object is moving and the second object is at rest. Suppose, further, that it is not a head-on collision, and after the collision object o_1 is moving off at an angle θ_1 and object o_2 at an angle θ_2. In this example, the total momentum must be considered as a vector quantity, since the motion is 2-dimensional, see Fig. 5.2.

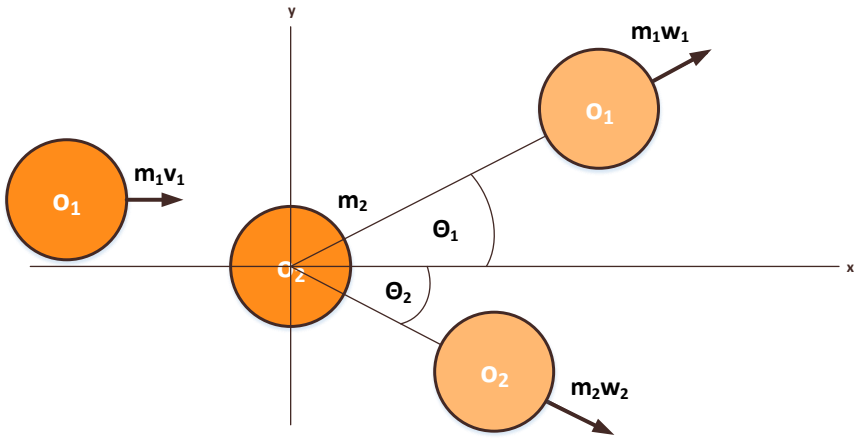


Fig. 5.2: A collision in a 2-dimensional environment.

The collision environment takes place within the $x - y$ plane, so the total momentum before and after the collision must be considered:

$$\begin{aligned} m_1 v_1 &= m_1 w_1 cos(\theta_1) + m_2 w_2 cos(\theta_2) \ , \\ m_1 w_1 sin(\theta_1) &= m_2 w_2 sin(\theta_2) \ . \end{aligned} \tag{5.7}$$

When this is a case of an elastic collision, the total kinetic energy of both objects before and after the collision is:

$$\frac{1}{2} m_1 v_1^2 = \frac{1}{2} m_1 w_1^2 + \frac{1}{2} m_2 w_2^2 \ . \tag{5.8}$$

In case that both objects are moving, the system of equations is following:

$$\begin{aligned} m_1 v_1 cos(\theta_1) + m_2 v_2 cos(\theta_2) &= m_1 w_1 cos(\Theta_1) + m_2 w_2 cos(\Theta_2) \ , \\ -m_1 v_1 sin(\theta_1) + m_2 v_2 sin(\theta_2) &= m_2 w_1 sin(\Theta_1) - m_2 w_2 sin(\Theta_2) \ , \end{aligned} \tag{5.9}$$

and for the case of an elastic collision, the total kinetic energy of both objects before and after the collision is Eq. 5.1.

In the following text an inelastic collision will be considered, because most of collisions occurring in the real world are not elastic. A certain fraction of the initial kinetic energy of the colliding objects is ordinarily transformed into some other kind of energy, e.g. heat energy in case of the ball games, or mechanical deformation in case of a truck accident. These collisions are called inelastic. In general, the presented equations are valid also for inelastic collisions except the equations of kinetic energy. The totally inelastic collision causes that the objects stick together and their velocity after the collision is equal, $w_1 = w_2$. In this case, the Eq. 5.2 is reduced and the final velocity of the stuck objects is:

$$w = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2} \ . \tag{5.10}$$

In other words, the common final velocity of the two objects is directly equal to the center of mass velocity of the entire system. Furthermore, suppose that the second object is initially at rest (i.e., $v_2 = 0$). In this special case, the common final velocity of the two objects is:

$$w = \frac{m_1}{m_1 + m_2} v_1 \ . \tag{5.11}$$

Note that the hitting object is slowed down by the collision. The fractional loss in the kinetic energy of the system due to the collision is given by:

$$f = \frac{m_1 v_1^2 - (m_1 + m_2) w^2}{m_1 v_1^2} \ . \tag{5.12}$$

The loss in E_k is small if the stationary object is much more lighter than the hitting one (i.e., if $m_2 \ll m_1$), and almost 100 % if the hitting object is much lighter than the stationary one (i.e., if $m_2 \gg m_1$). Of course, the lost E_k of the system is transformed into some other form of the energy mentioned above.

5.2 Collision of 2 Objects in 2 Dimensions

At the start, the positions and velocities of two objects o_1 and o_2 are given at time t. The quest is to determine if and when they will collide with each other.

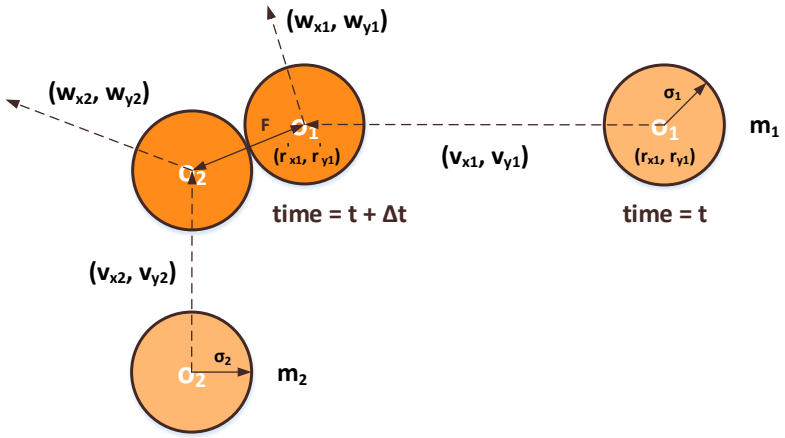


Fig. 5.3: Another collision in a 2-dimensional environment.

The situation is depicted in Fig. 5.3, where (r_{x1}, r_{y1}) and (r_{x2}, r_{y2}) stand for the positions of objects o_1 and o_2 at the moment of collision $t + \delta t$. When the objects collide, the centers of objects are distant $\sigma = \sigma_1 + \sigma_2$, which is:

$$\sigma^2 = (r'_{x1} - r'_{x2})^2 + (r'_{y1} - r'_{y2})^2 \ . \tag{5.13}$$

Before the collision, the objects were moving on straight-line trajectories with constant velocities. Thus,

$$\begin{aligned} r'_{x1} &= r_{x1} + \Delta t \times v_{x1} \ , \\ r'_{y1} &= r_{y1} + \Delta t \times v_{y1} \ , \\ r'_{x2} &= r_{x2} + \Delta t \times v_{x2} \ , \\ r'_{y2} &= r_{y2} + \Delta t \times v_{y2} \ . \end{aligned} \tag{5.14}$$

Substituting Eq. 5.14 with Eq. 5.13 gives a quadratic equation for Δt. Selecting the relevant root and simplifying the expression for Δt in terms of the known positions, velocities, and radii gives the Eq.5.15:

$$\Delta t = \begin{cases} \infty & \text{if } \Delta v \times \Delta r \geq 0, \\ \infty & \text{if } d < 0, \\ -\frac{\Delta v \times \Delta r + \sqrt{d}}{\Delta v \times \Delta v} & \text{otherwise,} \end{cases} \tag{5.15}$$

where

$$\begin{aligned} d &= (\Delta v \times \Delta r)^2 - (\Delta v \times \Delta v) \times (\Delta r \times \Delta r - \sigma^2) \ , \\ \Delta r &= (\Delta r_x, \Delta r_y) = (r_{x2} - r_{x1}, r_{y2} - r_{y1}) \ , \\ \Delta v &= (\Delta v_x, \Delta v_y) = (v_{x2} - v_{x1}, v_{y2} - v_{y1}) \ , \\ \Delta r \times \Delta r &= (\Delta r_x)^2 + (\Delta r_y)^2 \ , \\ \Delta v \times \Delta v &= (\Delta v_x)^2 + (\Delta v_y)^2 \ , \\ \Delta v \times \Delta r &= (\Delta v_x) \times (\Delta r_x) + (\Delta v_y) \times (\Delta r_y). \end{aligned} \tag{5.16}$$

According to the equations depicted above, the $\Delta t \geq 0$, only if $\Delta v \times \Delta r \geq 0$ or $d < 0$ the quadratic equation has no solution for $\Delta t > 0$.

5.3 Types of Fork-lift Truck Collisions

This section describes the collision situations which can occur between two fork-lift trucks. The multiple-truck collisions are not considered, because they are always simplified into particular collisions between two trucks. The typical warehouse layout is of a rectangular shape. Only two dimensions are considered, so the layout is represented by a 2D matrix called the cellular model. The path of each truck is described by particular moves (i.e. consecutive cells in a model). Consequently, the current position of the truck is given by the coordinates in the cellular model of the warehouse. The truck is able to move in 4 possible ways represented by the von Neumann neighborhood (i.e. left, right, up, down, and null). The null value means that the truck is not moving anywhere or is already on the target cell. The cellular model of the warehouse is limited by the cells marked as racks and walls.

In the case of collision prediction, the direction of movement is used especially for the determination of a *threshold*. The threshold represents the percentage of occupancy of cell by truck when the collision is detected on the cell. The threshold is a number from the bounded interval $< 0, 1 >$ usually expressed in percentage. It can be decided if there is a possibility to avoid a collision without any intervention

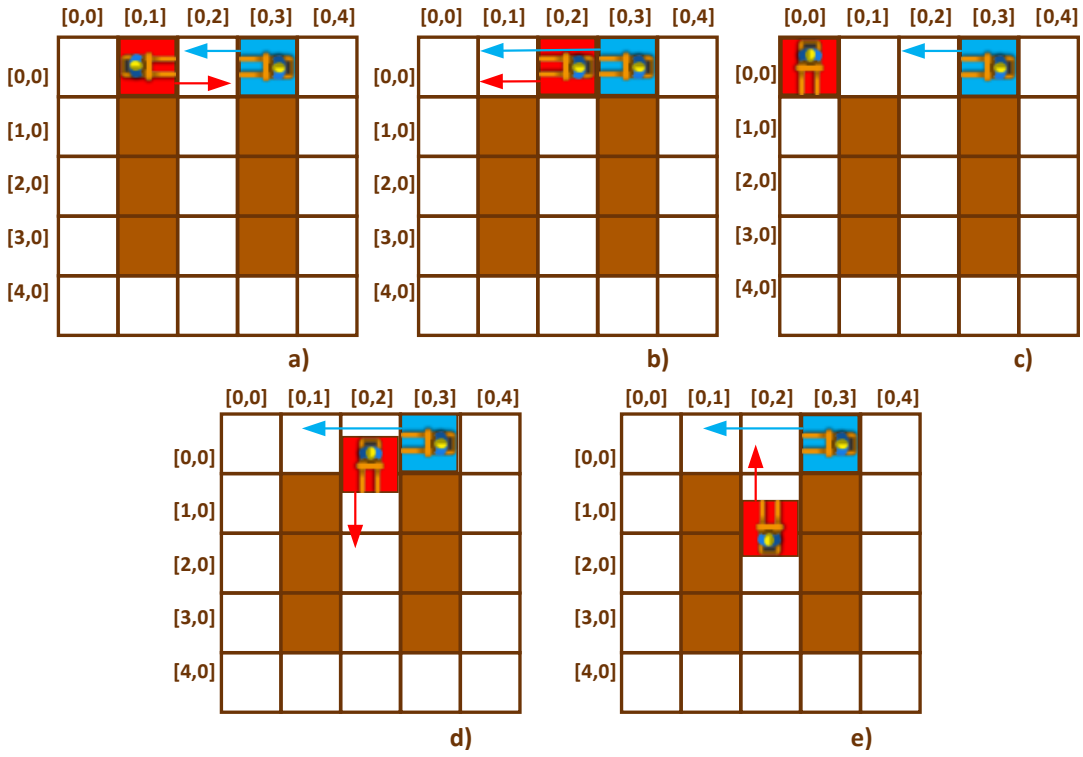


Fig. 5.4: Types of collisions in the warehouse environment.

of an operator with the help of threshold. Of course, it does not mean that the high value of threshold avoids the real collision. The threshold must be set carefully due to the size of cells and trucks. Note that the warehouse has two types of cells, *wide aisles* and *narrow aisles*. While the wide aisles are located around the perimeter of the warehouse and the trucks have a possibility to get out of one's way, it is not possible in the narrow aisles between the racks.

Collisions can be divided into two basic categories. The first category is the collision in *a straight direction*, depicted in Fig. 5.4. The trucks in this category are moving in the opposite directions (see Fig. 5.4a), in the same direction (see Fig. 5.4b), or only one truck is moving (see Fig. 5.4c). The second category represents the collisions in *a perpendicular direction*. Trucks in this category collide in the right angle, either indirectly, i.e. the first truck is leaving the cell of collision but it is not quick enough and the second truck hits the first truck, (see Fig. 5.4d), or directly, i.e. both trucks are moving towards the cell of collision (see Fig. 5.4e). Off course, the other parameters such as the type of aisle (wide aisle or narrow aisle) have to be considered.

5.4 A Numerical Example of Truck Collision

The following text describes a numerical example of collision prediction. Fig. 5.5 shows an indirect perpendicular collision. More particularly, truck o_1 (colored blue) is moving from cell $[4, 1]$ to the cell $[2, 2]$ and truck o_2 (colored red) is moving from cell $[0, 0]$ to cell $[2, 1]$. This example describes the situation where truck o_1 hits the back of truck o_2. The part of truck o_2 on cell [2,0] is 75 %, which is higher than the established threshold 20 % in this case. The complete paths of both trucks, cell by cell, are depicted in Tab. 5.1.

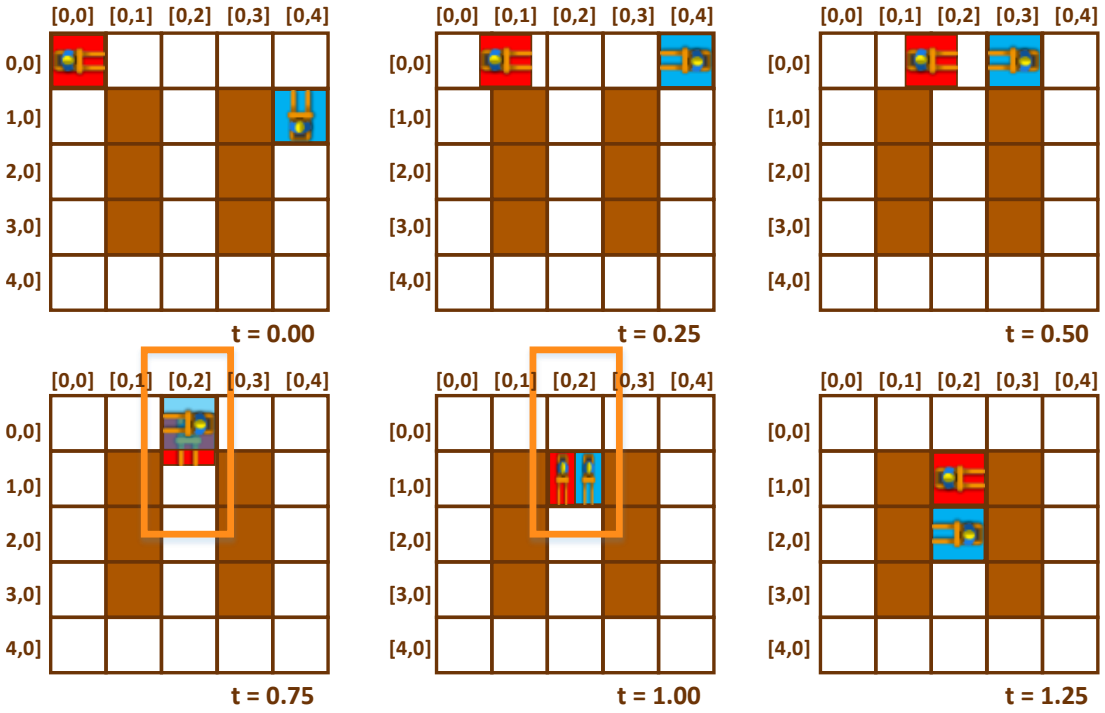


Fig. 5.5: A numerical example of the collision detection.

Velocities of trucks are important variables of collisions prediction. Generally, if the velocity of one truck (e.g. o_1 in this case) is very high and the velocity of the second truck (o_2) is smaller, the collision can be avoided. The collision time is calculated from the exact paths of both trucks and their velocities, both related to the type of cell where the collision is going to occur (i.e. wide or narrow aisles). The path is represented by the cells which are passed by trucks in the cellular model.

The problem described in Tab. 5.1 is depicted in Fig. 5.5. In the first step the cells containing both trucks in the same time (i.e. the collision cells) must be calculated. This is not so obvious from Tab. 5.1, but these are cells [2,0] and [2,1] depicted in Fig. 5.5 in orange color. The process of collision calculation is present in the next paragraph.

Tab. 5.1: The coordinates & paths of truck o_1 and truck o_2

Index of Step	0	1	2	3	4	5
Coordinates of truck o_1	$[4,1]$	$[4,0]$	$[3,0]$	$[2,0]$	$[2,1]$	$[2,2]$
Percentage of truck o_1	100 %	100 %	100 %	100 %	100 %	100 %
Coordinates of truck o_2	$[0,0]$	$[1,0]$	$[2,0]$	$[2,1]$	$[2,1]$	$[2,1]$
Percentage of truck o_2	100 %	75 %	50 %	25 %	100 %	100 %
Steps of truck o_1	NULL	UP	LEFT	LEFT	DOWN	DOWN
Steps of truck o_2	NULL	RIGHT	RIGHT	DOWN	NULL	NULL

The velocity of truck o_1 is $v_1 = 4\ ms^{-1}$ and velocity of truck o_2 is $v_2 = 3\ ms^{-1}$. The distance which both trucks have to overcome is for truck o_1 equal to $s_1 = 5\ cells$ and for truck o_2 equal to $s_2 = 3\ cells$. With these data t_1 and t_2 can be simply determined. Then, the time in which both trucks can pass one cell has to be computed. The results of calculation are depicted in Tab. 5.2. The value s_{ref} describes the size of movement of each truck which is done in the time of one time step of the fastest truck expressed in percents, e.g. there are only two trucks, so o_1 is the truck with the highest velocity, so $s_{1ref} = 100\ \%$ – every step is one whole cell, obviously. Truck o_2 is slower, so in one step it is moving $s_{2ref} = 75\ \%$ of the cell. For better understanding, everything is shown in Fig. 5.5. And this is the process how the collisions are detected. Note the narrow aisle constraint, so the collision occurs only in $t = 1.00\ s$.

Tab. 5.2: A numerical computation of the collision prediction.

	v	s	t	t_{cell}	s_{ref}
Truck o_1	4	5	1.25	0.25	100 %
Truck o_2	3	3	1.00	0.33	75 %

5.5 The Design of Collision Prediction Algorithm

The previous section described the numerical computation of collision cell and collision time. To make a solution complete, it is necessary to know the position (v_x, v_y) and the velocity (r_x, r_y) of trucks. Furthermore, it is necessary to know if there is any manipulation with pallet (loading, unloading, storing etc.) on the cell or not, because this manipulation also consumes some amount of time. For this reason, a *flag* which carries the information about the type of manipulation is added. Then, the flag tells what type of manipulation is being done and how many time units it will take. In the following text the design of the collision prediction algorithm will

be presented and shown in the class diagram, see Fig. 5.6. Only the most important parameters and methods are depicted in the class diagram. The description of partial classes is present below the image.

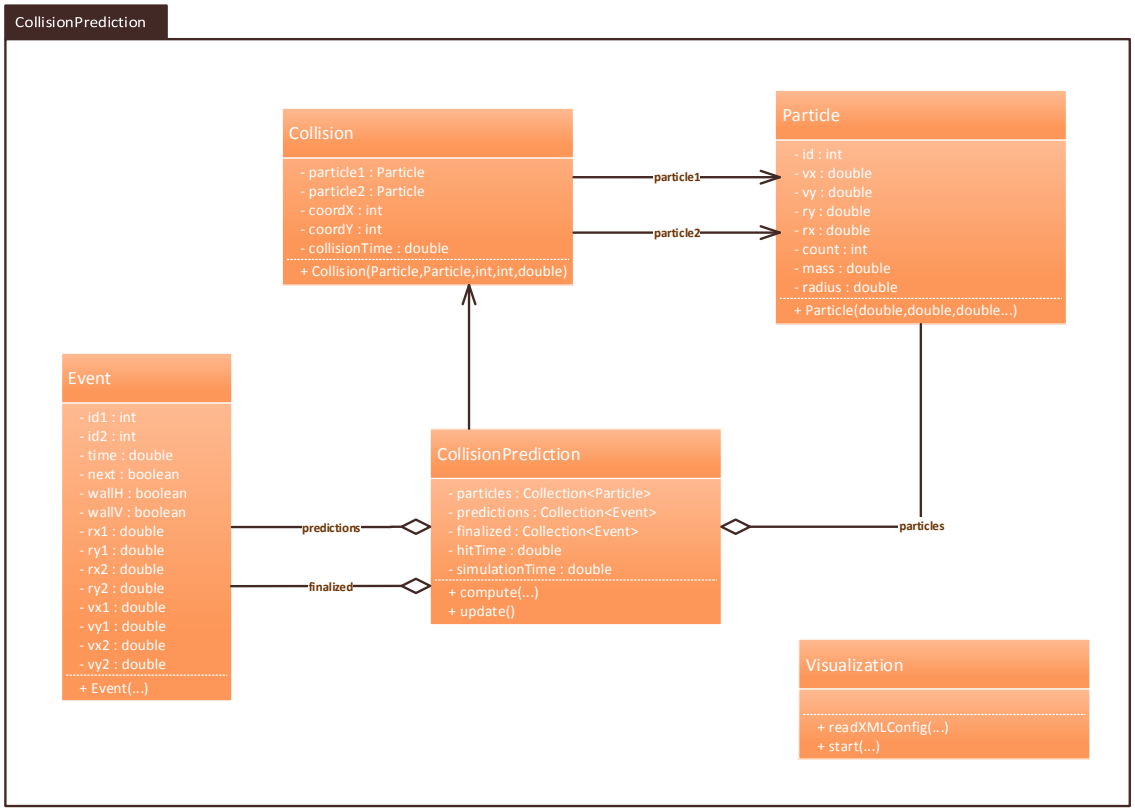


Fig. 5.6: The simplified class diagram of the collision detection algorithm.

The class *Particle* represents each object which can be a possible element of collision. This class contains parameters which describe each particle, such as the identification (*id*), the coordinates (*vx, vy*), the velocity (*rx, ry*) in the directions x and y, the count of collisions with other particles (*count*), the mass of the particle (*mass*), and the radius of the last step (*radius*).

The class *Collision* represents each collision which can occur in simulation. The collision always consists only of two particles (*particle1, particle2*), the coordinates of the cell where the collision has been detected (*coordX, coordY*), and of course the time of collision (*collisionTime*).

The class *CollisionPrediction* computes collisions and the output is the XML file describing all collisions in a given simulation time interval. The class references the class *Particle*, because the system has to be aware of all particles which are present. The class also references the class *Collision* because it has to be aware of the time when the collision occured, the collisions are computed by the method *compute()*. The method *compute()* is used only to detect the first collision in a time. Other

collisions are irrelevant, because the direction and the velocity of objects after the first collision change and can significantly affect other particles and collisions. Then, the *update()* method is called, which recomputes new positions of all objects in the environment till the next collision. After the update of variables the computation continues until the simulation time interval is exceeded. When the simulation is done, the log of collisions is created with the help of *Event* class.

The class *Event* represents the collision prediction logger. The class contains the information about the object identifiers, the time of collision, the velocity of objects in both directions of axes, and of course the coordinates of objects. Other information is about the collision with a horizontal and a vertical wall, which is actually not in the warehouse implementation. If both of these variables are set to *false*, it is a collision of two objects, otherwise only one identifier of object is set. And the last variable denotes if there are more collisions in the same time. The class *Visualization* is only a supportive class for visual validation of the collision prediction algorithm.

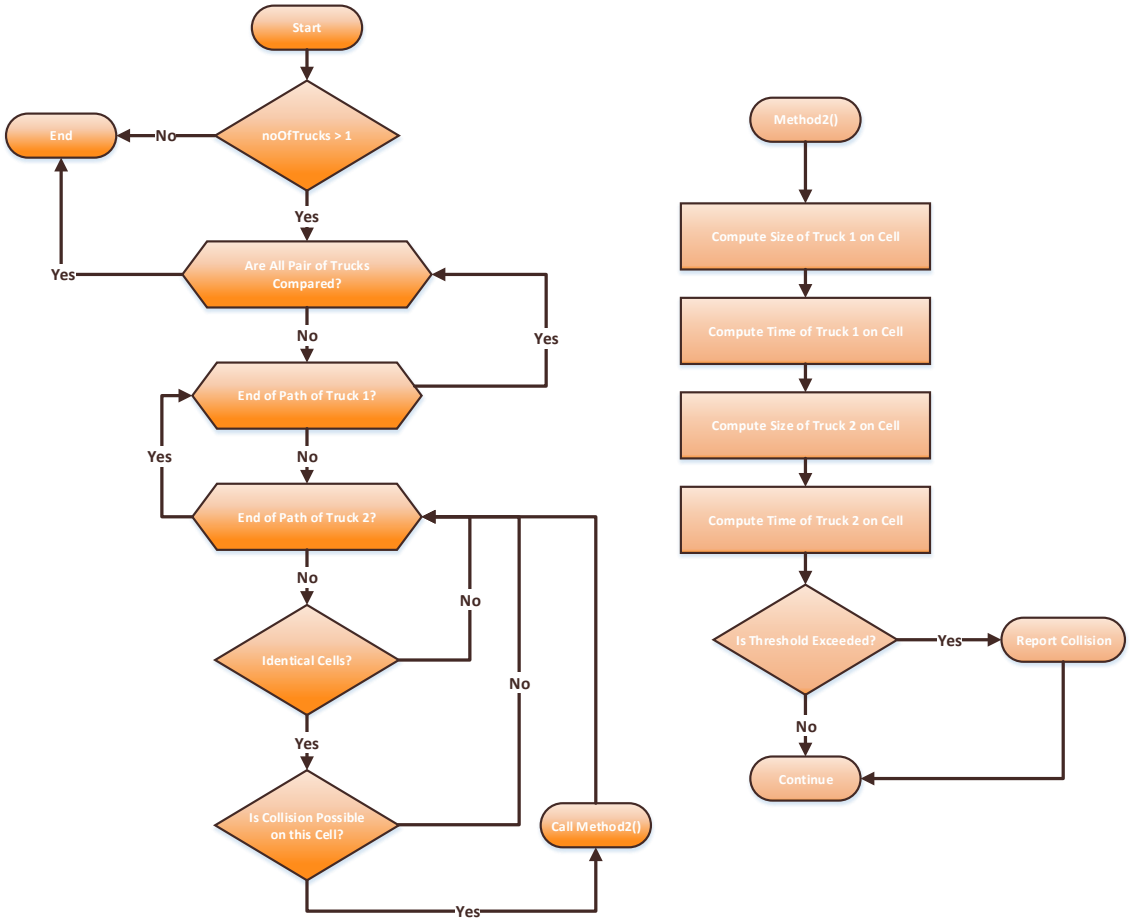


Fig. 5.7: A block scheme of the collision detection algorithm.

Two main methods, described in details in the further text, are used for the collision prediction and the computation of collision time. The first method is for the detection of the same cells in the paths of trucks. The method considers how many trucks are in the simulation, because if there is only one truck, no collision can be predicted and the process ends. If there are more trucks, two local variables $aTime$ and $bTime$ for each pair of trucks are created. These variables are used sums of times during which the trucks are moving through the warehouse.

Afterwards, the prediction algorithm runs two loops in which all trucks are checked with each other. Then inside the loops there are other two loops where the paths of both compared trucks are checked cell by cell for the identical cells which they are passing through. If there are identical cells, the second method is called which decides if there is a collision on specific cell or not. Furthermore, the $bTime$ is incremented by the time quanta spent on the specific manipulation if the flag is set. After all steps the $aTime$ is incremented if the flag has been set.

The second method implements the detection and computation of collision of two trucks on the same cell. The input of this method includes two objects describing the parameters of trucks. The variables a and b denote the indexes of path steps of both trucks, the variables $aTime$ and $bTime$ describe the incremental time of truck movement. The *Threshold* represents a threshold of cell, the $hitTime$ denotes a collision time, and m_1 and m_2 are used for the direction of movement of trucks on the collision cell.

6 THE EVOLUTIONARY FRAMEWORK

This chapter describes the design of the Evolutionary Framework which has been developed for the purposes of several research projects including the project related to this thesis. At first, a brief introduction and the motivation of development of a new framework are described. The main motivation to develop the new framework is that the most of existing frameworks consist of e.g. outdated and unmaintained algorithms, algorithms which cannot be modified for specific problems and on the other hand, algorithms too specific without possibility of generalization, with a lack of documentation, a lack of suitable examples and use cases, with unsuitable license under which the frameworks are distributed and so on. Also the programming language in which the framework is developed can be unsuitable, which can be a significant problem for integration to some projects.

The framework was developed also with the intent to unify the use of optimization algorithms and to use only one tool inside our research group. This should help to develop a single project implemented in the Java programming language which could serve as a shared code library and to help increase a cooperation among researchers in laboratory. Everyone can contribute by their own algorithms, which should enable to improve and extend the framework, and of course to use different algorithms for each problem and make them easily comparable to everyone. With such framework each researcher does not have to implement the algorithms from scratch. Another reason for creating the new optimization framework was to create the library which can be distributed under the friendly license in the scientific community as well as in the business community. The framework is distributed under the GNU Library General Public License (LGPL, also called Lesser GPL) which is a compromise between the strong-copyleft GNU GPL and permissive licenses. The result is that the software published under the LGPL can be linked with (used by) a non-(L)GPL program which has a copyrighted source code. The framework is available for free to anyone and it is an open source software product.

The rest of the chapter is structured as follows. Section 6.1 describes the design of the framework and the design of the GP module with focus on GP driven by the CFG. In this section the simplified class structure with an appropriate description is presented. Section 6.2.1 describes the initialization method, and the genetic operators such as crossover and mutation, also driven by the Context-free Grammar are described in section 6.2.2 and section 6.2.3. Section 6.3 describes the first use case example which is a design of non-cryptographic hash function, section 6.4 deals with a method for localization of Common Carotid Artery (CCA) in the transverse section of B-mode ultrasound images for medical purposes.

6.1 The Design of Framework Architecture

It is not an easy task to design a flexible, modular, and robust framework architecture and meet all the demands including the definition of optimization problems and desired parameters of GP algorithm. The framework is based on layered architecture which is easily extensible for future development and its architecture is described in the following text. The framework architecture is shown in Fig. 6.1 – Fig. 6.1.

The first layer contains classes that are mentioned to be a common ground for all algorithms based on the evolutionary computation and should be helpful and essential for particular implementations of such algorithms. This layer contains tools for random number generation, statistics of program run, configuration parameters etc. This functionality is used by classes in higher layers. The second layer consist of basic classes which serve as models or templates for particular implementations of algorithms. This layer includes common interfaces, configurations of algorithms, and prescriptions of production classes representing population, algorithms and evolution process. The third layer represents particular extensive modules, e.g. the GP module presented in these further sections, and its particular production classes.

The main class is the class *Program* which starts the optimization process. The class *Config* contains common important configuration parameters for all optimization algorithms. The other first layer classes are the class *Logger* for logging of all actions which come up in the framework, the class *RandomGenerator* which represents a random number generator, the interface *IFitnessEvaluator* which gives the prescription of how to specify the evaluation algorithms for every specific problem, and the interface *IEvolutionSpecifier* which prescribes the methods which initialize the population of individuals and evolves each generation during evolution process.

The second layer classes contain also some basic implementations of how the evolution process should look like. The *Population* class contains basic operations over the whole population and a set of chromosomes of the class type *Chromosome* which gives a general template for all the chromosome types. Next important element is the class *EvolutionSpecifierAdapter* which implements an interface *IEvolutionSpecifier* and tells how the main concept of sequence of genetic operators should be started. The class *DefaultEvolutionSpecifier* is inherited from the class *EvolutionSpecifierAdapter* and represents a particular default implementation of how the population should be processed in a single evolution step. Another very important class is the class *Grammar* which represents a definition of CFG. The grammar is constructed from single rules specified by the class *Rule*. Each rule of the grammar consists of objects constructed by the class *ActionTree* which in fact represents a particular implementation of the class *Action* which is actually the action, function or non-terminal symbol performed on a tree node.

The third layer here represents a particular implementation of the GP module. The definition of Context-free Grammar represented by the class *Grammar* was incorporated into the particular implementation of the class *DefaultEvolutionSpecifierTree*. The class *DefaultEvolutionSpecifierTree* is implemented specially for Tree-based GP. The concrete implementation of chromosome was implemented into the class *TreeChromosome* which represents a data structure of Tree-based GP. The tree structure consists of nodes represented by the class *Node* which can construct both a terminal symbol and also a non-terminal symbol.

Important building blocks have to propose the universal representation of candidate solutions, evolutionary operators such as selection, crossover, mutation, and others can take many forms and must meet various criteria and comply with restrictive conditions. Some operators such as selection, can be designed globally for most of optimization techniques, while others such as recombination and mutation operators must be concretized for each kind of evolutionary technique or a certain problem. This section is focused on a framework design and the GP module driven by the CFG. The framework was published in [313].

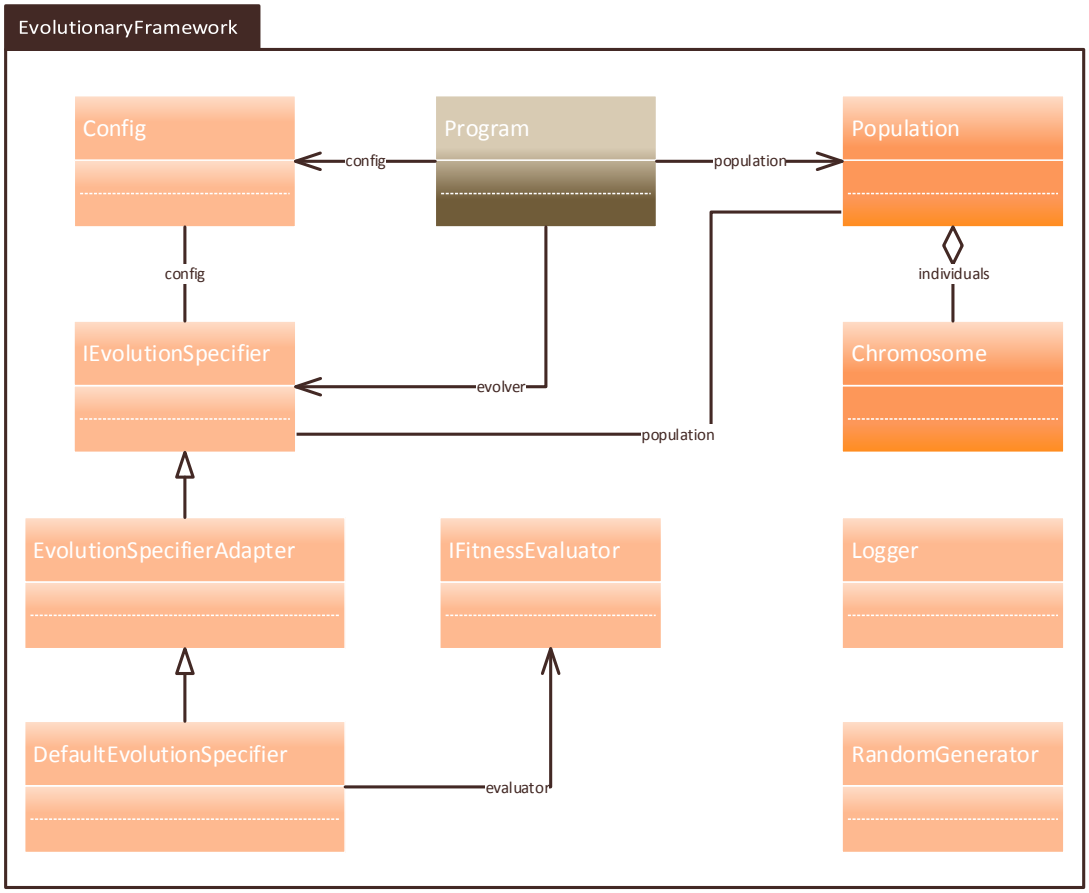


Fig. 6.1: The class diagram of Evolution Framework part I, the first layer.

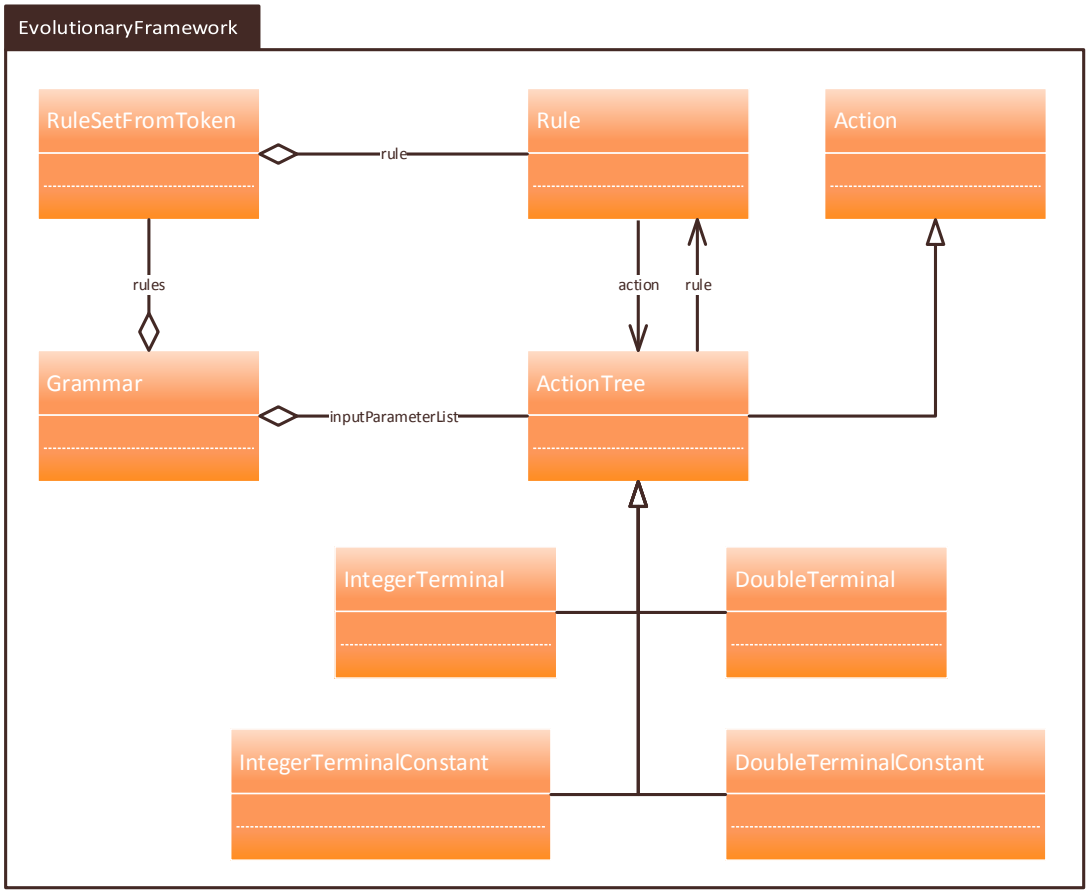


Fig. 6.2: The class diagram of Evolution Framework part II, the second layer.

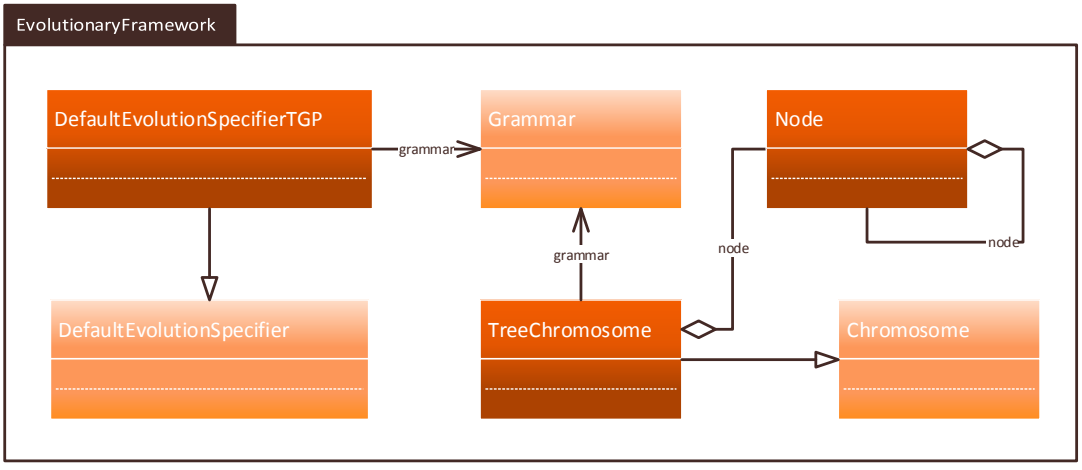


Fig. 6.3: The class diagram of Evolution Framework part III, the third layer.

6.2 Grammar Driven Genetic Programming

The GP module proposed in this section was built on the standard tree-based data structures, proposed by J. R. Koza in [247], and it was supported by CFG. This combination of methods, so called GGGP, was implemented on the basis of inspiration by the concept presented in the papers [267] and [280]. The main goal of employing this technique into the GP module was the fact that the use of grammar simplified significantly the search space, solved the closure problem, and always facilitated generating of valid individuals.

The Context-free Grammar G is defined as a 4-tuple $G = \{\Sigma_N, \Sigma_T, S, P\}$, $\Sigma_N \cap \Sigma_T = \emptyset$, where Σ_N stands for a set of non-terminal symbols, Σ_T stands for a set of terminal symbols, S represents the start symbol of the grammar, and P is the set of production rules written in the Backus-Naur Form. The CFG depicted in Fig. 6.4 is used in this section to clarify the examples and it is inspired by the grammar defined in [267] and [280]. An example of syntactical tree generated with respect to the defined grammar is also depicted in Fig. 6.4.

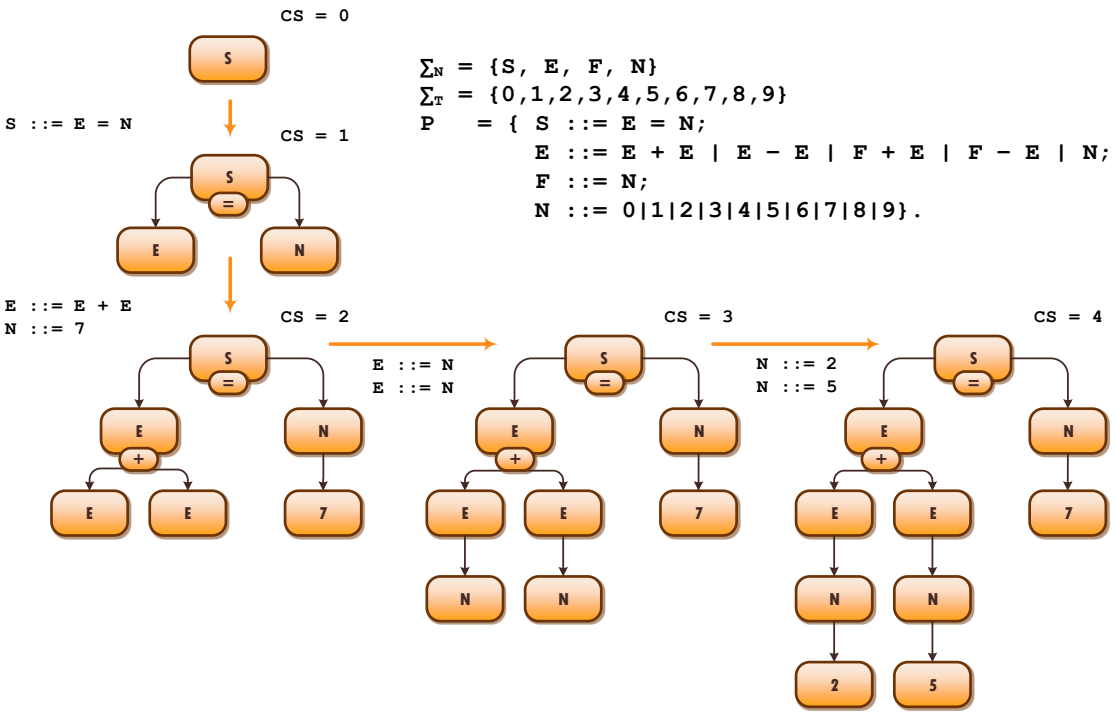


Fig. 6.4: Definition of the Context-free Grammar and an example of the process generating syntactical tree for the sentence $(2 + 5 = 7)$.

Since the CFG was applied and the initialization method was implemented similar to GBIM, the other genetic procedures implemented, such as the standard genetic

sub-tree crossover and the sub-tree mutation, have to be adapted to this new concept as well. The adaptation of the genetic operators was done similarly to the proposed solution by the author of the paper [267]. The crossover was inspired by GBX and the mutation by GBM proposed in [280].

6.2.1 Grammar Driven Initialization Method

The GP module becomes the GGGP module. In the first stage the initialization process has been adapted. The initial population is now generated by the method inspired by GBIM method proposed in [280] which helps to increase the convergence speed when new individuals are generated. This is caused by the included grammar with which the proposed method is able to generate always valid individuals belonging to the search space, because they all represent a candidate solution of the problem. The definition of the grammar establishes formal rules, syntactical restrictions of the chromosome structure of the candidate solutions.

The main difference between the GBIM and the implemented method is that we have narrowed down the implementation for the method only to the most important things connected to initialization. The rest of the code was implemented to the grammar or other code elements. The constructed syntactical tree, candidate solution, in [267] and [280] based on CFG is a little bit different from the proposed syntactical tree in this paper. In the GBIM the non-terminal symbols such as $+$, $-$, $*$, / are handled in the same way as the terminal symbols. In this proposal, the non-terminal symbols are considered as actions (or operations) of an element which produces consequent rules. So, when the production rule $S ::= E = N$ is given, the action of S is an equal sign ($=$) and the elements of this action are the non-terminal symbols E and N, which can generate consequent production rules. The initialization method is described below and it is built on the same four definitions as the GBIM in [267]:

Definition 1 – The length of each terminal symbol is 0; denoted as $a \in \Sigma_T$.

Definition 2 – The length of the production rule that only derives a terminal symbol is 1; $L(A ::= \alpha) = 1$, $\forall A \in \Sigma_N$ and $\forall a \in \Sigma_T$.

Definition 3 – The length of the production rule $A ::= \alpha$ is the result of adding one to the maximal length of the symbol constituting the consequent; $L(A ::= \alpha)$.

Definition 4 – The length of the non-terminal symbol A is the minimal length of all its productions; $L(A)$.

Fig. 6.4 shows the process of generation of the individual which belongs to the proposed grammar in the same figure, taking the maximal depth of tree $D = 5$ as an argument. The grammar computes the necessary information according to Def. 1 – Def. 4 and then the initialization method is applied:

1. The first step is to create a root of syntactical tree and simultaneously propose the information of the maximal depth of this tree.
   - In the example depicted in Fig. 6.4, there is only one rule for the root symbol derivation, which is $S ::= E = N$. If there would be more possibilities, one would be randomly selected.
   - Of course, the condition of minimal production length which has to be lower than the maximal depth of tree has to be satisfied.
   - In the beginning, the current size (CS) of tree was 0 and after the first step it was 1.

2. When the root symbol is generated, the rest of tree is generated randomly according to the given grammar and the maximal depth of tree.
   - CS of tree is increased by 1 in every step until the end of every production rule is reached.
   - In each step, the rule which satisfies the condition $CS + L(A ::= \alpha) \leq D$ is randomly selected.
   - This process is repeated until every branch of tree is finished by the leaf of terminal symbol, and then the CS is incremented by one.

6.2.2 Grammar Driven Crossover Operator

The crossover operator has to be adjusted to the CFG model, because the complete random process of sub-tree swapping was eliminated when the CFG was employed. So, several rules of how the crossover works were defined. The crossover operator presented in this section was inspired by the GBX operator proposed in [280] and slightly simplified in few steps. The process of how the operator works is described below and depicted in Fig. 6.5.

1. The first step is to select one syntactical tree as the first parent and decide if it is suitable to crossover all nodes, or only non-terminal symbols. As a default option we consider that all nodes can be processed. Then the set N that contains all appropriate symbols of the first parent, except the root, is created.
2. If $N \neq \emptyset$, then one element of this set is selected randomly. This element is called crossover node ($CN1$). If N is empty, there is no node of given type

to be crossed and the process starts from scratch. The symbol E shaded gray was selected as $CN1$ in Fig. 6.5.

3. The parent of $CN1$ regarding to given CFG in Fig. 6.4 represents the symbol A of the grammar (see definitions). This symbol produces one or more consequent rules. All of these consequent rules derived from A are stored in the set R. Regarding our example, the parent of $CN1$ is the symbol E, therefore the set $R = \{E+E, E-E, F+E, F-E, N\}$.
4. Then, according to [280] the tuple $T = (l, p, \alpha)$ is calculated, where α refers to the production rule which generates $CN1$, l is the length of production rule – it means the number of terminal and non-terminal symbols in the rule, in our example α is $E+E$ and its length is 2, because this rule has two operators of the symbol E, and p stands for the position of $CN1$ in the production rule. The tuple $T = (2, 1st, E+E)$.
5. After the tuple T is determined, all the rules of different length of l are removed from the set R. So, $R = \{E+E, E-E, F+E, F-E\}$ in this case.
6. Thereafter, each element of R is compared to the rule α which is $E+E$ except the element on the position p, and those where the change in action or in symbol on the right is detected are removed. So, the set R is adjusted again $R = \{E+E, F+E\}$.
7. In this step, the set X is formed by all symbols which are in R on the position p, so the set $X = \{E, F\}$.
8. If $X \neq \emptyset$ then the crossover symbol CS is randomly selected from X. The set P is created from the second parent and filled in by all nodes which include the symbol CS. If X is an empty set, it is not possible to determine the crossover node in the second parent. In such case, new $CN1$ has to be randomly selected from N and the process continues by step no. 2. In this example, $CS = F$ and P contains all nodes from the second parent the symbol of which is F (only one element in this case).
9. If $P \neq \emptyset$ the $CN2$ is randomly selected from P. Otherwise, CS is removed from X and step 8 is performed again. In this example, only one node F is present and is stated as $CN2$.
10. The $P1$ value is calculated as the depth of node $CN1$ plus the depth of sub-tree whose root is $CN2$. $P2$ is calculated similarly. If $P1$ or $P2$ exceeds the value of the maximal depth of tree D, then $CN2$ is removed from the set P and the process continues by step 9. In this case, $P1 = 2+2$ and $P2 = 3+2$, which means that both $P1, P2 \leq 5$, therefore it is possible to cross sub-trees and continue the process.
11. Finally, two new descendants are generated by swapping the sub-trees and given to a new population.

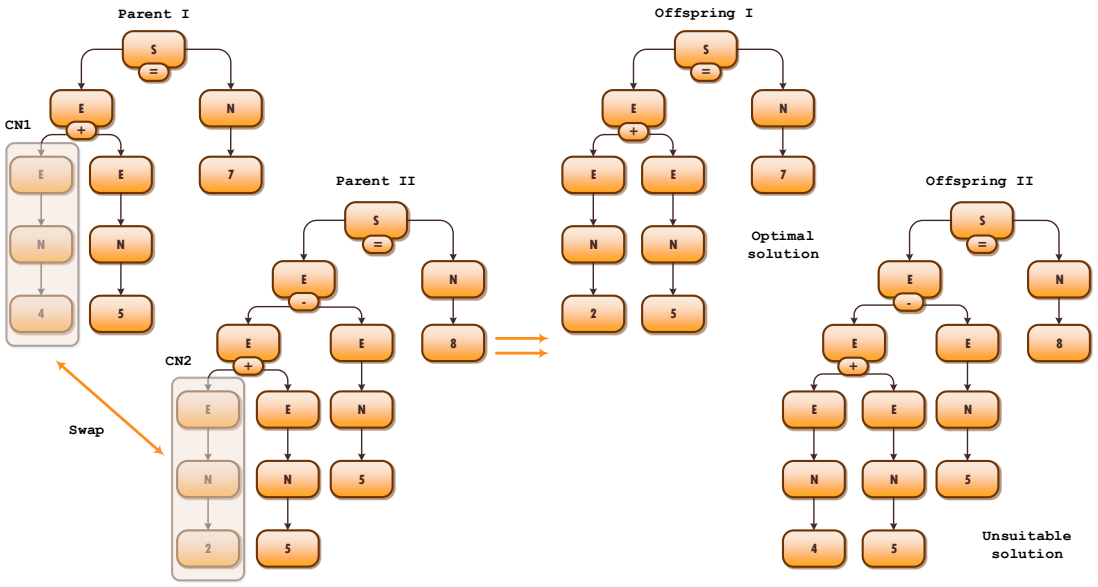


Fig. 6.5: An example of the Grammar Driven Crossover.

6.2.3 Grammar Driven Mutation Operator

The Grammar-Driven Mutation (GDM) works identically with the Grammar-Driven Crossover in the first seven steps, just the crossover node ($CN1$) is renamed to a mutation node (MN), and the following steps are described below:

8. If $X \neq \emptyset$ the CS is randomly chosen, otherwise it is impossible to find a valid mutation node MN. If the operation is impossible, the node MN is removed from N and the process continues with step 2.
9. Now, the mutation length (ML) is calculated as the depth of the MN minus the maximal given depth of the tree D. And the value 0 is assigned to the current depth $CD = 0$ of the new sub-tree generated within the scope of mutation.
10. Then the set of production rules PP is stated similarly to GDIM and the condition $CD + L(CD ::= \alpha) \leq ML$ has to be satisfied, where $\alpha \in \Sigma_N \cup \Sigma_T$. If the PP is an empty set, CS is removed from the set X and the mutation process continues with step 8.
11. Choose a production rule randomly from the set PP and continue with the random generation of sub-tree as this were an initialization of a new tree. Of course, the condition of the maximal tree depth has to be fulfilled, which is $(CD + 1) + L(A ::= \alpha) \leq ML$ in this case.

6.3 Use Case – Non-Cryptographic Hash Design

This use case presents one of the first examples which incorporate the use of the GP module of Evolutionary Framework. The main goal of this task was to design a non-cryptographic hash function comparable to the other known function of this type e.g. GPHash [301], [302], which is a hash function generated by the GP. GPHash function is considered a fast algorithm and its results regarding collisions are also quite impressive. Another hash function used for comparison is the non-cryptographic hash function named FNVHash. The basis of algorithm was taken from an idea sent as reviewer comments to the IEEE POSIX P1003.2 committee by Glenn Fowler, Phong Vo and Landon C. Noll in 1991. This function is used in domain name servers, databases, web search engines, email servers, anti-spam filters and many other applications. DEKHash is an algorithm proposed by the well-known mathematician Donald E. Knuth in The Art of Computer Programming – Volume 3, under the topic of sorting and search. DJBHash is a hash function presented by professor D. J. Bernstein and at the same time it is one of the most efficient non-cryptographic hash algorithm published. BJHash is a hash function introduced by R. Jenkins and it is mostly used in data structures such as hash tables.

At the beginning, a non-terminal symbols set (functions) and terminal symbols (inputs) have been set as basic building blocks. The non-terminal set comprises: logical conjunction (and, &), logical disjunction (or, |), logical negation (not,˜), multiplication (mult, $\times$), summation (sum, +), logical exclusive disjunction (xor, $\wedge$), and right bit rotation (rr). The terminal set comprises *hash* (the hashed value), *character* (character at certain position in the processed input string sequence), and *magic_no* (magic number inspired by FNVHash). The number brings the non-linearity and produces a hash of certain length. The value of number is 0x811C9DC5 (in decimal form it is 2,166,136,231 and of course, the number is a prime number).

The objective function has been designed in the following way. The sum of collisions acquired by hashing 10^6 random strings of 32bit length. Firstly, the hash is computed, bit by bit, from the 32bit input string by the candidate hash function produced automatically by GP. The 10^6 strings generated at random are hashed by the candidate solution. Secondly, it is calculated how many collisions were produced by the candidate hash function, and this is the score of the hash function, the fitness value. The hash function which reaches the minimum number of collisions is declared the best solution given by the evolution framework.

The grammar used is described in Algorithm 1 and in more details in [314] as well as the whole design of the GP algorithm, inputs in the form of terminal and non-terminal symbols, a fitness measure, configuration parameters etc. The pseudo-code of resulting non-cryptographic hash function is depicted in Algorithm 2.

Algorithm 1 The grammar defined for the hash function design.

$Root := E$,
$E := E \wedge E \mid E \text{ or } E \mid E \,\&\, E \mid E + E \mid E \times E \mid$
$\quad E \wedge C \mid E \text{ or } C \mid E \,\&\, C \mid E + C \mid E \times C \mid$
$\quad E \wedge N \mid E \text{ or } N \mid E \,\&\, N \mid E + N \mid E \times N \mid$
$\quad N \wedge C \mid N \text{ or } C \mid N \,\&\, C \mid N + C \mid N \times C \mid F$,
$F := RR\ N \mid RR\ F \mid RR\ E \mid$
$\quad NOT\ N \mid NOT\ F \mid NOT\ E \mid N$,
$C := 0x811C9DC5$,
$N :=$ *hash* (from previous iteration, the first is equal to 0) |
char (character of string in successive order).

Algorithm 2 The proposed EFHash function algorithm.

input: string *str* of size *length* to be hashed
output: *hash* as a long number

$magic\_no \leftarrow 0x811C9DC5$
$hash \leftarrow 0$

for $i \leftarrow 0$ **to** *length* **do**
  $hash \leftarrow ((hash \,\&\, magic\_no) + str.getCharAtPosition(i))$
  $hash \leftarrow hash \times ((\sim ((\sim hash) + magic\_no)) >> 2)$
end for

return *hash*

Many different configurations of terminal and non-terminal sets and fitness function have been tried. The rates of crossover and mutation have been subjected to exhaustive examination too. And after many weeks of testing the GP algorithm, the appropriate parameters which refer to controlling the run were set, please see [314], and the best solution was finally found.

The speed test was carried out as follows. All hash algorithms have been implemented in JAVA programming language, in which all tests were carried out. At the beginning, the strings of 32, 64, 128, 256, 512, and 1024 bits were randomly generated. In each group there were 10^6 random strings. The hash function processed each group ten times. These ten measurements were averaged arithmetically and stored. The results of time simulation are depicted in Tab. 6.3 and Fig. 6.6.

Tab. 6.1: Speed test of non-cryptographic hash algorithms, results in [ms].

Algorithm	32bit	64bit	128bit	256bit	512bit	1024bit
BJHash	0.367	0.413	0.606	0.986	1.816	2.871
EFHash	0.074	0.105	0.169	0.293	0.550	1.065
GPHash	0.074	0.111	0.177	0.301	0.575	1.125
DEKHash	0.056	0.074	0.104	0.183	0.318	0.598
DJBHash	0.057	0.074	0.103	0.181	0.314	0.595
FNVHash	0.061	0.084	0.111	0.190	0.340	0.640

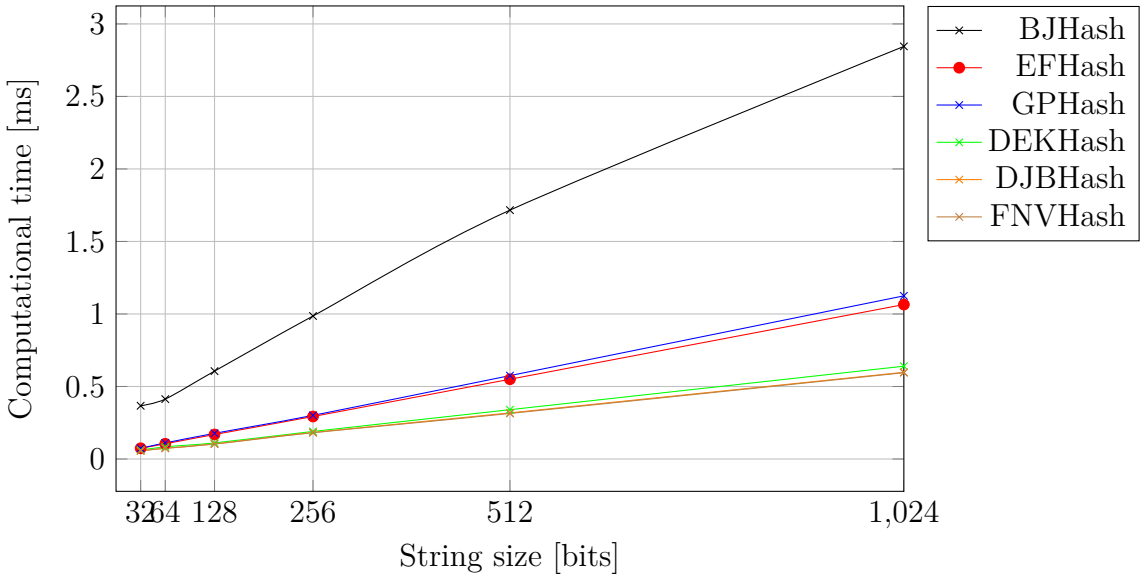


Fig. 6.6: Speed test of non-cryptographic hash algorithms.

The first collision test was performed on the same randomly generated strings as the speed test. There are six sets of different bit lengths and each of sets contains unique strings. The strings are randomly generated from characters a–z and 0–9. All sets are mixed up to one set with 10^6 randomly generated strings in the range from 32 bits to 1024 bits. Hash functions were started on this set. The first 10^2, 10^3, 10^4, 10^5, and 10^6 strings of each set were processed and the results of reached collisions are depicted in Tab. 6.3.

The second collision test shows how many hashes (in average) a hash function algorithm can produce before generating the first collision. Besides, it is depicted how the number of collisions grows when the number of hashes grows. The Mersenne Twist Generator was used in this test. The test set consists of random numbers only. The maximum length of random number is 1,024 bits. The hash function was processed ten times. These measurements were averaged arithmetically and rounded up. The results reached are depicted in Tab. 6.3.

Tab. 6.2: The collision test of hash functions I.

Algorithm	10^2	10^3	10^4	10^5	10^6
BJHash	0	0	0	2	119
EFHash	0	0	0	0	0
GPHash	0	0	0	0	0
DEKHash	0	35	319	1293	1340
DJBHash	0	29	288	951	992
FNVHash	0	0	0	0	0

Tab. 6.3: The collision test of hash functions II.

Algorithm	1	2	3	4	5
BJHash	117 145	168 322	178 507	182 541	188 182
EFHash	73 561	148 137	165 383	251 736	306 792
GPHash	76 519	133 368	137 280	156 540	161 076
DEKHash	123 094	187 204	206 570	212 548	233 598
DJBHash	57 724	90 071	127 812	188 884	218 495
FNVHash	152 515	207 282	210 895	215 018	227 487

The total number of collisions, which occurred during hashing the set of 10^7 randomly generated strings by Mersenne Twister Generator are shown in Fig. 6.7. The resulting values of all algorithms are very similar, but it can be seen that EFHash reached a very good position.

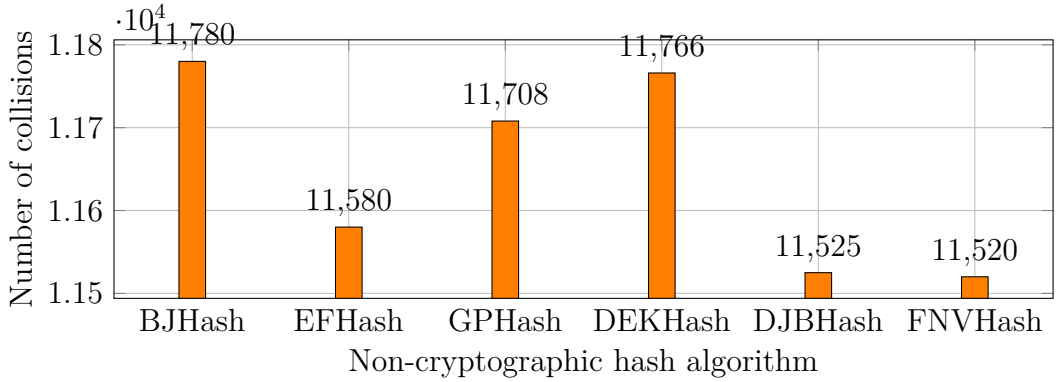


Fig. 6.7: Quantitative test of non-cryptographic hash algorithms.

The proposed results show that the EFHash is at an average level of the existing non-cryptographic hash functions. However, the direct comparison with GPHash shows that they are basically at the same level even if different GP algorithms were used for the development of these functions, which proves a good architecture of the proposed Java Evolutionary Framework. The use cases were implemented in the Java language, and all test were deployed on the Intel C2D E8400 architecture.

6.4 Use Case – Artery Localization Method

This use case example presents the use of the GP module in the biomedical image analysis. The goal of this example is to localize a common carotid artery in B-mode ultrasound images, which is a source of important information that doctors can use to evaluate the patients' health in non-invasive way. The most often measured parameters are arterial stiffness, lumen diameter, wall thickness, and other parameters which depend on the localization of artery in the image. The GP module in this use case example was used to automatically design an image filter for initial localization of the artery in the image which must precede the main measurements.

First of all, it was necessary to establish a set of non-terminal symbols and a set of terminal symbols. The set of functions includes image transformations such as blurring operations (Gaussian blur) for smoothing the input image, then the operations which analyze the degree of curvature in the image (Hessian, Laplace, Sobel, Watershed), the operations for image equalization (Histogram equalization), the morphological operations (Erosion, Dilatation, Close, Open), the thresholding operations (Thresholding, Entropy Thresholding), and finally the operation for circle detection (Hough). In the terminal set there is now only the analyzed image and the integer value used in some filters from the non-terminal set. The CFG used for the design of algorithm for the CCA localization from images is depicted in Algorithm 3.

Algorithm 3 The grammar defined for the CCA artery localization.

$$
\begin{aligned}
&Root ::= HoughTransform,\\
&HoughTransform ::= Erode\ Integer\ Integer\ Integer\ |\\
&\qquad\qquad\qquad Dilate\ Integer\ Integer\ Integer\ |\\
&\qquad\qquad\qquad Close\ Integer\ Integer\ Integer\ |\\
&\qquad\qquad\qquad Open\ Integer\ Integer\ Integer\ |,\\
&Erode \mid Dilate ::= Threshold \mid EntThreshold,\\
&Close \mid Open \ ::= Threshold \mid EntThreshold,\\
&Threshold \ ::= HistEqual, Laplace, Sobel, Watershed,\\
&EntThreshold ::= HistEqual, Laplace, Sobel, Watershed,\\
&HistEqual ::= Hessian,\\
&Hessian \ ::= Blur \mid Logarithm,\\
&Laplace \ ::= Blur \mid Logarithm,\\
&Sobel \ ::= Blur \mid Logarithm,\\
&Watershed ::= Blur \mid Logarithm,\\
&Blur \ ::= Logarithm\ Integer \mid Image\ Integer,\\
&Logarithm ::= Image.
\end{aligned}
$$

Subsequently, the database of 155 images was created and the center position of circular CCA was marked in each of these images. The training database consists only of 9 images, and the rest was used for the testing process.

The objective function was designed for the evaluation of individuals based on the calculation of the accuracy of the detection of centers in all arteries. The filter was designed in 40 evolution steps with initialization population which contained 100 chromosomes, with 80 % crossover rate and 30 % mutation rate. The total time of the final filter design was 12 hours 11 minutes. The detection of artery by an individual chromosome (candidate filter) takes from 1.2 s to 1.8 s in an individual image. The resulting filter is depicted in Fig. 6.8.

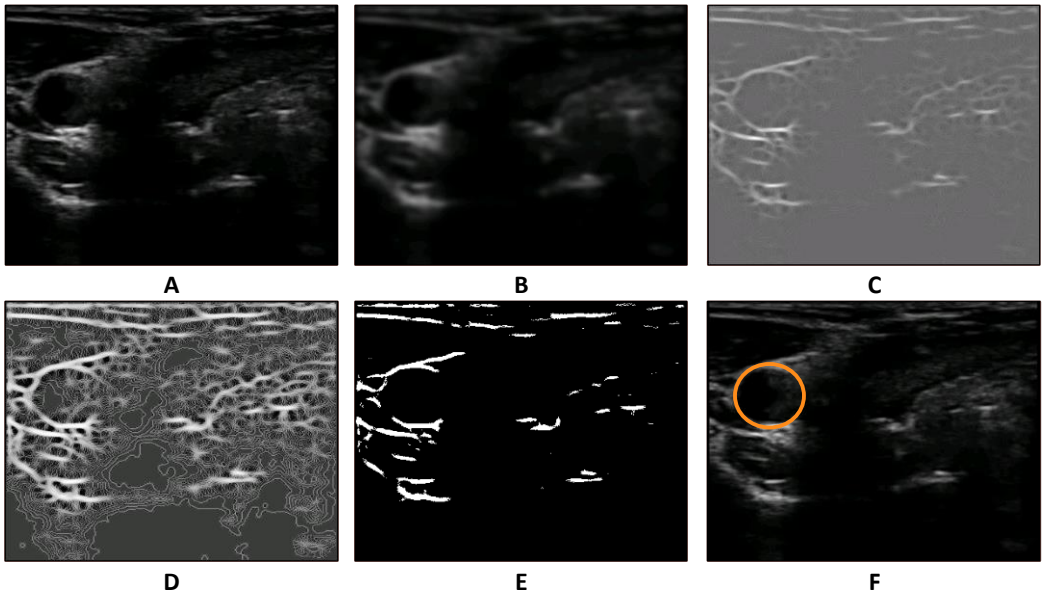


Fig. 6.8: The steps of processing of the designed image filter – a) Original image. b) Output of Gaussian smooth, c) Output of Hessian, d) Output of Histogram equalization, e) Output of Threshold, and f) Output of Hough transform with the final localized artery in the input image.

The input image is given to the entry of *Gaussian blur*. This filter is one of the linear smoothing filters and its main task is to reduce noise. The blurring value must not be chosen too large to prevent the filter to remove some important properties of the image. The blurred image then enters the block which analyzes the degree of curvature in the image using the *Hessian*. The operator indicates with a light color the areas where a certain curvature is apparent. This can be used with advantage to find circular shapes as required by assignment. The proposed filter then performs *histogram equalization* which enhances the image contrast. This is followed by *binary thresholding*. The accepted method comes out of the assumption that the input image is generated from two signals: the foreground and the background signal.

An ideal thresholding occurs at the moment when the sum of entropies of the two signals reaches the maximum. Based on this premise, a formula is established to determine the ideal threshold, which is then used for binary thresholding. Over the binary images, the morphological operations of erosion and dilation, constituting together the operation *Close*, were performed. The last step, strictly prescribed by grammar, is *Hough transform*. Besides the input image, it takes also the minimum and the maximum size of the circle and the step to analyze the range of values.

The success of the CCA localization in images of the testing set was approximately 75 %. The success at this stage of development was very good. More about the experiments using ultrasound images as inputs is described in [315], [316].

The experiment was further developed and performed on ultrasound video sequences with much better results. The training database contained 16 video sequences and the validation database contained 52 video sequences which were used for the evaluation of accuracy. The resulting success of the proposed solution was 82.7 %, which exceeded the current state of the art by 4 % while the computation time requirements were acceptable. More about this experiments can be found in the paper [317] which proposes a complete process of automatic construction of a machine vision system for the CCA localization in B-mode ultrasound video sequences.

The set of non-terminal symbols in the video sequence version of the problem was enhanced. Approximately 30 different functions were implemented for our specific utilization – for designing the CCA localization process. The function set contains, for example, optical flow (Lucas & Kanade), morphological operations (dilatation, erosion, open, close), blur (Gaussian, anisotropic diffusion, median filtering), thresholding (fixed-level binary threshold, adaptive threshold, thresholding with automatic threshold setting according to entropy), sharp and many others [317].

In this implementation three types of parameters are use: numerical parameters, images, and video sequences. Numerical parameter is an input of numerical constants that can be used for the settings of parameters of particular building blocks of image processing. The value is generated from the range specific for the component. For example, two parameters entering the thresholding building block, where the first specifies the threshold (range from 0 to 255) and the second specifies the thresholding method (range from 0 to 12). The parameter image is used as image input and the parameter video sequence is used for video sequences (i.e. a set of consecutive images). Because the grammar for this problem is too complex, only a simplified version for the localization from images is depicted in Algorithm 3.

The objective function has to reflect all the important measurable parameters. In this implementation, the most precise determination of the centers of circular artery in a set of training video sequences is the most important thing. Therefore, the first and the most important parameter is the number of precise localizations h (number of phenomena, the artery was localized with a deviation of less than the chosen 20 px) and the second parameter is the accuracy of localization centers of the artery p_i in a particular video sequence in the training set, i belongs to the interval from 1 to N. The formula is depicted in Eq. 6.1.

$$f \;=\; h \times 100,000 + \sum_{i\,=\,1}^{N} p_i. \tag{6.1}$$

Multiplying the value h by the constant 100,000 secures assigning a significant weight to this variable in the process of calculation of the fitness function, and the fitness value will be well analyzable. The value p_i gives the precision of artery localization in the i-th video sequence. The values p_i are in the range from 0 to 255, where the highest value represents a better localization according to Eq. 6.2:

$$p_i \;=\; 255 - min(10 \;\times\; dist(X_i^s, X_i^d), 255), \tag{6.2}$$

where X_i^s is the determined artery center in the i-th video sequence, and X_i^d is the center localized by the proposed algorithm. The operation *dist* is the calculation of Euclidean distance. The result of proposed filter is depicted in Fig. 6.9.

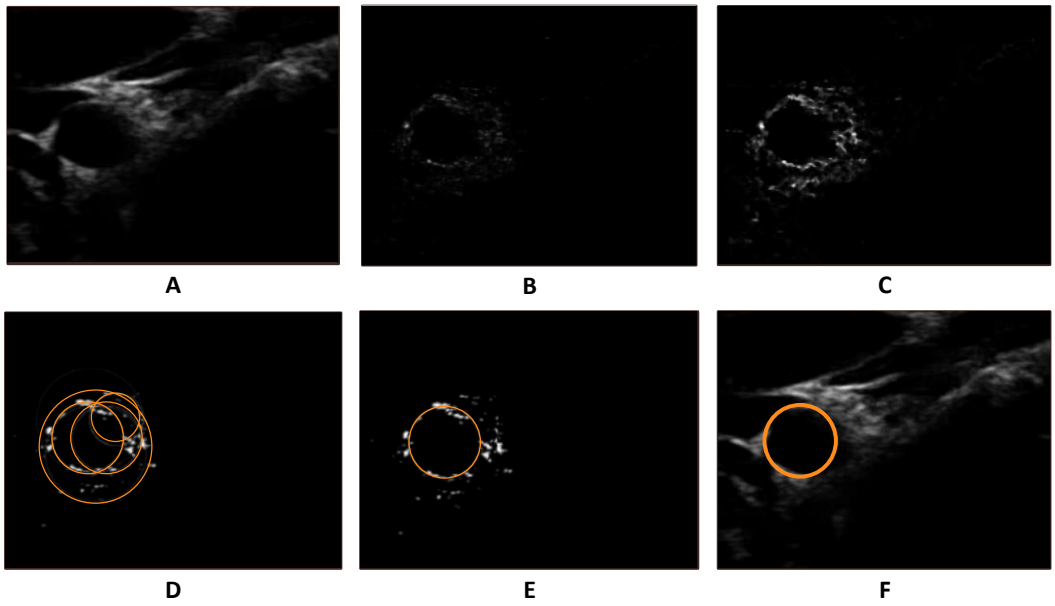


Fig. 6.9: The steps of processing of the designed video sequence filter – a) Original image, b) After optical flow, c) After further processing, d) The set of circles found by the Hough transform, e) Selected circle, and f) Image with the localized artery.

7 THE WAREHOUSE OPTIMIZATION ALGORITHM

This chapter describes how all the expert knowledge and algorithms implemented were put together into one algorithm for warehouse process optimization. Chapter represents the main contribution of the thesis which should directly lead to reach the goal of the thesis and confirm the hypothesis by results in the next chapter.

The chapter is divided into four sections. Section 7.1 describes the design of the optimization algorithm from scratch. This section depicts how the set of terminal and non-terminal symbols was identified and defined, the complete design of objective function, the parameters for controlling the run of the optimization algorithm and the termination criterion with the method for designating the result of the algorithm's run. Section 7.2 describes the set of recursive rewriting rules which were used for the search space restriction. Section 7.3 describes the genetic operators which were designed specially for the problem of warehouse process scheduling, and section 7.5 gives a description of algorithm's work-flow.

7.1 The Design of Optimization Algorithm

First of all, the design of the algorithm had to be made with the support of the expert consultant in the area of logistic optimization, who was willing to tell the basic facts about warehouse optimization, such as basic building blocks for the GP algorithm. The optimization algorithm was designed from scratch, which means that five preparatory steps of the GP algorithm were discussed with the expert.

7.1.1 The Set of Terminal Symbols

In this case, the terminal symbol represents one single indivisible operation in the warehouse. As terminal symbols were identified the following operations which represent the very basic *tasks*, which are common to all employees in the warehouse, and other warehouse equipment such as trucks and employees. These basic building blocks, terminal symbols, are depicted in Tab. 7.1.

The tasks *TaskLoad* and *TaskUnload* are directly connected with the pallet manipulation, while *TaskMove*, *TaskRelax*, and *TaskWait* can be performed even with an unloaded truck, or even without a truck completely. The *TaskMove* represents all transports and transfers of employees and trucks through the warehouse. The *TaskRelax* represents a break of an employee for food, toilet etc. The *TaskWait* represents a waiting time when the aisle is congested and the employee with the truck has to wait until it will be free, it is quite a variable time window which should be

Tab. 7.1: The terminal symbols identified for the GP algorithm.

Employee	The employee.
Commodity	The commodity or goods.
Coordinate	The [x, y] coordinates.
ForkLiftHand	The fork-lift hand pallet truck.
ForkLiftLow	The fork-lift low pallet truck.
ForkLiftHigh	The fork-lift high pallet truck.
TaskLoad	The employee loads the pallet.
TaskMove	The employee moves the pallet or only the truck from A to B.
TaskRelax	The task represents the break of an employee.
TaskUnload	The truck unloads the pallet.
TaskWait	The task represents the wait-time of an employee.

minimized. Of course, there is a lot more operations such as *TaskCheck* for checking the order, *TaskPack* for packing the order, *TaskShip* for shipping the order and many other tasks that represent all operations in the warehouse.

The *Employee* represents an employee of the warehouse. All employees have to have assigned a truck of following three types: *ForkLiftHand* pallet truck, *ForkLiftLow* pallet truck, or *ForkLiftHigh* pallet truck. The trucks have different parameters, such as maximal velocity, size, level of loading the goods in the axis z, etc. Of course, there can be lots of other instruments and machines such as *PackingMachine* etc.

7.1.2 The Set of Non-terminal Symbols

The set of non-terminal symbols was determined as follows. The basic function is represented by the *Workplan*. The work-plan represents the structure of the jobs which must be fulfilled by one employee. The work-plan consists of *Job*s. This is a function which can directly represent a *JobInStore* or *JobOutStore* or can be divided into one of these particular jobs and other jobs. The *JobInStore* represents the job which starts both in the warehouse or outside the warehouse, but always ends in the warehouse, such as storage. The *JobOutStore* represents the job which starts in the warehouse, but ends outside the warehouse, such as truck loading. The set of non-terminal symbols is depicted in Tab. 7.2.

Tab. 7.2: The non-terminal symbols identified for the GP algorithm.

Job	Job represents job from the job buffer.
JobInStore	Job represents job in the warehouse or heading to the warehouse.
JobOutStore	Job represents job starting in the warehouse and heading outside.
Workplan	The basic function which starts the structure of the work-plan.

7.1.3 The Design of Objective Function

The next step is to define the objective function for the suitability measurement of particular individuals of the population. In most cases, the suitability is defined as an error of the resulting individual of the GP algorithm. The closer this value is to zero, the better solution it is. Due to the NP-hardness of this problem, it is not possible to clearly determine an optimal solution to the problem of scheduling. However, the main task is to achieve the lowest time required to successfully carry out all jobs in the work-plan. The final suitability can be obtained as the maximal final times among all work-plans, i.e. $C_{max} = max(C_1, \ldots, C_n)$. This maximum represents the inverse of the suitability solution, therefore the best solution will be an individual with the lowest maximum of end time, depicted in Eq. 7.1.

$$\frac{1}{fitness} = max(C_1, \ldots, C_n). \tag{7.1}$$

To avoid unnecessary hassle during deployment in practice, collisions of trucks are expected. Therefore, the resulting fitness functions will include the sum of the maximum end time and the number of collisions (Θ) that occurred during all tasks which were processed throughout the warehouse. Also, individual weighting factors (w_1, w_2) were assigned to all parts of fitness function and the fitness function will be therefore (see Eq. 7.2):

$$\frac{1}{fitness} = w_1 \times max(C_1, \ldots, C_n) + w_2 \times \Theta. \tag{7.2}$$

Now, the fitness function formed should be sufficient. But, for better results, and even for a better distribution of jobs through the employees, another parameter will be added. The average duration of a single job. The resulting fitness function is then a function in Eq. 7.3 and of course, accompanied by the weighting factor w_3:

$$\frac{1}{fitness} = w_1 \times max(C_1, \ldots, C_n) + w_2 \times \Theta + w_3 \times \frac{\sum_{j=1}^{n} T_j}{n}. \tag{7.3}$$

7.1.4 The Parameters for Controlling the Run

Thanks to the grammar defined in section 7.2 it is ensured that the basic criteria for the GP algorithm run, such as sufficiency requirement and closure requirement, are fulfilled. The parameters for controlling the run of the algorithm are following:

7.1.5 The Termination Criterion and The Result Design

The final step is to define the initialization conditions which end the algorithm. As the optimal solution is not known, the termination condition is defined as the

Tab. 7.3: The parameters of controlling the run of the GP algorithm.

Population Size	The size of the population, the number of chromosomes.
Evolution Size	The size of evolution process, the number of generations.
Fitness Precision	The desired precision of the fitness function.
Simulation Time	The maximal time of the simulation.
Elitism Rate	The number of individuals copied to the new population.
PA Mutation Rate	The rate of path mutation.
JO Mutation Rate	The rate of job order mutation.
SJ Mutation Rate	The rate of swap job mutation.
SW Mutation Rate	The rate of swap work-plan mutation.
SP Mutation Rate	The rate of split job mutation.

maximum number of generations if the desired precision of result is not met. Then, the best individual is selected and claimed as the optimal solution.

7.2 The Design of Context-Free Grammar

The next step is to define the relationships between the terminal and the non-terminal symbols which form grammar together. The grammar gives syntactic restrictions to the algorithm and clearly defines a list of allowed terms in the language understandable to machines. Using the grammar in the GP algorithm can significantly reduce the search space. The structure of CFG is defined by Algorithm 4, and a descriptive example is shown in Fig. 7.1.

Algorithm 4 The grammar defined for the warehouse optimization.

$Root ::= Workplan\ Employee\ Equipment,$
$Workplan ::= Job \mid Job\ Job \mid JobInStore \mid JobOutStore$
$Job ::= JobInStore \mid JobInStore\ Job \mid$
$\quad JobOutStore \mid JobOutStore\ Job$
$JobInStore ::= TaskLoad\ TaskMove\ TaskUnload\ Coord\ Coord\ Commodity$
$JobOutStore ::= TaskLoad\ TaskMove\ TaskUnload\ Coord\ Coord\ Commodity$
$Equipment ::= Truck \mid Machine$
$Truck ::= ForkLiftHand \mid ForkLiftLow \mid ForkLiftHigh$
$Machine ::= PackingMachine \mid CleaningMachine$

The starting symbol is denoted as $ROOT$. The $ROOT$ has three basic components, which are $Workplan$, $Employee$, and $Equipment$. The $Workplan$ represents a set of jobs for the $Employee$ which should be done with the help of $Equipment$. The $Workplan$ can be rewritten to Job or two $Jobs$, or directly to $JobInStore$ or $JobOutStore$. The Job can evolve to both $JobInStore$ and $JobOutStore$ or one of these functions accompanied by another Job. The $JobInStore$ and $JobOutStore$ are pretty much the same functions which differ only in the purpose of use. Both jobs consist of a predefined chain of tasks, i.e. $TaskLoad$, $TaskMove$, $TaskUnload$, $Coord$ of starting place, $Coord$ of end place, and $Commodity$ which is transferred through the warehouse. The tasks $TaskRelax$ and $TaskWait$ are added into this chain of tasks at request of an employee. The $Employee$ is selected from the database of employees. The $Equipment$ is either $Truck$ or $Machine$. The $Truck$ is of three types $ForkLiftHand$, $ForkLiftLow$, and $ForkLiftHigh$. The $Machine$ represents everything else.

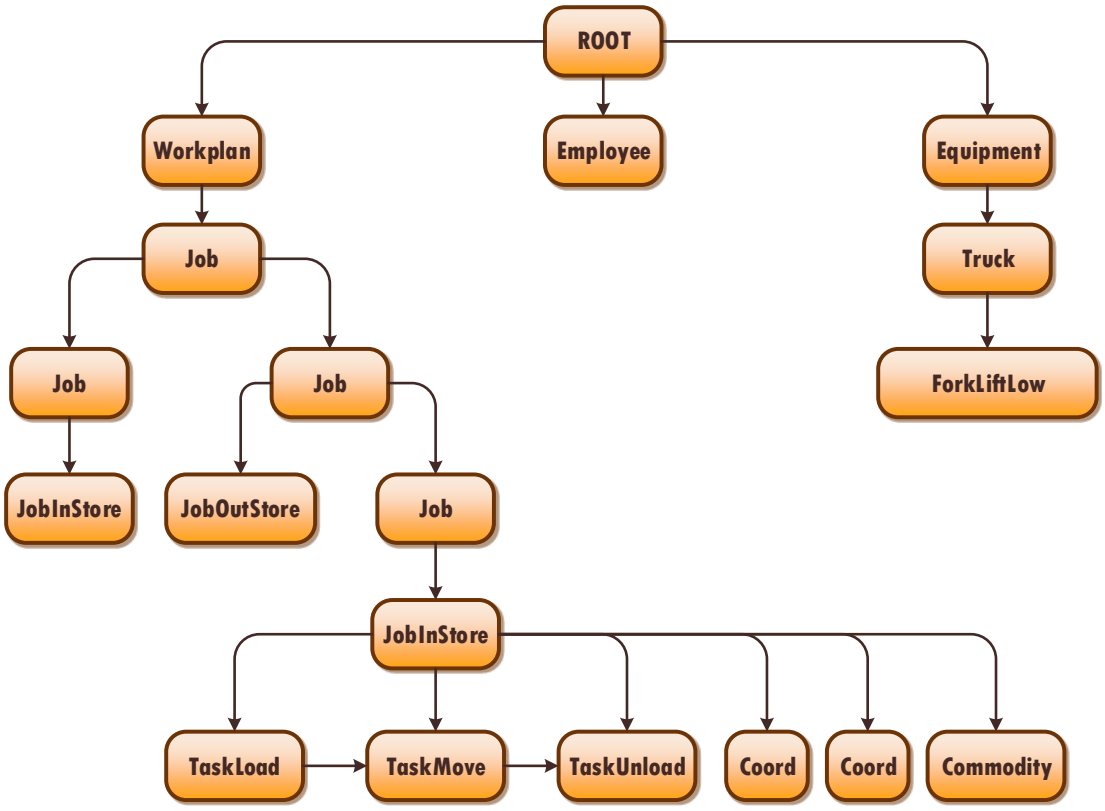


Fig. 7.1: An example of the tree generated by the context-free grammar.

7.3 The Design of Optimization Operators

The genetic operators are described in this section. The design of all operators is described and accompanied by a graphical example of how the each of them works. Together, five new operators were designed, implemented and tested, namely Path Mutation 7.3.1, Job Order Mutation 7.3.2, Swap Job Mutation 7.3.3, Swap Work-Plan Mutation 7.3.4, and Split Job Mutation 7.3.5. All mutation operators designed are depicted in figures. The orange colored blocks depict the movement between two jobs and the brown colored blocks depict the changed, mutated parts of the task, job, or work-plan.

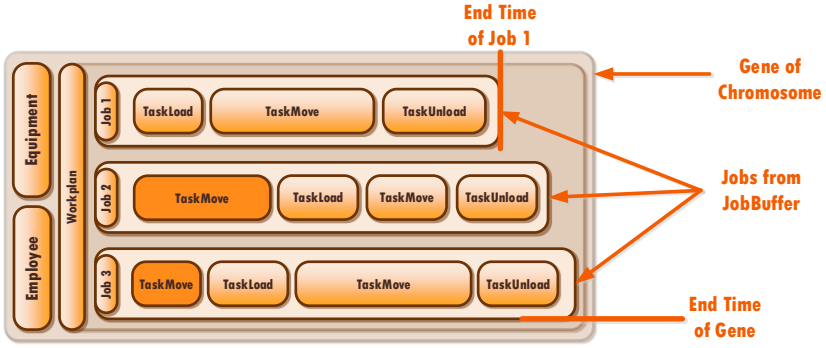


Fig. 7.2: An example of the gene structure used in the optimization algorithm.

7.3.1 Path Mutation

Path Mutation 7.3 is the first genetic operator designed for the purpose of this work. This kind of mutation is the simplest operator and its purpose is to change the path of truck used to process a specific task during the *TaskMove* (e.g. transportation of a pallet through the warehouse) between the *TaskLoad* and *TaskUnload*. The advantage of this operator lies in changing the path, especially when the collision of two trucks is very probable or the aisle between racks is congested for any reason. In the future, this operator could also be used for the path mutation of *TaskMove* between jobs, or any other movement, e.g. when the truck is heading the parking lot in the case of *TaskRelax* or at the end of employee's shift.

The example in Fig. 7.3 shows how the operator works. The parameter R_{pa} represents a percentage of chromosomes which are selected for this mutation. First, the job is selected at random in a chromosome. In this case, it is job number one. Second, the *TaskMove* with *Path A* is selected in the job. Then, the Path Mutation is applied and the transportation path is changed to *Path B* which changes the *TaskMove* operation and it should be shortened. The change is depicted in brown in the figure. This operator was published in the paper [318].

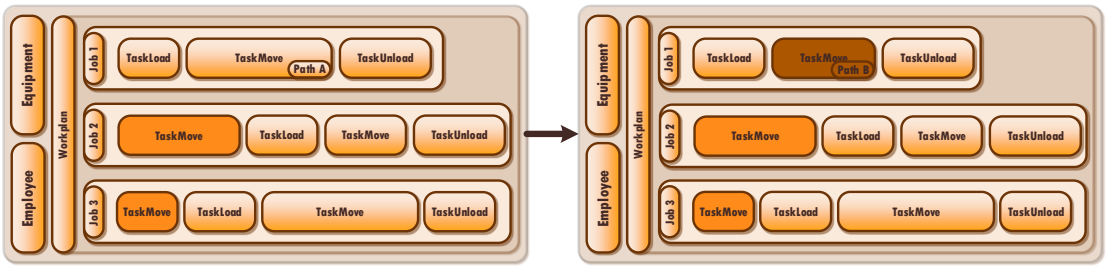


Fig. 7.3: An example of the Path Mutation operator.

7.3.2 Job Order Mutation

The Job Order Mutation 7.4 is the second genetic operator designed. This operator is also quite simple, and its aim is to shuffle the jobs in the work-plan of one employee. This operator can show its advantage especially when the first task in the work-plan looks to be quite distant and it is more logical to process a closer job and then go further and further and process more distant jobs. This mutation operator brings the main advantage in cases when the work-plan of employee consists of many distant tasks. This can bring the continuity of work from one side of the warehouse to another. In other words, the work-plan should be processed from the closest jobs to more distant jobs.

The example in Fig. 7.4 shows how the operator works. The parameter R_{jo} represents the percentage of chromosomes which will be mutated by this operator. When the work-plan is selected based on the probability of R_{jo}, the operator starts. First, the first job is selected at random, in this case *Job 3*. Second, the second job is selected at random, in this case *Job 2*. Then two selected jobs are swapped. As it can be seen in the figure, the finish time of work-plan is shorter, because the third job was closer to the first job and the *TaskMove* between *Job 1* and *Job 2* is shorter after the mutation. This operator was published in the paper [318].

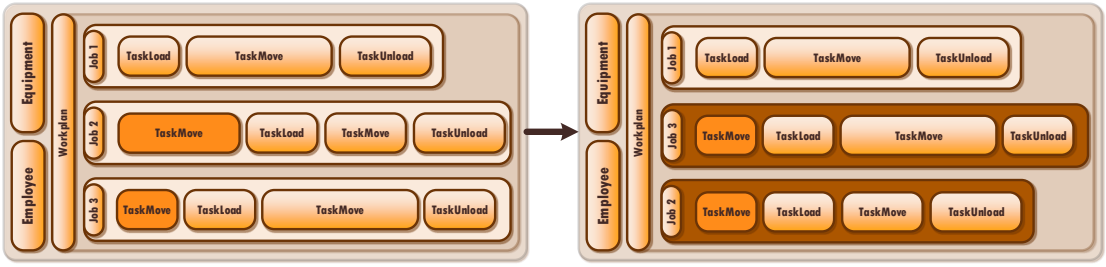


Fig. 7.4: An example of the Job Order Mutation operator.

7.3.3 Swap Jobs Mutation

The Swap Jobs Mutation 7.5 is the third operator presented. This operator is the first which uses two work-plans. The mutation is based on the simple principle which is the swap of two jobs between two work-plans. This operator is advantageous especially in situations when two employees are located in opposite parts of the warehouse and the jobs are randomly assigned, but the employee can fulfill the closest job instead of more distant. The operator swaps the jobs between the work-plans, which helps to save the time of processing and minimizes the path crossing of the trucks.

The example in Fig. 7.5 represents how the operator works. The parameter R_{sj} denotes the percentage rate of this mutation. First of all, two chromosomes are selected based on the probability of R_{sj}. Second, one job is randomly selected in each chromosome. *Job 2* is selected in *Workplan A*, and *Job 3* is selected in *Workplan B*. The third step is the swap of selected jobs. The swap also causes a change of *TaskMove* time, which is given by movement from the first job to second job. The swapped jobs are colored brown.

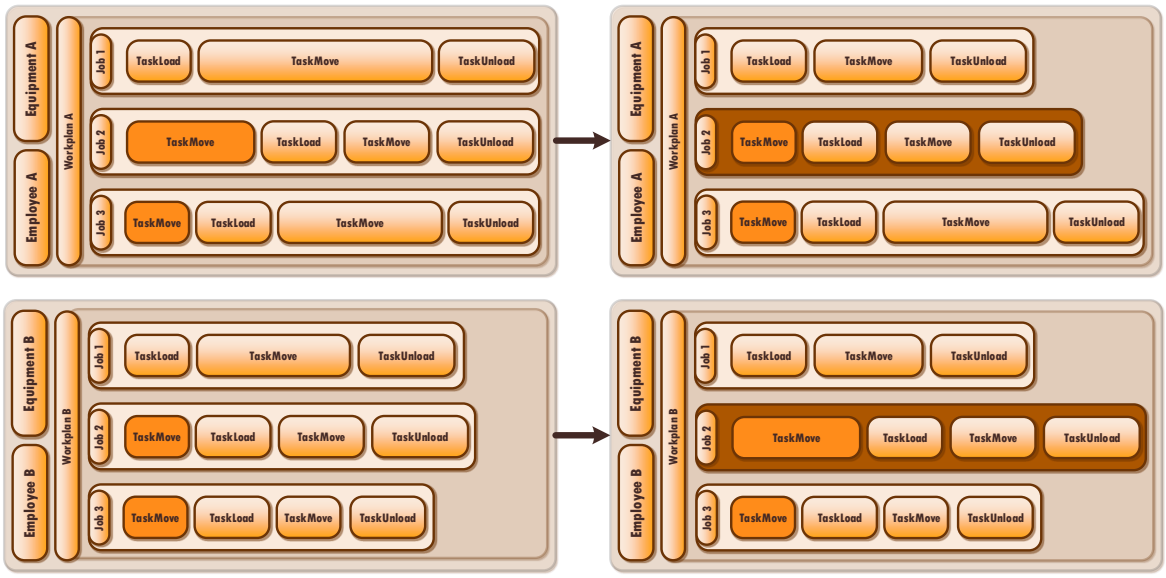


Fig. 7.5: An example of the Swap Job Mutation operator.

7.3.4 Swap Work-Plan Mutation

The Swap Work-Plan Mutation 7.6 is the fourth operator in use. The Swap Work-Plan Mutation operator is based on the principle of work-plan swapping among the population of individuals. The work-plan is bounded with an employee and

an assigned truck. The possibility of work-plan swapping could bring another time improvement, because another employee can have a faster truck, or a truck more suitable for another batch of jobs due to storage level demands etc.

The example in Fig. 7.6 shows the Swap Work-Plan Mutation. The operator works with the parameter R_{sw} which denotes how many individuals will be processed by this operator. When the chromosomes are selected based on this probability, the process of mutation is started. The first step is to randomly select two work-plans and the second step is to swap the work-plans.

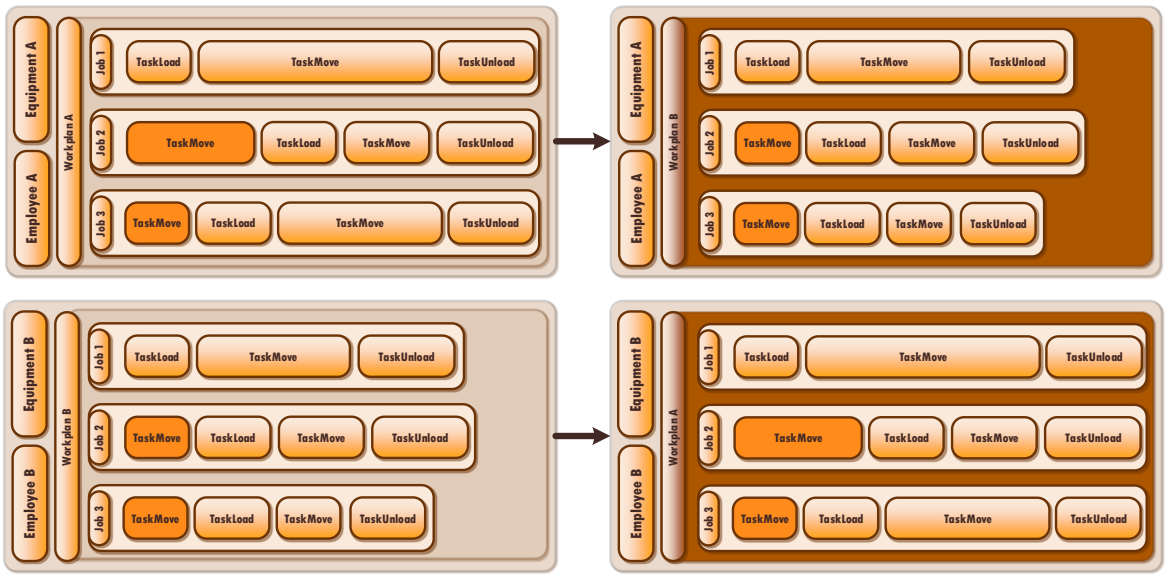


Fig. 7.6: An example of the Swap Work-Plan Mutation operator.

7.3.5 Split Job Mutation

The Split Job Mutation is depicted in Fig. 7.7 and it is the last operator designed for the problem solving. The main aim of this mutation is to split the job into two single jobs with the minimal level of dependence. The key utilization of this operator is in very difficult situations, when it is preferred to process one job with more employees, e.g. *Employee A* with the assigned *ForkLiftLow* truck is going to process *Job A* which starts somewhere in the middle of the warehouse, during his transfer he is able to load the commodity of *Job B* and move it a little bit closer to *Employee B* with the *ForkLiftHigh* truck. *Employee B* loads the commodity of *Job B* and goes to his destination, e.g. at level five of the shelf. During this process *Employee A* is working on *Job A*, because he would not be able to finish *Job B* anyway, since he is equipped only with the *ForkLiftLow* truck. This operator brings an advantage of cooperation into the problem.

The example of mutation operator is depicted in Fig. 7.7. First, genes for mutation were selected with probability R_{sp}. Then the first work-plan was randomly selected, as well as the job in this work-plan. In this example it is *Workplan A* and *Job 1* with the longest *TaskMove*. The *TaskMove* was split up into two jobs *Job A.1* and *Job A.2*. *Job A.1* is kept by *Workplan A*, but *Job A.2* was moved to *Workplan B* at the end of the job list.

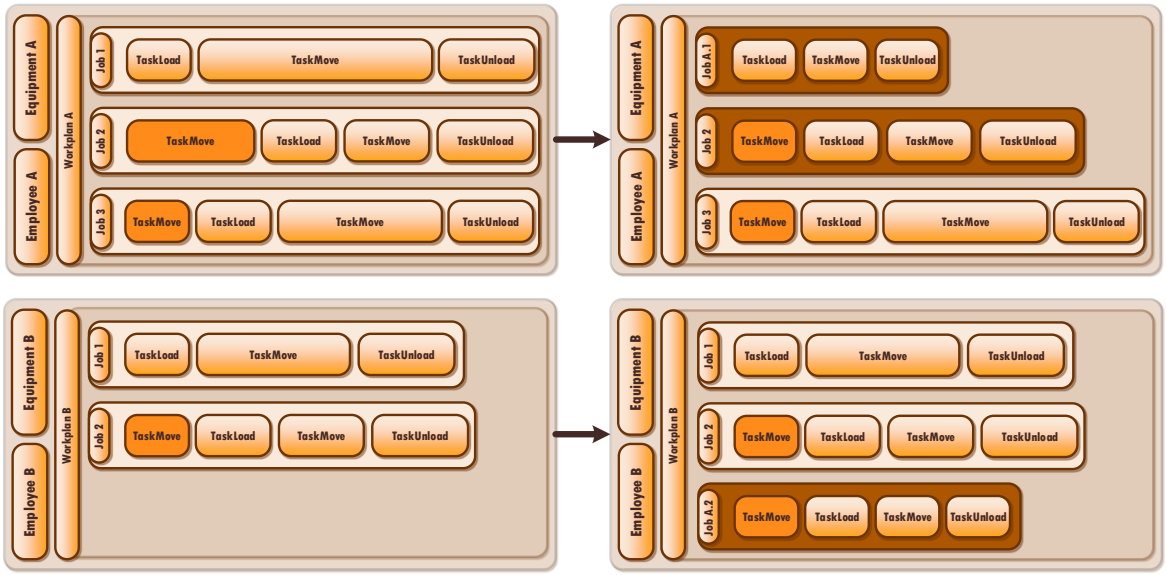


Fig. 7.7: An example of the Split Job Mutation operator.

7.4 Maintaining Mechanisms of Algorithm

Of course, with the implementation of genetic operators, several maintaining mechanisms for the algorithm were also implemented. One of the mechanisms is checking if the employee is authorized to work with the equipment, or if he/she is authorized to process the given job, because not all roles of employees can do each type of jobs.

The maintaining mechanisms connected to genetic operators are as follows. The first mechanism is to ensure that the job which was split up can be completed successfully. It means that the first part of the job will be processed before the second part of the job. So, in situations like this, the *TaskWait* is inserted into the beginning of the second part of the job, which ensures the continuity. Already split up parts of the job can be further divided by algorithm. So, ensuring of the continuity is a key feature. A related situation occurs when the job is split up, then re-ordered or swapped. The continuity of jobs has also to be ensured. The situation when the job is split up, the parts are swapped and re-ordered and they

meet together in one work-plan can also come up during the generations. In such situation, the parts of one job in one work-plan are again connected to one single job. So, this is how the work-plans and jobs are maintained, to be completed successfully.

7.5 The Work-flow of Optimization Algorithm

The optimization algorithm designed in this work is shown in Fig. 7.8. The blocks of algorithm and the basic structure of the algorithm is based on the conventional flowchart proposed by John R. Koza in [247], [248]. The main difference is the change of the genetic operators which were designed specially for the problem of warehouse optimization. Also, the fitness measurement was enhanced by the parameters related to the warehouse environments.

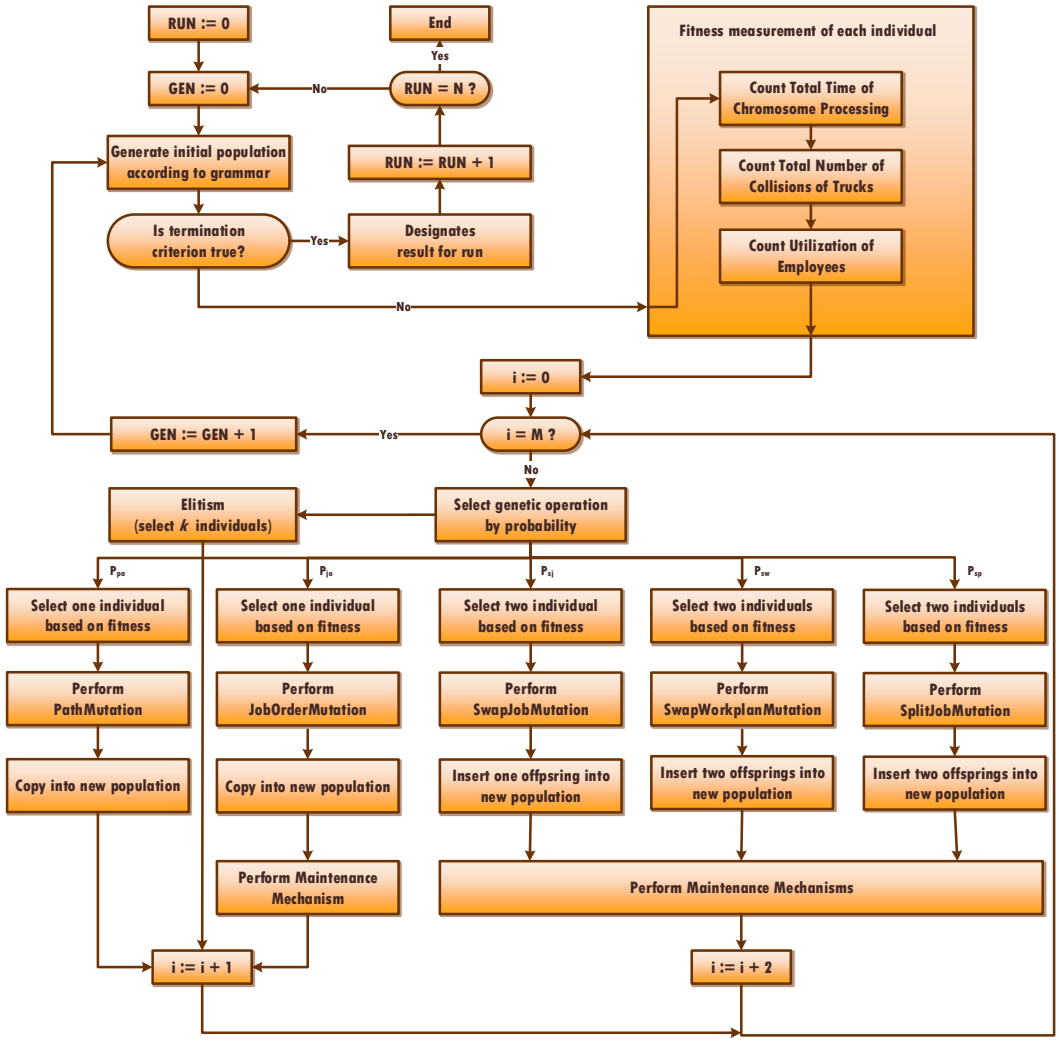


Fig. 7.8: An example of the block scheme of designed optimization algorithm and all genetic operators used including the elitism and the fitness measurement.

8 BENCHMARKING AND TEST SETS

This chapter describes the benchmark definitions and test sets. Section 8.1 is dedicated to the description of layout of the tested warehouse. Section 8.2 outlines the steps of experimental integration which lies in the basic assumptions for benchmarks and test sets definitions, which are based on historical data of warehouse work-flow. Section 8.3 deals with information standardization and normalization, needed for benchmarking. Section 8.4 deals with special real data samples extracted directly from historical data. Section 8.5 deals with the synthetic data generator designed specially for the performance testing of the algorithm proposed in this work. The results of testing are described in the next chapter. The benchmark test and reached results were presented in the paper [318], [319], and [320].

8.1 Layout of Tested Warehouse

The referenced warehouse described in this section is based on a real-world situation. The warehouse layout is built on the warehouse presented in Fig. 2.1 – traditional layout 1. The particular simulation model used in the implemented algorithm is depicted in Fig. 8.1. The warehouse consists of the same parts as the example in Fig. 2.1, such as: trucks importing and exporting commodities; receiving and shipping areas; warehouse gates; office of employees; fork-lift hand pallet truck (able to operate with shelves at level 0, which represents the floor) located on coordinates [8; 1]; fork-lift low truck (operates with shelves at levels 0–2) located on coordinates [2; 5]; fork-lift high truck (operates with shelves at levels 0–9) located on coordinates [10; 7]; the example of pallet, carrying commodity, is located on coordinates [1; 0], and all stationary racks in the warehouse, with shelves 0–9 for commodity storing are colored orange. The rest of the warehouse consists of wide aisles and manipulation areas for receiving, packing, checking and other processes.

The warehouse, in fact, is described by three coordinates [x, y, z]. In the reference model, 10 columns of racks are in the warehouse. Each column consists of 19 racks standing next to each other and every rack has 10 shelves one above another to store pallets (0 indicates standing on the floor). The warehouse space is divided into x equal sized cells, where the cell size was chosen in view of the fact that it coincided with the largest dimensions of the trucks. The velocity is the most important parameter of trucks and it is a central parameter of the time simulation when moving commodities through the warehouse. Time delay with the imposition of the floor rack is now negligible. This implies that the velocity of the truck has in this basic benchmark the most significant impact on the time of processing of the whole buffer of jobs.

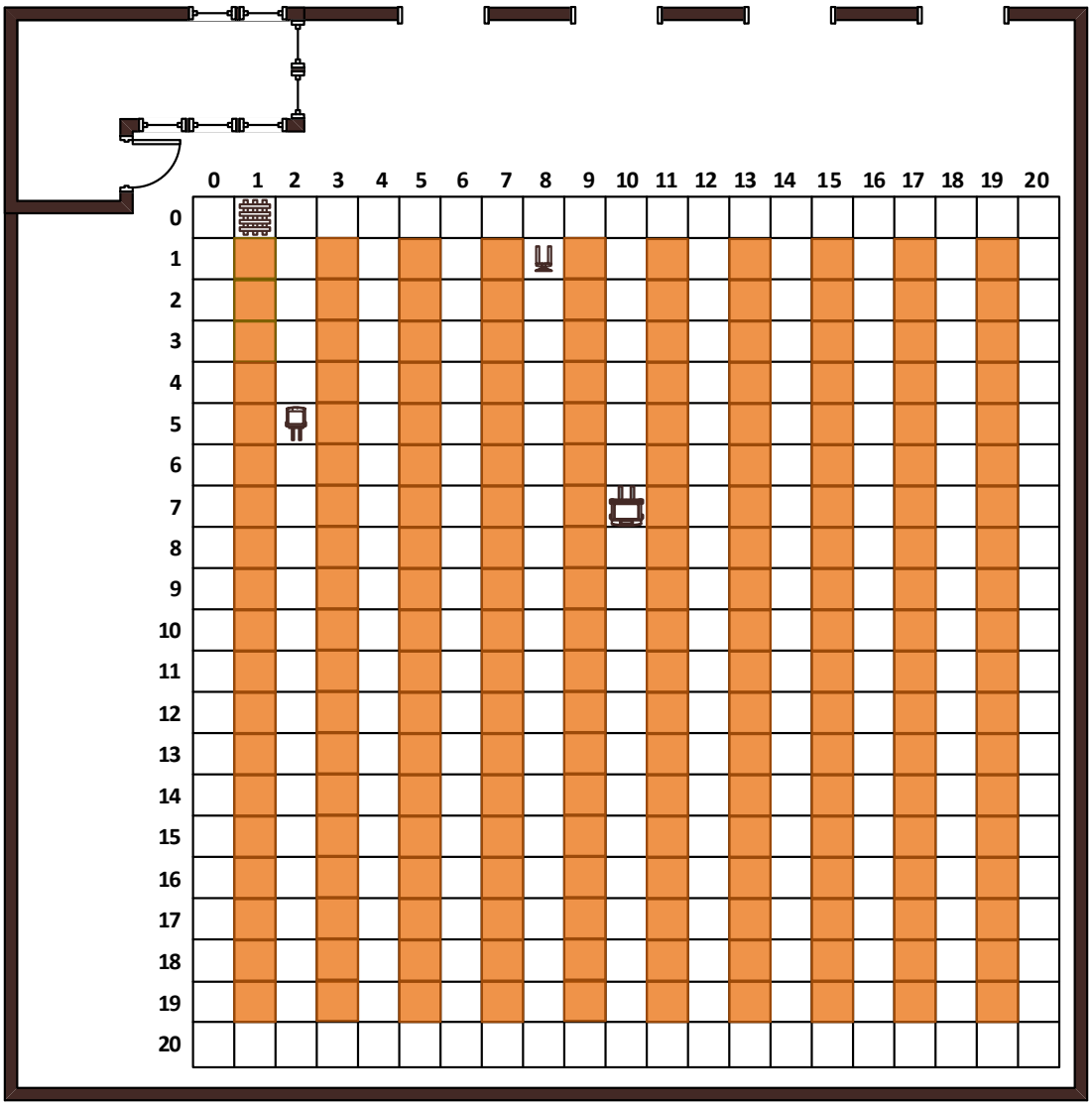


Fig. 8.1: An example of the reference warehouse environment.

8.2 Definition of Benchmarking

The basic assumption for the benchmarking of designed optimization algorithm was the setting of the basic reference level of results – the baseline which was used in further experiments for the comparison of the state of the art of real-world warehouse environments and the proposed optimization algorithm. Since the expert consultant was fully available, and the historical data from the warehouse work-flow were also obtained, the benchmarks were developed on these foundations.

The data obtained were summarized and exported from the SAP information system. The data represent the full use of the warehouse during the last two weeks before the Christmas time. The data were provided with expert's comments and further backstage knowledge of the expert. The reference level should be set as

simply as possible and it should comprehensively describe the state of the art before the implementation of the new system. It serves as a starting point for evaluating the results of the new system and the impact of the implementation and deployment of the new system into the warehouse environment. The result should be the noticeable differences between the initial state and the state after the system implementation.

The established reference level also describes the state of the warehouse in every scenario, e.g. the number of pallets entering the warehouse, as well as the number of pallets leaving the warehouse, all of that in a specified unit of time (in ten minutes, in one hour, in one working shift, in the whole day, etc. – the working shift means eight hours of work, the whole day means three consecutive working shifts). Furthermore, there are also fixed data related to the individual employee, such as processing speed of the jobs, reliability and performance. Likewise, the data on storage resources was also recorded and stored (e.g trucks used in the warehouse, their parameters, especially velocity, handling characteristics and suitability of particular resources for different operations).

Furthermore, the data related to job roles of individual employees were recorded, as well as associated privileges to the equipment. Based on these data, so far the jobs were distributed by supervisor – the operational manager, who was trying to maximize the utilization of all resources (including employees) with regard to the priority of the job – it means that the operator takes into account only a time factor in which it is necessary to complete the job (such as loading and unloading trucks, store specific commodities in the freezer, etc.).

Currently, when the jobs are scheduled by the operational manager, one factor is very noticeable, the stress. When the operational manager is in tense situations the scheduling of work is done very inefficiently. Reserve employees are mobilized and the utilization of employees is not done optimally. With the new optimization algorithm which should help the operator, it will be easy to compare the results of operator's scheduling and the result of the proposed system. At first glance, it will be obvious how to set the evolutionary core of the optimization algorithm and how it affects the results of the optimization.

8.3 Standardization, Normalization

The standardized warehouse has also standardized operations with fixed time norms. The fixed time norms were made by expert consultant based on the veritable time of processing measured in the warehouse several weeks and averaged. The standardized operations and normalized time of these operations are described in Tab. 8.1.

There are also standardized roles for employees. Performance of a specific role is

Tab. 8.1: Standardized operations in the warehouse environment.

ID	Operation	Required	Fix Time Norm
1	Unloading	Always	2
2	Receiving	Always	4
3	Partial Transport	Sometimes	Distance / Velocity
4	Storing, level 0	Always	2
5	Storing, level 1–2	Sometimes	4
6	Storing, level 3–9	Sometimes	7
7	Shifting up to 30 m	Sometimes	Distance / Velocity
8	Shifting beyond 30 m	Sometimes	Distance / Velocity
9	Picking, heterogeneous	Always	15
10	Picking, homogenous	Sometimes	2–7, based on level
11	Dropping, level 1–2	Sometimes	2
12	Dropping, level 3–7	Sometimes	7

normalized based on historical data of performance of single employees, see Tab. 8.2. The heading of Tab. 8.2 represents the *Role* of an employee, the normalized performance (*Perf.*) of the role, and the numbers of operations (*On*). The table characterizes the roles of employees in the way of suitability, it means which operation is the most suitable for the specific role. The suitability is described in Tab. 8.3.

Tab. 8.2: Standardized roles of employees in the warehouse environment.

Role	Perf.	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12
Handler jr.	1	4	1	3	4	3	2	3	4	2	3	3	3
Handler sr.	3	3	4	3	3	4	4	3	2	2	2	2	3
Storeman jr.	1	3	3	3	2	2	2	3	3	3	4	4	3
Storeman sr.	3	2	2	3	2	2	2	3	2	4	3	3	3
Shift leader	5	1	2	1	1	1	1	1	2	1	2	1	1

Tab. 8.3: The suitability table for the roles of employees and operations.

Suitability	Value
May not	1
Unsuitable	2
Suitable	3
Best suitable	4
Not necessary	5

Based on the layout of warehouse, standardized operations, and normalized times, the ground for benchmarking and testing of optimization was set.

8.4 The Test Set – Real Data

On the basis of the experimental integration research related to benchmarking definition several basic scenarios representing real situations from the warehouse work-flows were defined. The scenarios were divided into several specific sets that form logical groups of benchmarks for measuring the performance of the proposed optimization algorithm which can be easily compared to the results achieved by the operational manager. First, we get the data, as the jobs were processed, scheduled respectively, by the operational manager. As it was already written above, these data were extracted from the warehouse work-flows from the pre-Christmas time, when the operational manager was most burdened by stress and fatigue. These scenarios represent a set of jobs which were distributed to employees during a working shift. Solving these scenarios by a skilled operational manager is the reference point for the proposed optimization algorithm. During the experimental integration of the optimization algorithm 60 scenarios were extracted. These scenarios were divided into 5 categories according to the intensity of work load for employees and the level of difficulty of scheduling for operational manager. The scenarios are described in the following five sub-sections and they are accompanied by suitable figures.

8.4.1 Scenarios no. 01–10

The first set of scenarios represents the most simplified cases extracted from the operational data. These scenarios are simplified in such a way that all the trucks in the warehouse are the fork-lift hand pallet trucks. This simplification was introduced because these trucks are used in all types of warehouse environments and it is possible to demonstrate the performance of the proposed optimization algorithm on the simplest types of problems where the optimization of work-flow is not so complicated and it could be easily done only by looking at the problem without the aid of a mathematical or software apparatus. The results obtained in these examples have demonstrated the competitiveness of the algorithm even though the scenarios encompassed the minimal space for optimization.

This set of scenarios includes the most simple and realistic scenarios which are defined as follows. Each truck is the fork-lift hand pallet truck. Each employee performs one simple task from the beginning to the end. The collision of trucks, the distance between the job and the employee, employee's performance, and truck's velocity are not taken into account, as they are all of the same type. An illustrative example is described in the scenario in Tab. 8.4 and is accompanied by visual representation of the scenario in Fig. 8.2.

Tab. 8.4: An example of the simplest set of scenarios no. 01–10.

Scenario no. 09	
Employees	3 x Handler jr. – coordinates [4, 6]; [6, 4]; [12, 2]
Equipment	3 x Fork-lift hand truck (same for all employees)
Description	The first employee (red) loads the pallet on cell [1, 0] and stores on shelf [0, 8]. The second employee (blue) loads the pallet on cell [5, 0] and stores on shelf [11, 7]. The third employee (green) loads the pallet on cell [11, 0] and stores on shelf [7, 5]. All employees work simultaneously.

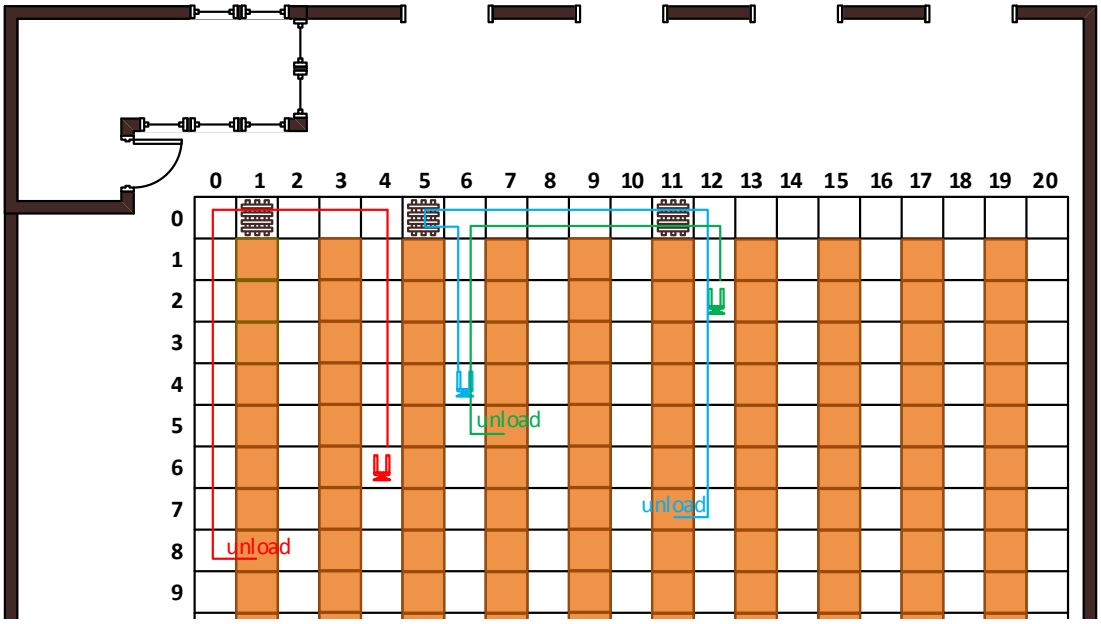


Fig. 8.2: An example of the scenario no. 09 from the first set.

8.4.2 Scenarios no. 11–20

The second set of scenarios also includes realistic scenarios of the warehouse workflow. These relatively simple examples contain two types of trucks – fork-lift hand pallet trucks and fork-lift low pallet trucks, which are able to store goods on shelves on level 1 and 2 as well as on the ground level 0.

This set of scenarios is defined as follows. The trucks are of two types, as mentioned in the previous paragraph. Each employee performs one simple task, collisions of trucks are not taken into account, as well as the distance between the job and the employee. The velocity of truck is a key parameter since two types of trucks are used. Deployment of various truck types can also bring visible improvements in terms of time processing of jobs even in simple scenarios. At the scale of entire shift

in the warehouse, many such simple optimizations can yield significant reductions in the time required for processing the job buffer. A set of scenarios shares the composition of trucks, employees, pallets and coordinates of jobs with the previous set of scenarios. The example scenario from this set is described in Tab. 8.5 and it can be seen in Fig. 8.3.

Tab. 8.5: An example of the set of scenarios no. 11–20.

Scenario no. 17	
Employees	1 x Handler sr. – coordinates [10, 0] 1 x Store-man sr. – coordinates [12, 1]
Equipment	1 x Fork-lift hand truck – coordinates [10, 0] 1 x Fork-lift low truck – coordinates [12, 1]
Description	The first employee (red) loads the pallet on cell [1, 0] and stores on shelf [7, 7]. The second employee (blue) loads the pallet on cell [5, 0] and stores on shelf [7, 8]. All employees work simultaneously.

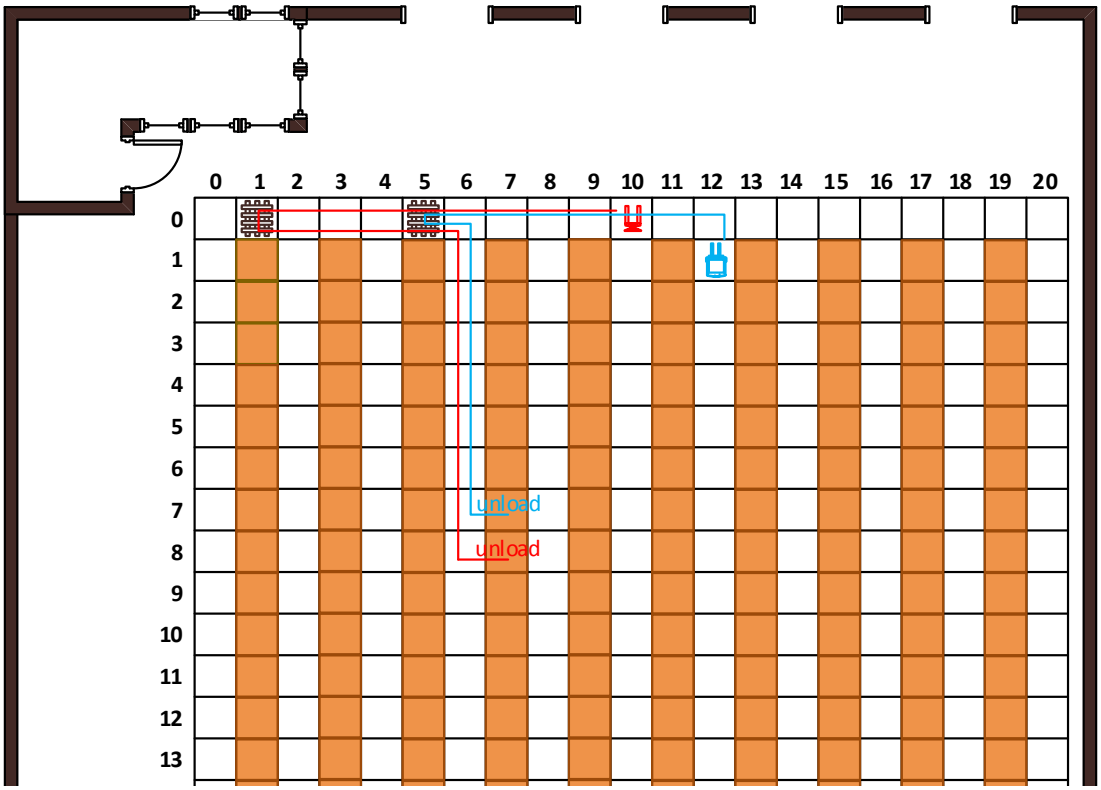


Fig. 8.3: An example of the scenario no. 17 from the second set.

8.4.3 Scenarios no. 21–30

The third set of scenarios represents more difficult situations or situations of the previous sets of scenarios which were extended by further conditions. This set of scenarios is defined as follows. The scenarios contain three types of trucks, the well-known fork-lift hand pallet truck, the fork-lift low pallet truck and the fork-lift high pallet truck. The fork-lift high pallet trucks are able to operate in racks with the goods on the shelves up to level 9. These trucks are proportionately slower when moving from place A to place B then other trucks, because they are more robust.

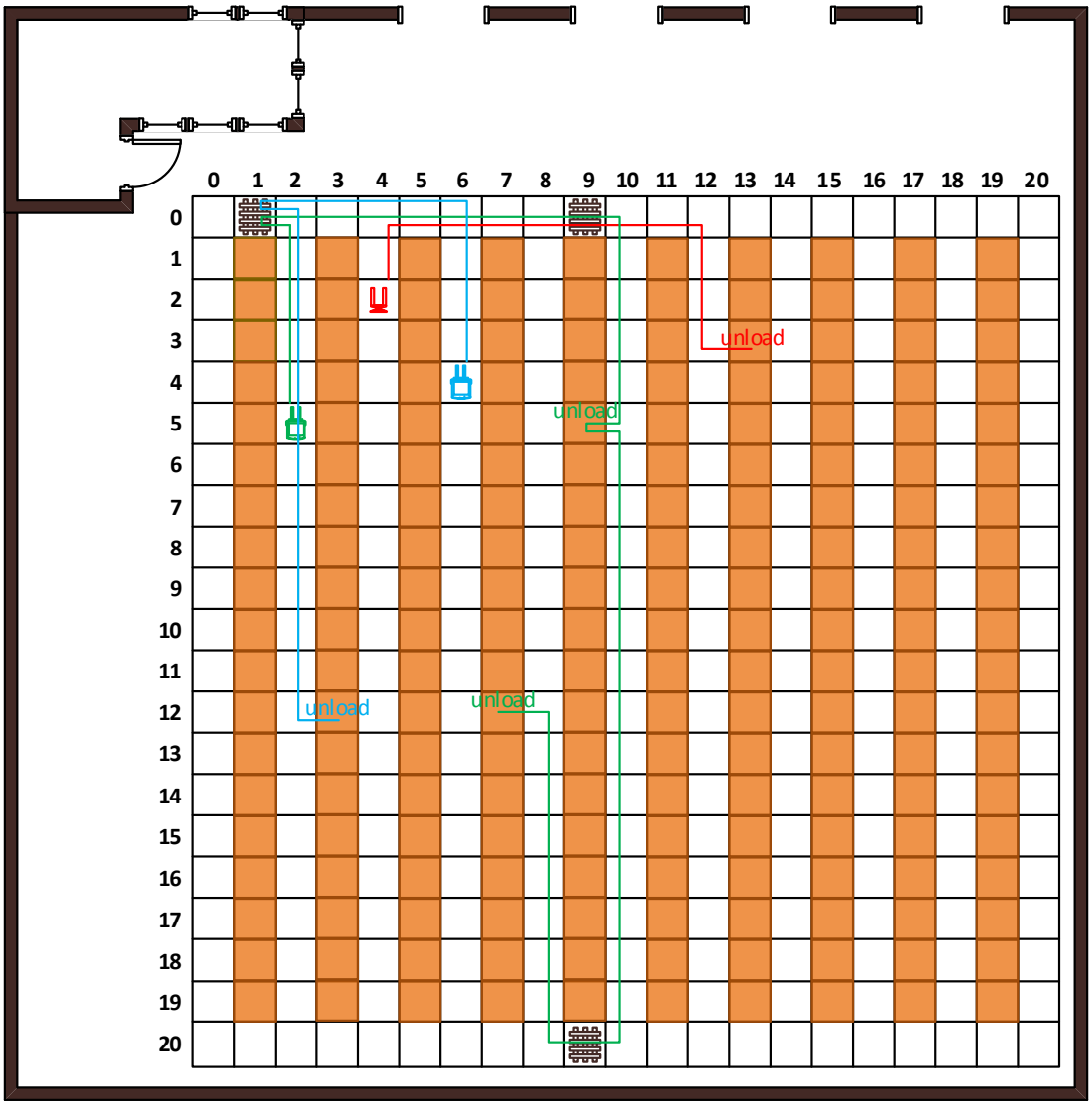


Fig. 8.4: An example of the scenario no. 25 from the third set.

Furthermore, each employee performs only one simple task. The algorithm takes into account the distance of the job and the employee, employee's performance and truck's velocity. Thus, it is ensured that the employee with the fork-lift hand pallet

truck will not pass the routes disproportionately long compared to motorized pallet trucks. This set of scenarios does not take into account collisions of the trucks yet. The illustrative scenario is described in Tab. 8.6 and in Fig. 8.4.

Tab. 8.6: An example of the set of scenarios no. 21–30.

Scenario no. 25	
Employees	1 x Handler jr. – coordinates [4, 2] 1 x Store-man jr. – coordinates [6, 4] 1 x Store-man sr. – coordinates [2, 5]
Equipment	1 x Fork-lift hand truck – coordinates [4, 2] 2 x Fork-lift low truck – coordinates [6, 4]; [2, 5]
Description	The first employee (red) loads the pallet on cell [9, 0] and stores on shelf [13, 3]. The second employee (blue) loads the pallet on cell [1, 0] and stores on shelf [3, 12]. The third employee (green) loads the pallet on cell [1, 0] and stores on shelf [9, 5]. Then, the third employee moves to cell [9, 20] and loads the pallet and stores it on shelf [7, 12]. All employees work simultaneously.

Tab. 8.7: An example of the set of scenarios no. 31–40.

Scenario no. 33	
Employees	1 x Handler jr. – coordinates [4, 2] 1 x Store-man jr. – coordinates [6, 4] 2 x Store-man sr. – coordinates [2, 5]; [0, 8]
Equipment	1 x Fork-lift hand truck – coordinates [4, 2] 3 x Fork-lift low truck – coordinates [6, 4]; [2, 5]; [0, 8]
Description	The first employee (red) loads the pallet on cell [9, 0] and stores on shelf [13, 3]. The second employee (violet) loads the pallet on cell [1, 0] and stores on shelf [3, 12] then moves to cell [3, 20], loads the pallet and stores on shelf [7, 13]. The third employee (green) loads the pallet on cell [1, 0] and stores on shelf [9, 5]. Then, the third employee moves to cell [9, 20], loads the pallet and stores it on shelf [7, 12]. The fourth employee (blue) loads another pallet on cell [3, 20] and stores on shelf [7, 3], then moves to cell [9, 0], where another pallet is waiting, and stores it on shelf [15, 9]. All employees work simultaneously.

8.4.4 Scenarios no. 31–40

The fourth set of scenarios also contains quite complex examples. These are new scenarios and also more advanced scenarios from the previous sets (the same scenarios but in a wider time scale). Every employee in this set of scenarios fulfills one or more jobs. Furthermore, the distance between the employee and the job and the distance between the job and the prospective job are taken into account. In addition, the performance of employees and the truck velocity are also covered as in the previous sets of scenarios. In this set of scenarios, for the first time, collisions of trucks are taken into account and the algorithm is trying to avoid them, or penalize solutions containing collisions. Notice that the trucks overlapping in the aisles around the perimeter of the warehouse are not considered as collisions, since they are wide aisles where the trucks can avoid each other (see Fig. 8.5 and Tab. 8.7).

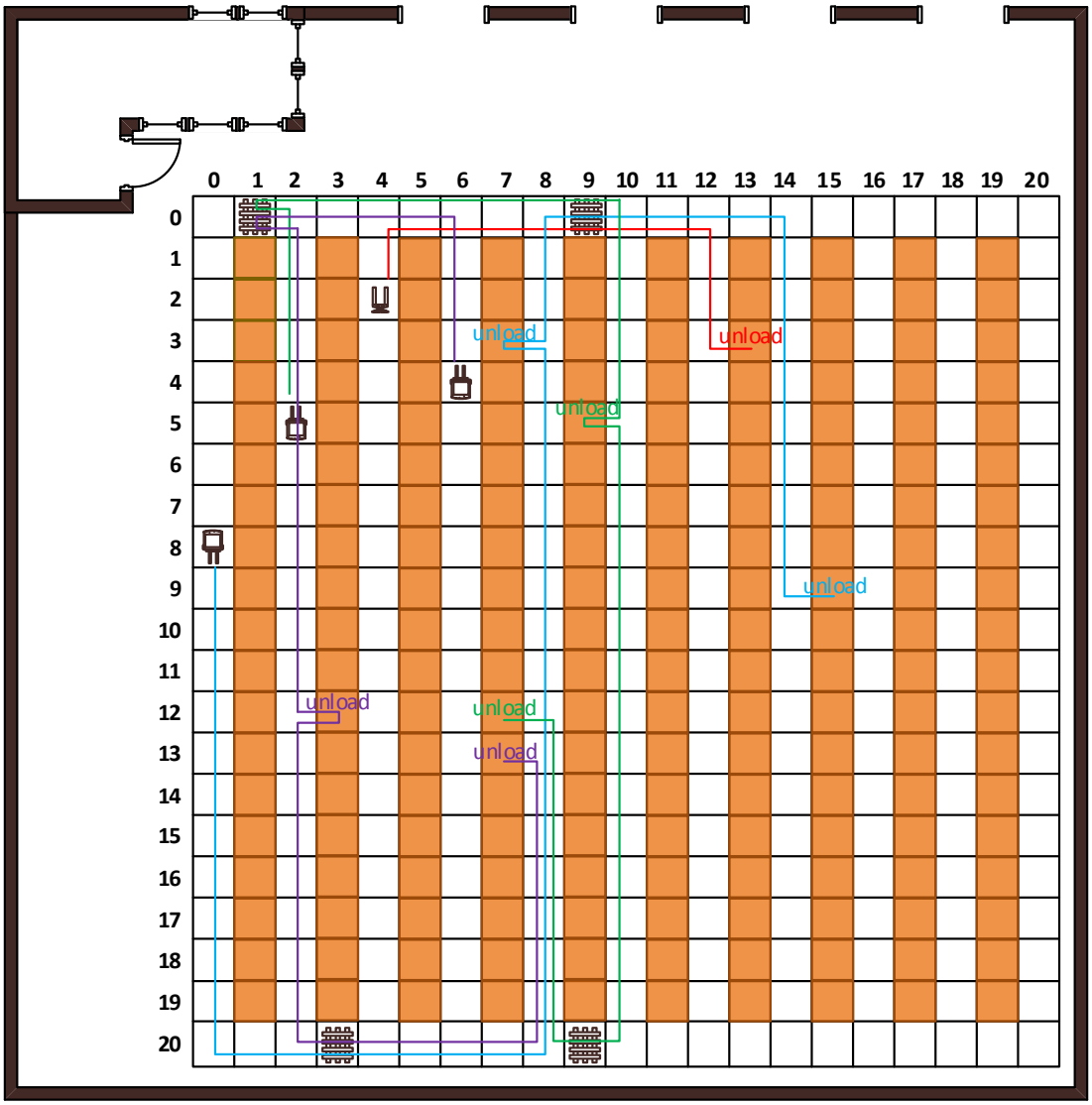


Fig. 8.5: An example of the scenario no. 33 from the fourth set.

8.4.5 Scenarios no. 41–60

The last set of scenarios contains twenty most complex and difficult cases representing the warehouse work-flow, which are characterized mainly by the possibility of cooperation of two or even more employees on one single job. The scenarios may include trucks of all three types mentioned in the paragraphs above. Each employee performs one single job or a list of jobs with the possibility of job sharing.

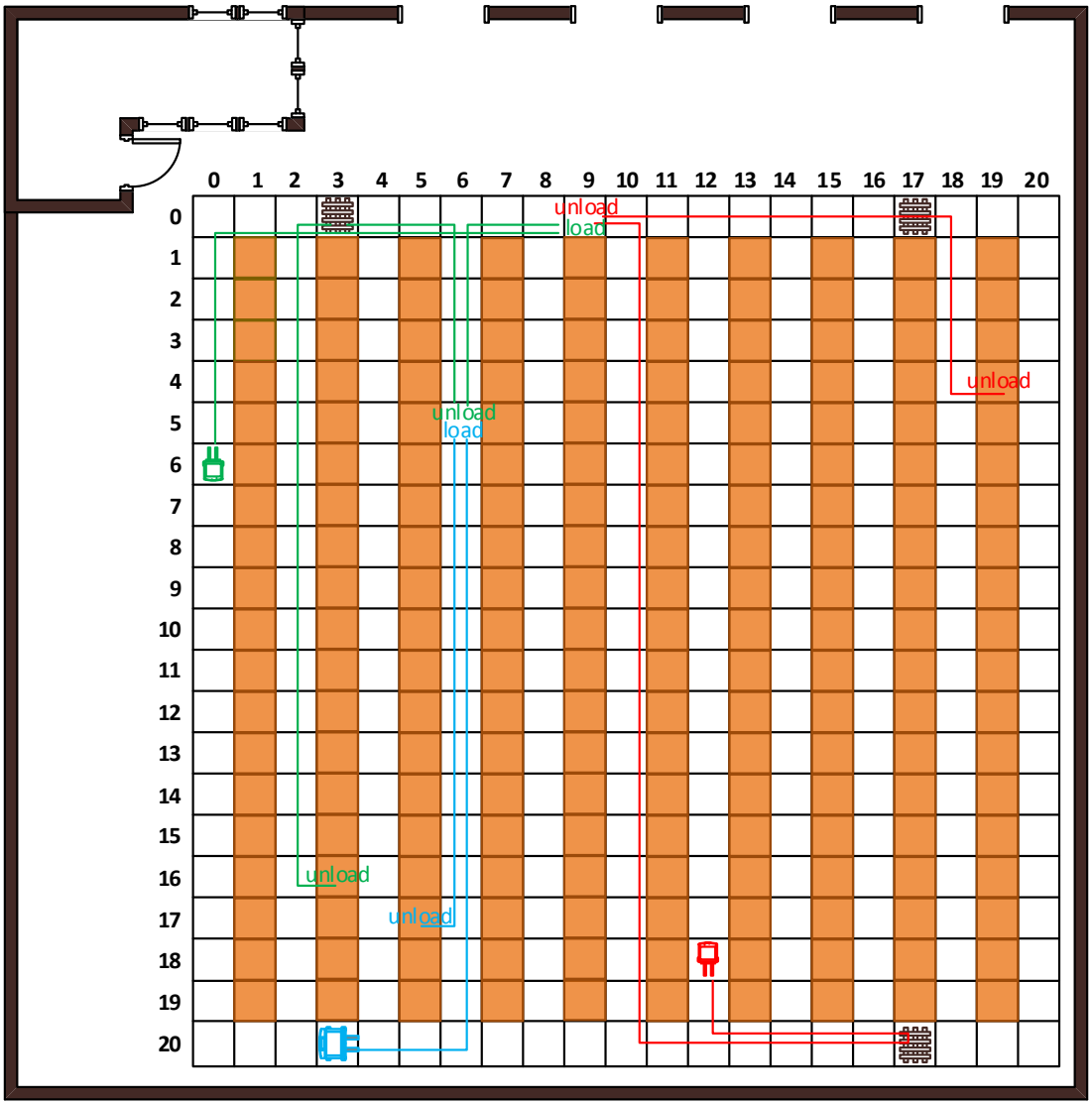


Fig. 8.6: An example of the scenario no. 54 from the fifth set.

Furthermore, the distance between the employee and the job and the distance between the job and the prospective job are taken into account. In addition, the performance of employees and the truck velocity are taken into account in the previous set of scenarios. In this set of scenarios collision of trucks are again computed and taken into account when the fitness function computes the suitability value.

The optimization algorithm tries to avoid all possible collisions, or penalize such a solution. Trucks overlapping in the aisles around the perimeter of the warehouse are not considered as collisions, since they are wide aisles where the trucks can avoid each other. This set of scenarios model the worst case examples of the logistic warehouse work-flow without any simplification or condition relaxation. The illustrative example is depicted in Fig. 8.6 and described in Tab. 8.8.

Tab. 8.8: An example of the set of scenarios no. 41–60.

Scenario no. 54	
Employees	2 x Store-man jr. – coordinates [12, 18]; [0; 6] 1 x Store-man sr. – coordinates [3, 20]
Equipment	2 x Fork-lift low truck – coordinates [12, 18]; [0; 6] 1 x Fork-lift high truck – coordinates [3, 20]
Description	The first employee (red) loads the pallet on cell [17, 20] and stores it on intermediate cell [9, 0] for processing by another employee, then he moves to cell [17, 0], loads the pallet and stores it on shelf [19, 4]. The second employee (green) loads the pallet on intermediate cell and moves it to cell [6, 5], then he continues to cell [3, 0] loads the pallet and stores it on shelf [3, 16]. The third employee (blue) loads the pallet on intermediate cell [6, 5] and stores it on shelf [5, 17]. All employees work simultaneously.

8.5 The Test Set – Synthetic Data

Since the scenarios described in the previous section are quite short (they consist of the units jobs), there is a need to construct more complex testing data consisting of dozens or hundreds of jobs. This led to the design and implementation of the synthetic data set generator which is able to generate such tests based on a few input parameters. The generator consists of four classes, such as the *NameGenerator* for generating the fictional names of employees, the *GeneratorConfig* which allows to set the values of parameters, the *Generator* which generates single parts of the warehouse work-flow, such as employees, equipment, commodities, and a list of jobs. The last class is the *BatchFile* class which runs the synthetic data generator and generates tests. The generator uses the same warehouse layout as the real data set of scenarios. The simplified class diagram is depicted in Fig. 8.7.

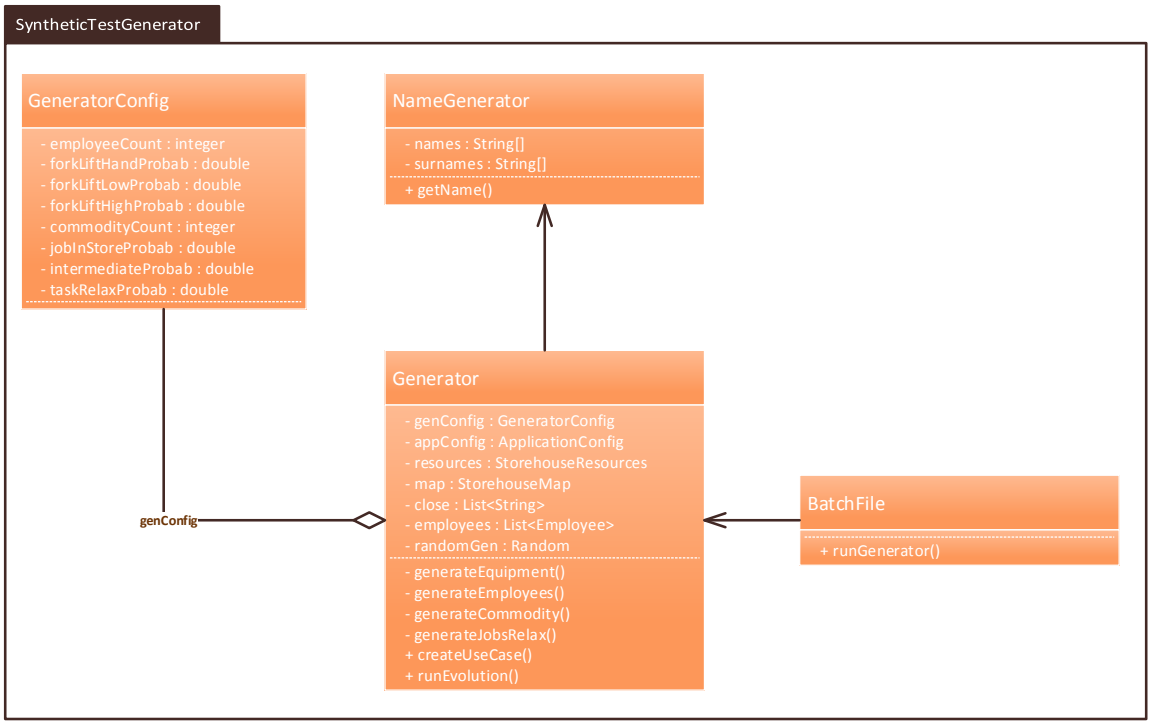


Fig. 8.7: The class diagram of the synthetic test set generator.

The most interesting part of the generator is the *Generator* class which operates on the basis of several configuration parameters which are described in Tab. 8.9. At first, the given number of employees is generated. The employees are assigned the starting coordinates $[x,\ y]$ which are always placed in some aisle, given by the formula Eq. 8.1:

$$x\ = 2 \times (i\ mod\ \frac{w\ +\ 1}{2}), \tag{8.1}$$

where i stands for the number of employees which should be generated, decreased by the number of already generated employees, and w stands for the width of the warehouse (the number of cells in y axes), which is 21 in this case. The coordinate y is generated at random in the range of the aisle. All coordinates which have been already generated are placed in the *close* list, which prevents that two or more employees will share the same cell on the map of warehouse. During the generation process, the employee is also assigned a random name, surname, and a truck. The type of truck is selected based on the input parameters of probability described in Tab. 8.9. Notice that the probability here is the probability of truck occurrence of all trucks. So, the 50% probability of fork-lift hand truck means that about 50 % of all generated trucks are fork-lift hand trucks. The sum of all probabilities is 1.0.

The second part of the synthetic test generator deals with the commodity and job generation. The commodity represents only one pallet. The start point as well as the end point of commodity is generated and assigned to the newly generated job. The number of jobs is directly connected to the number of commodities. Furthermore, the type of job is generated, i.e. if the job is the in store or out store job, and there is also possibility that the commodity is placed on an intermediate cell, and that it must be moved in or out the store. The start point of commodity is generated by the same formula as the employee's start point. The end point of the job can be the rack, intermediate cell, or the warehouse gate, which is randomly selected.

Tab. 8.9: The configuration parameters of the synthetic test set generator.

Parameter	Description
EmployeeCount	The number of employees.
ForkLiftHandProbab	The probability of fork-lift hand truck.
ForkLiftLowProbab	The probability of fork-lift low truck.
ForkLiftHighProbab	The probability of fork-lift high truck.
CommodityCount	The number of commodities.
JobInStoreProbab	The probability that the job is the in store job.
IntermediateProbab	The probability of where the commodity is placed.
TaskRelaxProbab	The probability of *TaskRelax* occurrence in jobs.

9 MEASUREMENT AND VALIDATION

This chapter describes the results of benchmarking. Section 9.1 describes results of the real data set measured by the optimization algorithm and the comparison with the results reached by the operational manager. The results of the real data set proved the competitiveness of the system with the results reached by the operational manager. Section 9.2 gives the results reached on the synthetic data sets. These results of measurement are a great source for further improvement and optimization of the proposed optimization algorithm, since the synthetic data are much more complex and difficult than the data in the real data set.

9.1 The Results of Real Data Set

The parameters for controlling the run and the parameter representing the stop condition were set according to the values depicted in Tab. 9.1. The precision of the fitness function was not set, because this parameter is not used in the evolutionary optimization algorithm adapted for the warehouse optimization problem.

All the measurements were applied to 20 individuals in 20 generations, which was quite enough for a crashing majority of scenarios to outperform the results reached by the operational manager or at least to reach the same level of performance. The elitism was set to 5 %, so only the best individual of the population will be copied to the new population without change. The rates of other operators were set to 60 %, which gives a better chance to breed the population. The Split Job Mutation was set to 10 % because this operator is applicable mostly in the most complex cases, otherwise this operator could possibly cause a slower convergence to the solution in simple cases. The high rate of mutations causes high computational demands, but it is helpful during the breeding process. Finally, only 20 individuals survive in each generation after the decimation process.

Tab. 9.1: The settings for controlling the run of the GP algorithm.

Population Size	20 individuals
Evolution Size	20 generations
Fitness Precision	not set
Elitism Rate	5 %
PA Mutation Rate	60 %
JO Mutation Rate	60 %
SJ Mutation Rate	60 %
SW Mutation Rate	60 %
SP Mutation Rate	10 %

The first line (Operational Manager (OM)) in all tables (Tab. 9.2 – Tab. 9.7) in the following five subsections represents the results reached by the operational manager, which were set as the reference baseline. Other lines represent the measurements of the optimization algorithm. The measurements differ in the combinations of the used genetic operators. The best results reached by the optimization algorithm are depicted in the orange colored box. Other results, the same level or the worst level in comparison to the reference baseline, are depicted in the black colored text. If the result reached by OM was outperformed or the same result was reached, the OM result is in the red colored box. If the result reached by OM is better than all results reached by the optimization algorithm, the OM result is in the green colored box. This coloration is valid for all measurements in sections 9.1 and 9.2.

During the process of benchmarking and testing, all genetic operators were tested alone, to see the performance of each single operator. Then, all pairs of genetic operators were tested, all the triplets of operators, all the quartets of operators, and in the last run of the algorithm all operators were tested together.

The results depicted in the following subsections show that each genetic operator itself is competitive with the results of the operational manager. The combination of genetic operators was used to get a wider variability in individuals across the population and to gain more variants of solutions. In general, it can be concluded that the use of all operators together reduces the computational performance, but the results of the computations are better than the results of single operators, especially in more complex and difficult situations.

During the development of the optimization algorithm various settings of the algorithm were tested. Various sizes of population and various numbers of generation steps were given under investigation. The 20 individuals in the population and 20 generation steps seem to be an optimal setting with respect to the computational time and the precision of results for the purpose of testing scenarios 1 – 60. The elitism rate was also tested, but when the population size is 20 individuals, 5 % (one individual) is sufficient to maintain a non-decreasing character of the fittest individual during the evolution process with the preserved diversity of population.

Furthermore, the rate of all mutation operators was given also under investigation. It was proved that the rate of the mutation is not a key parameter as well as the parameter used in the evolution process when the population size is only 20 individuals. Only one parameter is significantly different: the rate of the Split Job Mutation. If the scenarios tested are quite simple, the operator can literally break the jobs in to small sub-jobs and distribute them across the employees. This behavior of the algorithm is not required in such simple tasks.

9.1.1 Scenarios no. 01–10

This subsection describes the results of the first ten scenarios. The results are depicted in Tab. 9.2. As it can be seen from the table, all scenarios except no. 6 was outperformed at least by one single genetic operator or a combination of several genetic operators. This gives 90 % precision of the first test set of scenarios. It must be said that such good results were not expected in the simple scenarios like these. The set represents only very simple scenarios where there is only a very limited space for any optimization. The results in the table suggested that certain optimization of scheduling can be done even in very simple cases.

Tab. 9.2: The results of the measurement of scenarios no. 01–10.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	7.00	8.17	7.00	8.17	6.83	7.50	7.67	5.67	7.00	8.17
PA	7.00	8.17	8.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
JO	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
SJ	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
SW	7.00	8.17	12.17	7.50	6.67	11.50	7.00	5.33	6.83	8.00
SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
PA,JO	7.00	8.17	8.17	8.17	6.83	10.67	7.67	5.33	6.83	8.00
PA,SJ	7.00	8.17	8.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,SW	7.00	8.17	7.00	8.17	6.67	7.83	7.67	5.33	6.83	8.00
PA,SP	7.00	8.17	8.50	8.17	6.83	7.83	7.67	5.33	6.83	8.00
JO,SJ	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
JO,SW	7.00	8.17	12.17	8.17	6.83	7.67	7.00	5.33	6.83	8.00
JO,SP	7.00	8.17	13.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
SJ,SW	7.00	8.17	13.17	8.17	6.67	7.67	7.00	5.33	6.83	8.00
SJ,SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
SW,SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
PA,JO,SJ	7.00	8.17	8.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,JO,SW	7.00	8.17	6.50	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,JO,SP	7.00	8.17	10.00	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,SJ,SW	7.00	8.17	7.00	8.17	6.83	7.83	7.00	5.33	6.83	8.00
PA,SJ,SP	7.00	8.17	7.00	8.17	6.83	11.00	7.67	5.33	6.83	8.00
PA,SW,SP	7.00	8.17	7.00	8.17	6.83	7.83	7.67	5.33	6.83	8.00
JO,SJ,SW	7.00	8.17	7.00	8.17	6.83	7.67	7.67	5.33	6.83	8.00
JO,SJ,SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
JO,SW,SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	7.00
SJ,SW,SP	7.00	8.17	12.17	8.17	6.67	11.50	7.67	5.33	6.83	8.00
PA,JO,SJ,SW	7.00	8.17	12.83	8.17	6.83	7.67	7.67	5.33	6.83	8.00
PA,JO,SJ,SP	7.00	8.17	12.83	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,JO,SW,SP	7.00	8.17	7.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
PA,SJ,SW,SP	7.00	8.17	8.17	8.17	6.83	7.83	7.00	5.33	6.83	7.00
JO,SJ,SW,SP	7.00	8.17	12.17	8.17	6.83	11.50	7.67	5.33	6.83	8.00
All operators	7.00	8.17	7.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
Average	7.00	8.17	10.01	8.15	6.81	9.36	7.56	5.34	6.84	7.94
Variance	0,00	0,00	6,06	0,01	0,00	3,27	0,06	0,00	0,00	0,06
Deviation	0.00	0.00	2.46	0.12	0.06	1.81	0.24	0.06	0.03	0.25
Mode	7.00	8.17	12.17	8.17	6.83	7.83	7.67	5.33	6.83	8.00
Minimum	7.00	8.17	6.50	7.50	6.67	7.50	7.00	5.33	6.83	7.00
Maximum	7.00	8.17	13.17	8.17	6.83	11.50	7.67	5.67	7.00	8.17

9.1.2 Scenarios no. 11–20

This subsection shows the results of the second dozen of scenarios. The results are depicted in Tab. 9.3. As it can be seen from the table, all the scenarios except no. 2 and no. 6 were outperformed, or the same level of performance was reached, at least by one single genetic operator or a combination of several genetic operators. This gives 80 % precision of the proposed optimization algorithm of the second set of scenarios. As this is still quite a simple set of scenarios, no big improvement of results was expected in comparison to the operational manager, but still, the algorithm showed the competitiveness.

Tab. 9.3: The results of the measurement of scenarios no. 11–20.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	6.25	7.33	7.00	7.33	6.33	7.50	7.67	5.50	7.00	8.17
PA	6.25	7.67	8.00	7.33	6.38	7.88	7.13	5.33	6.67	7.00
JO	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
SJ	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
SW	6.25	7.67	10.88	5.13	6.38	10.63	7.13	5.33	6.67	7.00
SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
PA,JO	6.25	7.67	8.50	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,SJ	6.25	7.67	7.17	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,SW	6.25	7.67	7.00	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,SP	6.33	7.67	8.50	7.33	6.38	7.83	7.13	5.33	6.83	7.00
JO,SJ	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
JO,SW	6.25	7.67	10.88	7.33	6.33	10.63	7.13	5.33	6.67	7.00
JO,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
SJ,SW	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
SJ,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
SW,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.13	7.00
PA,JO,SJ	6.25	7.67	7.00	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,JO,SW	6.25	7.67	7.00	7.33	6.38	7.83	6.50	5.33	6.67	7.00
PA,JO,SP	6.25	7.67	7.00	7.33	6.38	7.88	7.13	5.33	6.67	7.00
PA,SJ,SW	6.25	7.67	7.17	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,SJ,SP	6.25	7.67	7.17	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,SW,SP	6.25	7.67	9.50	7.33	6.33	7.83	7.13	5.33	6.67	7.00
JO,SJ,SW	6.25	7.67	10.88	7.33	6.38	10.63	6.50	5.33	6.67	7.00
JO,SJ,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
JO,SW,SP	6.25	7.67	10.88	7.33	6.33	10.63	7.13	5.33	6.67	7.00
SJ,SW,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
PA,JO,SJ,SW	6.25	7.67	7.17	7.33	6.38	7.83	6.50	5.33	6.67	7.00
PA,JO,SJ,SP	6.25	7.67	8.50	7.33	6.38	7.83	7.13	5.33	6.67	7.00
PA,JO,SW,SP	6.25	7.67	7.17	7.13	6.38	7.83	7.13	5.33	6.67	7.00
PA,SJ,SW,SP	6.25	7.67	7.00	7.33	6.38	7.83	6.50	5.33	6.67	7.00
JO,SJ,SW,SP	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
All operators	6.25	7.67	7.17	7.33	6.38	7.88	7.13	5.33	6.67	7.00
Average	6.25	7.66	9.10	7.26	6.37	9.14	7.06	5.34	6.67	7.04
Variance	0,00	0,00	3,08	0,15	0,00	1,96	0,05	0,00	0,01	0,04
Deviation	0.01	0.06	1.76	0.38	0.01	1.40	0.23	0.03	0.12	0.20
Mode	6.25	7.67	10.88	7.33	6.38	10.63	7.13	5.33	6.67	7.00
Minimum	6.25	7.33	7.00	5.13	6.33	7.50	6.50	5.33	6.13	7.00
Maximum	6.33	7.67	10.88	7.33	6.38	10.63	7.67	5.50	7.00	8.17

9.1.3 Scenarios no. 21–30

This subsection describes the results of the third dozen of scenarios. The results are depicted in Tab. 9.4. As it can be seen from the table, all the scenarios except no. 9 was outperformed at least by one single genetic operator or a combination of several genetic operators, which gives together 90 % precision of the optimization algorithm of the third set of scenarios. Still, one scenario was not solved at the same level of performance as it was solved by the operational manager.

Tab. 9.4: The results of the measurement of scenarios no. 21–30.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	10.50	11.67	13.75	17.00	13.38	12.25	20.38	21.63	20.38	17.00
PA	10.38	11.67	13.00	11.50	12.63	12.00	19.50	23.13	24.63	16.63
JO	10.00	11.67	14.83	11.50	12.63	11.83	20.75	27.33	24.83	16.63
SJ	9.83	11.67	14.83	11.50	11.88	11.17	19.50	26.88	23.50	16.63
SW	10.00	11.67	14.83	11.50	12.63	12.00	20.88	27.17	21.25	16.63
SP	10.00	11.67	15.33	11.50	12.63	12.00	20.50	27.00	24.83	16.63
PA,JO	10.83	11.67	14.13	11.50	12.50	12.00	20.00	21.75	23.33	15.63
PA,SJ	10.25	11.67	13.00	11.50	11.50	12.00	21.50	21.75	20.75	14.88
PA,SW	10.83	11.67	13.00	11.50	12.63	12.00	19.83	20.63	23.00	17.13
PA,SP	10.00	11.67	13.75	11.50	12.63	12.00	20.63	24.00	22.13	15.38
JO,SJ	9.83	11.67	14.00	11.50	11.88	11.17	20.63	25.83	23.83	14.88
JO,SW	11.00	11.67	14.00	6.38	11.88	12.00	20.13	26.13	21.25	16.13
JO,SP	10.25	11.67	14.83	11.50	12.50	12.00	19.50	27.88	23.33	15.63
SJ,SW	10.00	11.67	10.63	11.50	11.88	12.00	19.88	27.17	23.33	14.88
SJ,SP	9.83	11.67	14.83	11.50	12.50	11.83	19.25	26.63	22.88	15.38
SW,SP	6.83	11.67	14.83	11.50	12.63	12.00	16.33	25.00	23.50	15.38
PA,JO,SJ	9.83	11.67	14.00	11.50	11.88	11.17	19.83	22.13	20.75	15.13
PA,JO,SW	9.83	11.67	13.38	11.50	12.63	11.17	19.25	22.63	22.13	16.63
PA,JO,SP	10.38	11.67	13.75	11.50	11.50	12.00	19.83	22.38	22.88	16.88
PA,SJ,SW	9.83	11.67	13.00	11.50	11.50	12.00	19.25	22.13	21.25	15.63
PA,SJ,SP	9.83	11.67	12.50	11.50	11.50	11.17	18.17	22.38	21.13	16.63
PA,SW,SP	10.00	11.67	12.50	11.50	12.63	12.00	19.88	23.13	22.13	16.63
JO,SJ,SW	9.83	6.00	14.00	6.38	11.88	11.17	19.83	27.00	23.50	14.88
JO,SJ,SP	10.00	11.67	14.83	11.50	11.88	11.17	21.63	27.33	23.33	15.13
JO,SW,SP	10.00	11.67	14.00	11.50	11.88	11.17	16.00	26.88	24.17	16.63
SJ,SW,SP	10.00	11.67	14.00	11.50	11.88	11.83	20.50	26.25	24.00	16.63
PA,JO,SJ,SW	9.83	11.67	12.50	11.50	11.50	11.17	18.75	21.50	20.75	14.88
PA,JO,SJ,SP	10.50	11.67	12.88	11.50	11.88	11.88	18.17	22.13	20.75	15.63
PA,JO,SW,SP	9.83	11.67	12.50	11.50	12.75	12.00	19.83	23.00	23.00	15.38
PA,SJ,SW,SP	9.83	11.67	13.75	11.50	11.50	11.17	20.75	22.88	20.50	16.63
JO,SJ,SW,SP	9.83	11.67	11.63	6.38	11.88	11.38	16.17	27.00	23.75	11.50
All operators	9.83	11.67	12.50	11.50	11.50	11.63	20.63	22.38	22.75	15.88
Average	9.99	11.49	13.60	11.19	12.14	11.70	19.61	24.40	22.61	15.81
Variance	0,43	0,97	1,12	3,31	0,25	0,15	1,84	5,44	1,81	1,14
Deviation	0.65	0.99	1.06	1.82	0.50	0.38	1.36	2.33	1.34	1.07
Mode	9.83	11.67	14.83	11.50	11.88	12.00	19.83	27.00	23.33	16.63
Minimum	6.83	6.00	10.63	6.38	11.50	11.17	16.00	20.63	20.38	11.50
Maximum	11.00	11.67	15.33	17.00	13.38	12.25	21.63	27.88	24.83	17.13

9.1.4 Scenarios no. 31–40

This subsection describes the results of the fourth dozen of scenarios. The results are depicted in Tab. 9.5. In this case, all the scenarios in the set were outperformed for the first time at least by one single genetic operator or a combination of several genetic operators, which is a great result of the optimization algorithm in the fourth set of scenarios. The best results of this set were obtained by the combination of operators, not any operator alone.

Tab. 9.5: The results of the measurement of scenarios no. 31–40.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	14.25	14.25	14.63	16.00	10.75	13.63	20.38	15.33	18.25	14.25
PA	16.00	16.13	15.13	14.00	10.75	16.00	23.38	19.17	24.75	14.00
JO	15.00	18.25	16.63	14.00	10.75	16.25	23.63	20.33	22.67	15.17
SJ	14.83	17.75	16.63	14.00	10.75	16.00	24.00	18.88	21.63	13.17
SW	14.83	16.83	14.63	15.00	10.75	16.13	25.63	21.17	22.50	13.38
SP	14.83	18.38	16.63	14.83	10.75	15.00	28.00	18.88	24.13	15.00
PA,JO	14.83	18.00	13.25	15.67	10.75	14.67	25.50	17.33	23.25	13.00
PA,SJ	14.83	16.25	16.13	15.00	10.75	13.63	22.38	16.83	23.00	14.83
PA,SW	14.25	15.88	16.13	15.00	10.75	12.75	21.75	17.67	20.75	13.38
PA,SP	14.83	18.38	15.75	14.00	10.75	16.00	24.25	18.17	21.67	13.50
JO,SJ	14.83	17.00	17.00	14.00	10.75	14.67	22.63	19.88	22.38	13.38
JO,SW	14.83	16.13	16.63	14.67	10.75	16.00	26.00	20.17	23.13	13.17
JO,SP	15.00	19.50	17.00	14.83	10.75	16.00	23.00	19.33	24.13	13.33
SJ,SW	14.83	16.13	16.63	14.00	10.75	10.25	24.38	18.88	22.33	13.83
SJ,SP	14.83	16.83	17.00	14.00	10.75	14.50	23.88	18.88	22.33	13.17
SW,SP	14.83	19.38	17.00	15.00	10.75	15.00	24.50	21.38	22.83	13.88
PA,JO,SJ	13.13	16.25	15.75	15.00	10.75	13.13	21.25	16.38	20.50	12.63
PA,JO,SW	14.38	16.13	14.13	15.00	10.75	16.13	22.00	17.33	21.67	15.33
PA,JO,SP	15.67	15.00	14.63	15.00	10.75	13.50	25.00	19.50	22.33	13.50
PA,SJ,SW	13.13	14.13	15.25	12.00	10.75	12.75	22.13	18.33	22.50	13.00
PA,SJ,SP	14.83	18.17	15.38	14.83	10.75	13.00	19.38	17.67	20.67	13.33
PA,SW,SP	14.50	16.50	15.88	14.00	10.75	14.50	22.00	13.63	19.50	13.17
JO,SJ,SW	14.83	16.25	15.88	14.00	10.75	13.67	22.17	18.88	22.67	13.17
JO,SJ,SP	14.83	16.13	17.25	15.00	10.75	13.67	22.88	19.33	22.13	13.50
JO,SW,SP	14.83	16.83	18.38	15.00	10.75	15.25	25.75	21.00	24.13	15.00
SJ,SW,SP	12.33	18.67	16.88	15.00	10.75	15.13	22.50	19.38	22.50	13.38
PA,JO,SJ,SW	13.13	14.75	13.83	14.00	10.75	13.00	23.00	18.17	20.88	13.38
PA,JO,SJ,SP	15.00	15.38	16.63	14.00	10.75	13.50	20.50	20.33	20.25	13.50
PA,JO,SW,SP	8.67	16.63	14.88	15.33	10.75	14.33	22.50	20.17	21.67	11.67
PA,SJ,SW,SP	13.13	14.63	14.50	14.00	10.75	13.00	23.00	18.00	21.00	11.88
JO,SJ,SW,SP	14.83	16.25	15.88	14.00	8.38	14.67	21.25	18.88	13.25	13.17
All operators	13.63	15.17	15.50	14.00	10.75	12.83	22.63	17.67	18.75	11.67
Average	14.33	16.62	15.85	14.51	10.68	14.33	23.16	18.65	21.69	13.49
Variance	1,64	1,96	1,27	0,53	0,17	1,90	3,27	2,68	4,48	0,78
Deviation	1.28	1.40	1.13	0.73	0.41	1.38	1.81	1.64	2.12	0.88
Mode	14.83	16.13	16.63	14.00	10.75	16.00	22.63	18.88	22.50	13.38
Minimum	8.67	14.13	13.25	12.00	8.38	10.25	19.38	13.63	13.25	11.67
Maximum	16.00	19.50	18.38	16.00	10.75	16.25	28.00	21.38	24.75	15.33

9.1.5 Scenarios no. 41–60

This subsection describes the results of the last set of scenarios. This set is twice bigger than the previous sets. It contains 20 examples in total. The results are depicted in Tab. 9.6 and Tab. 9.7. In this case, all the scenarios in the set were outperformed at least by one single genetic operator or by a combination of several genetic operators, which give great results in the last set of scenarios. The best results of this set were obtained by the combination of operators to which no single operator can compete. Since these are quite complex scenarios, the best time reached by the optimization algorithm was obtained by the combinations of operators.

Tab. 9.6: The results of the measurement of scenarios no. 41–50.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	18.25	11.50	15.50	11.00	16.38	10.75	26.88	22.25	16.50	22.75
PA	14.25	7.88	8.00	8.00	7.50	7.13	16.88	11.50	8.17	13.00
JO	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
SJ	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
SW	9.50	7.88	8.00	8.00	7.75	7.13	16.88	12.00	8.17	13.00
SP	14.25	7.88	8.00	8.00	7.75	7.13	16.88	12.00	8.17	13.00
PA,JO	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,SJ	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,SW	14.25	7.88	8.00	8.00	7.50	7.13	16.88	11.63	8.17	13.00
PA,SP	14.25	7.88	8.00	8.00	7.50	7.13	16.88	11.50	8.17	13.00
JO,SJ	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
JO,SW	9.50	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
JO,SP	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	13.25
SJ,SW	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
SJ,SP	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.63
SW,SP	14.25	7.88	8.00	8.00	7.75	7.13	16.88	12.00	6.25	13.00
PA,JO,SJ	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,JO,SW	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	13.00
PA,JO,SP	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,SJ,SW	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,SJ,SP	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,SW,SP	14.25	7.88	8.00	8.00	7.50	7.13	16.88	11.50	8.17	13.00
JO,SJ,SW	12.38	7.88	8.00	8.00	7.75	5.67	16.38	12.00	8.17	12.00
JO,SJ,SP	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
JO,SW,SP	12.38	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.88
SJ,SW,SP	9.50	7.88	8.00	8.00	7.75	6.00	16.75	12.00	8.17	12.88
PA,JO,SJ,SW	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.63
PA,JO,SJ,SP	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
PA,JO,SW,SP	12.38	7.88	8.00	8.00	7.50	7.13	16.75	11.50	8.17	12.00
PA,SJ,SW,SP	12.38	7.88	8.00	8.00	7.50	7.13	16.38	10.00	8.17	12.88
JO,SJ,SW,SP	9.50	7.88	8.00	8.00	7.75	7.13	16.38	12.00	8.17	12.00
All operators	12.38	7.88	8.00	8.00	7.50	7.13	16.38	11.50	8.17	12.00
Average	12.55	7.99	8.23	8.09	7.89	7.16	16.84	12.03	8.37	12.75
Variance	2,74	0,40	1,70	0,27	2,34	0,52	3,30	3,52	2,24	3,45
Deviation	1.66	0.63	1.30	0.52	1.53	0.72	1.82	1.88	1.50	1.86
Mode	12.38	7.88	8.00	8.00	7.50	7.13	16.38	12.00	8.17	12.00
Minimum	9.50	7.88	8.00	8.00	7.50	5.67	16.38	10.00	6.25	12.00
Maximum	18.25	11.50	15.50	11.00	16.38	10.75	26.88	22.25	16.50	22.75

As the examples represent really complex and difficult situations, the combinations of genetic operators gave very different results as it can be seen from the results of variance and standard deviation. This means that more difficult situations can result in many possible solutions, but of course, not all of them are considered as good results, even if they outperformed the reference baseline, because there are much better results given by the other combinations of genetic operators.

Tab. 9.7: The results of the measurement of scenarios no. 51–60.

Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
OM	31.88	27.88	33.50	20.00	29.25	27.50	30.13	37.13	26.38	32.17
PA	18.50	26.00	24.88	13.50	21.50	14.75	12.75	17.63	13.63	13.13
JO	20.38	27.13	25.63	15.75	24.50	14.75	12.83	17.63	13.63	12.63
SJ	20.17	26.00	24.88	15.75	24.00	14.75	12.83	17.63	13.63	12.63
SW	21.50	26.00	24.88	15.75	24.00	13.75	12.75	17.83	13.63	13.13
SP	21.50	26.00	26.33	15.75	24.63	14.75	12.83	17.63	13.63	12.63
PA,JO	20.50	23.63	23.63	13.50	23.13	14.75	12.83	17.83	13.63	12.63
PA,SJ	17.50	25.13	25.00	13.50	20.38	14.75	11.50	17.63	13.63	12.63
PA,SW	18.88	25.83	24.00	13.50	21.50	14.75	11.50	17.88	13.63	12.63
PA,SP	18.83	25.50	26.13	13.50	21.50	14.75	12.83	17.63	13.63	12.63
JO,SJ	20.17	25.50	25.63	15.75	23.88	14.75	12.83	17.63	13.63	12.63
JO,SW	22.00	26.00	27.63	15.75	24.00	15.50	12.83	17.63	13.63	12.63
JO,SP	21.50	27.38	27.63	15.75	23.88	14.75	12.83	17.63	13.63	12.63
SJ,SW	20.38	25.83	26.13	15.75	23.88	13.75	12.83	17.63	13.63	12.63
SJ,SP	21.00	26.63	27.88	15.75	24.00	14.75	12.83	17.63	13.63	12.63
SW,SP	21.00	26.00	28.38	15.75	24.00	14.75	12.83	17.63	13.63	13.88
PA,JO,SJ	17.75	24.25	23.88	13.50	21.50	14.75	11.50	17.75	13.63	12.63
PA,JO,SW	18.75	23.83	23.50	13.50	25.00	14.75	11.50	17.88	13.63	12.63
PA,JO,SP	17.25	24.00	25.38	13.50	24.75	14.75	11.50	18.25	13.63	13.13
PA,SJ,SW	18.67	23.75	23.88	11.25	21.50	14.75	11.50	18.25	13.63	12.63
PA,SJ,SP	17.83	23.17	23.00	11.25	23.88	14.75	11.50	18.67	13.63	12.63
PA,SW,SP	18.25	23.50	23.75	11.25	25.75	14.75	11.50	17.63	13.63	13.13
JO,SJ,SW	20.75	21.75	22.33	15.75	24.50	14.75	12.83	17.63	13.63	12.63
JO,SJ,SP	21.00	26.63	24.88	15.75	24.00	14.75	12.83	17.63	13.63	12.63
JO,SW,SP	20.38	27.38	26.13	15.75	24.50	14.75	12.83	17.63	13.63	12.63
SJ,SW,SP	18.25	26.63	28.38	15.75	24.63	14.75	12.83	12.88	13.63	13.13
PA,JO,SJ,SW	15.25	24.50	26.25	13.50	21.50	13.25	12.75	17.63	13.63	12.63
PA,JO,SJ,SP	17.00	23.25	24.00	11.63	22.75	14.75	12.75	17.63	13.63	12.88
PA,JO,SW,SP	19.00	21.25	24.17	13.50	23.38	14.75	11.50	17.63	13.63	12.63
PA,SJ,SW,SP	17.25	25.88	25.75	13.50	19.88	14.75	11.50	17.63	13.63	12.63
JO,SJ,SW,SP	21.33	26.00	27.88	15.75	24.63	14.75	11.50	18.25	13.63	12.63
All operators	15.63	24.13	24.50	11.25	21.50	14.75	11.50	17.83	13.63	12.63
Average	19.69	25.20	25.62	14.42	23.49	15.06	12.86	18.22	14.02	13.36
Variance	7,87	2,53	4,51	3,59	3,20	5,13	10,01	12,32	4,92	11,48
Deviation	2.81	1.59	2.12	1.90	1.79	2.27	3.16	3.51	2.22	3.39
Mode	20.38	26.00	24.88	15.75	21.50	14.75	12.83	17.63	13.63	12.63
Minimum	15.25	21.25	22.33	11.25	19.88	13.25	11.50	12.88	13.63	12.63
Maximum	31.88	27.88	33.50	20.00	29.25	27.50	30.13	37.13	26.38	32.17

9.2 The Results of Synthetic Data Set

The parameters for controlling the run and the parameter representing the stop condition were set according to the values depicted in Tab. 9.8. The precision of the fitness function was not set, because this parameter is not used in the evolutionary optimization algorithm adapted for the warehouse optimization problem.

Both measurements were applied again to 20 individuals in 20 generations. Both measurements were started 5 times for all combinations of operators and arithmetically averaged (*Avg Duration*), and for the information, the collisions were counted and also arithmetically averaged (*Avg Collisions*). The algorithm was set as follows. The elitism remained the same (5 %), so only the best individual of the population was copied to the new population without change. The first example contains 20 employees in a shift and 50 jobs in the buffer. The rates of genetic operators were set to 20 % or 60 %. The second example contains 20 employees and 100 jobs in the buffer. The rates of genetic operators were set to 20 % or 80 % to prove that quite nothing will change when the rate of mutation will by slightly higher (60 % in the first case or 80 % in the second case).

Tab. 9.8: The settings for controlling the run of the GP algorithm.

Population Size	20 individuals
Evolution Size	20 generations
Fitness Precision	not set
Elitism Rate	5 %
PA Mutation Rate	Tab. 9.9 = 20/60 % / Tab. 9.10 = 20/80 %
JO Mutation Rate	Tab. 9.9 = 20/60 % / Tab. 9.10 = 20/80 %
SJ Mutation Rate	Tab. 9.9 = 20/60 % / Tab. 9.10 = 20/80 %
SW Mutation Rate	Tab. 9.9 = 20/60 % / Tab. 9.10 = 20/80 %
SP Mutation Rate	Tab. 9.9 = 20/60 % / Tab. 9.10 = 20/80 %

After several weeks of testing, the results of the optimization algorithm were obtained. After the final testing and optimization of algorithm, the last measurement of averaged values took 2 days 18 hours 36 minutes (238907 s). The results of the example of 20 employees and 50 jobs are given in Tab. 9.9. The results of the example of 20 employees and 100 jobs are given in Tab. 9.10. Three best results are highlighted in the orange colored box in each table. All results were measured on the Intel C2D E8400 architecture, 8GB RAM. The algorithm, more particularly the initialization method of evolutionary process generating all individuals, and the application of mutation operators during the evolution, was done in 8 threads.

Tab. 9.9: The results of measurement of the synthetic scenario generated with 20 employees and 50 jobs. All combinations of genetic operators were tested with 20 % and 60 % of the mutation rate.

Emps	Jobs	PA	JO	SJ	SW	SP	Avg Duration	Avg Collisions
20	50	0,20	0,20	0,20	0,20	0,20	37,50	318
20	50	0,60	0,20	0,20	0,20	0,20	38,03	329
20	50	0,20	0,60	0,20	0,20	0,20	35,79	344
20	50	0,60	0,60	0,20	0,20	0,20	36,06	334
20	50	0,20	0,20	0,60	0,20	0,20	35,75	326
20	50	0,60	0,20	0,60	0,20	0,20	36,54	327
20	50	0,20	0,60	0,60	0,20	0,20	37,49	339
20	50	0,60	0,60	0,60	0,20	0,20	36,43	375
20	50	0,20	0,20	0,20	0,60	0,20	36,91	369
20	50	0,60	0,20	0,20	0,60	0,20	37,01	341
20	50	0,20	0,60	0,20	0,60	0,20	38,05	327
20	50	0,60	0,60	0,20	0,60	0,20	37,50	315
20	50	0,20	0,20	0,60	0,60	0,20	38,03	334
20	50	0,60	0,20	0,60	0,60	0,20	36,32	342
20	50	0,20	0,60	0,60	0,60	0,20	36,09	349
20	50	0,60	0,60	0,60	0,60	0,20	36,83	328
20	50	0,20	0,20	0,20	0,20	0,60	35,00	324
20	50	0,60	0,20	0,20	0,20	0,60	38,88	305
20	50	0,20	0,60	0,20	0,20	0,60	38,68	323
20	50	0,60	0,60	0,20	0,20	0,60	35,70	329
20	50	0,20	0,20	0,60	0,20	0,60	36,05	316
20	50	0,60	0,20	0,60	0,20	0,60	36,28	359
20	50	0,20	0,60	0,60	0,20	0,60	37,24	311
20	50	0,60	0,60	0,60	0,20	0,60	38,06	318
20	50	0,20	0,20	0,20	0,60	0,60	38,91	320
20	50	0,60	0,20	0,20	0,60	0,60	37,93	355
20	50	0,20	0,60	0,20	0,60	0,60	35,88	337
20	50	0,60	0,60	0,20	0,60	0,60	38,06	324
20	50	0,20	0,20	0,60	0,60	0,60	38,72	311
20	50	0,60	0,20	0,60	0,60	0,60	35,22	352
20	50	0,20	0,60	0,60	0,60	0,60	37,37	349
20	50	0,60	0,60	0,60	0,60	0,60	36,81	349
						Average	37.03	333.77
						Variance	1.17	286.25
						Deviation	1.08	16.92
						Mode	37.50	333.80
						Minimum	35.00	305.40
						Maximum	38.91	375.40

Tab. 9.10: The results of measurement of the synthetic scenario generated with 20 employees and 100 jobs. All combinations of genetic operators were tested with 20 % and 60 % of the mutation rate.

Emps	Jobs	PA	JO	SJ	SW	SP	Avg Duration	Avg Collisions
20	100	0,20	0,20	0,20	0,20	0,20	73,67	721
20	100	0,80	0,20	0,20	0,20	0,20	73,38	705
20	100	0,20	0,80	0,20	0,20	0,20	71,67	731
20	100	0,80	0,80	0,20	0,20	0,20	67,50	783
20	100	0,20	0,20	0,80	0,20	0,20	72,50	797
20	100	0,80	0,20	0,80	0,20	0,20	64,67	711
20	100	0,20	0,80	0,80	0,20	0,20	79,50	687
20	100	0,80	0,80	0,80	0,20	0,20	61,50	637
20	100	0,20	0,20	0,20	0,80	0,20	68,00	603
20	100	0,80	0,20	0,20	0,80	0,20	62,50	643
20	100	0,20	0,80	0,20	0,80	0,20	73,50	637
20	100	0,80	0,80	0,20	0,80	0,20	74,33	663
20	100	0,20	0,20	0,80	0,80	0,20	68,00	547
20	100	0,80	0,20	0,80	0,80	0,20	73,75	739
20	100	0,20	0,80	0,80	0,80	0,20	67,88	690
20	100	0,80	0,80	0,80	0,80	0,20	66,38	680
20	100	0,20	0,20	0,20	0,20	0,80	72,00	740
20	100	0,80	0,20	0,20	0,20	0,80	69,50	637
20	100	0,20	0,80	0,20	0,20	0,80	74,63	612
20	100	0,80	0,80	0,20	0,20	0,80	77,50	800
20	100	0,20	0,20	0,80	0,20	0,80	78,75	624
20	100	0,80	0,20	0,80	0,20	0,80	68,67	774
20	100	0,20	0,80	0,80	0,20	0,80	68,63	677
20	100	0,80	0,80	0,80	0,20	0,80	80,63	743
20	100	0,20	0,20	0,20	0,80	0,80	72,17	680
20	100	0,80	0,20	0,20	0,80	0,80	77,50	736
20	100	0,20	0,80	0,20	0,80	0,80	71,63	681
20	100	0,80	0,80	0,20	0,80	0,80	72,63	747
20	100	0,20	0,20	0,80	0,80	0,80	70,00	683
20	100	0,80	0,20	0,80	0,80	0,80	71,33	728
20	100	0,20	0,80	0,80	0,80	0,80	69,67	750
20	100	0,80	0,80	0,80	0,80	0,80	65,25	668
						Average	71.22	695.44
						Variance	21.25	3454.68
						Deviation	4.61	58.78
						Mode	77.50	637.00
						Minimum	61.50	547.00
						Maximum	80.63	800.00

10 CONCLUSION

This chapter gives a conclusion to the submitted doctoral thesis. Section 10.1 describes the discussion of the results reached by the proposed optimization algorithm, based on GP driven by the CFG. The results reached on the real and synthetic data are presented. Section 10.2 describes the summary of the thesis, the contribution to the area of scheduling and the final evaluation of results. And the last section 10.3 gives the ideas of future work which could possibly extend the proposed solution. This thesis also represents a part of the project reg. no. FR-TI1/444 "Research and Development of the System for Manufacturing Optimization" which was led by principal investigator Prof. Ing. Zdeněk Smékal, CSc.

10.1 Discussion of Results

The optimization algorithm showed very good results of the measurements of the real data set. The overall performance of the optimization algorithm reached 93.33 %, which means that the results of the operational manager were outperformed in 56 cases of the total number of 60 scenarios. The first three sets of scenarios reached only 86.67 %, but only because the scenarios in these sets are very simple case studies. It is worth of noting that in such simple scenarios the improvement of solution was not expected at all. The results worst or on the same level of performance as the operational manager were expected in these cases.

In fact, it is quite surprising that the algorithm was capable to find the solution which is better than the original one reached by the operational manager, because the scenarios represent such simple cases where the space for further optimization is almost unrecognizable. The second three sets of scenarios represent complex cases of scheduling problems. In these cases, the recognizable optimization was expected, because these cases are quite difficult to solve with conventional methods used in warehouses in an optimal way. In this cases, all results reached by the operational manager were outperformed by the proposed optimization algorithm.

The results reached on the real data sets depicted in section 9.1 show that any single operator can be used alone on both the very simple scenarios and the complex scenarios, but the performance of only one single operator, no matter which, is better in simple cases. More complicated scenarios should be optimized by the combination of at least two or three genetic operators, but the more operators used the better result should be obtained and the diversity of population will be supported.

The results reached on the synthetic data sets presented in section 9.2 show that the time of processing of job buffer increases linearly. It is also shown that the setting of mutation rate higher than 20 % does not have a significant impact on the

results. The only influenced factor is a computational time which rapidly decreases with a higher rate of mutations.

10.2 Summary of Thesis

The goals stated in chapter 3 were reached. The human factor was involved in the optimization parameters, which enhanced the standard mathematical model (see chapter 4). So, the performance of particular employees was taken into account which could positively influence the processing time of the scheduling of jobs. The multi-criteria fitness function was involved in the optimization process.

The fitness function can respect multiple criteria now. The function works with respect to the optimization of time processing, the balanced workload of employees, and the number of collisions of trucks. The variability of time planning and simulation was implemented. The simulation is capable to schedule the work for a few next minutes as well as for the whole working shift, or also longer time intervals, which of course is much more time consuming. The possibility of co-operative jobs was analyzed, designed, and implemented to the optimization algorithm as one of the genetic operators. The co-operative jobs helps to solve situations when one employee is able to do only a part of job and another employee has to finish the job.

New Evolutionary Framework, which is able to run on genetic programming algorithms driven by context-free grammar was developed as flexible and robust as possible. The Evolutionary Framework was tested also on different tasks where the validity of this solution was proved. The last, but not least, the goal was to design and to implement benchmark tests, which would prove the functionality of the proposed solution and prove that the hypothesis is true. The design and implementation of benchmark test was done and the results of measurement proved that the proposed optimization algorithm is competitive to the operational manager, the results were outperformed in 93.33 % of cases, which leaves a little space for further development of the evolutionary optimization algorithm.

Of course, the deployment of the algorithm is not suitable in all situations. If the scenarios are quite simple (small warehouses with units of employees), the performance of the algorithm is not so good as in the complex scenarios with difficult situations, dozens of employees and trucks. Even if the proposed algorithm is not suitable in all warehousing environments, the main goal of the doctoral thesis **"substantial increase of productivity and reduction of operating costs"** was reached and the hypothesis was proved to be true.

The main contributions of the proposed doctoral thesis are:

a) A comprehensive literature review of the warehouse optimization connected to the scheduling problems and the vehicle routing problem was written in chapter 2. The basics found in the literature helped to define the hypothesis and goals (chapter 3), and to extend the mathematical model (chapter 4 for the warehouse work-flow optimization with the help of employees' performance which positively influences the processing time of the scheduling process.

b) The new, extensible, flexible, and multi-platform Evolutionary Framework with the computational core based on GP driven by the CFG was developed, implemented, and validated (chapter 6). The framework is based on the existing GGGP approach presented also in this chapter.

c) The new algorithm for the warehouse work-flow optimization problem (chapter 7) based on the proposed framework (chapter 6) was developed and supported by several new genetic operators, which give the possibility of co-operative job processing. The fitness function can respect multiple criteria, such as the time of processing of the whole job buffer (the makespan), balanced workload of employees, and the number of collisions of trucks (chapter 5).

d) The problem of the warehouse work-flow optimization was described along with the motivation to solve the problem. The set of benchmark tests was created as well as the evaluation process. These together give the reference baseline of the results for the optimization (chapter 8).

The most interesting part of the thesis is a design of the optimization algorithm, which uses a new technology in the form of the high-level object oriented genetic programming, designed specially for this thesis. Such a solution can find the applications not only in the field of optimization of logistics environments, but also in e.g. manufacturing companies where the guarantee of smooth flow of material and flow of individual components must be ensured. The effort is made to prevent shutdowns and/or production interruptions and bring significant cost savings not only to manufacturing company, but also to other business partners in logistics chain.

10.3 Future of the Work

The future research will be focused on the dispatching rules. The rules will be investigated and implemented to the optimization algorithm. The aim of this work is to connect the logical priorities of the dispatching rules with the randomized process of GP. The individuals could be generated based on randomized process inspired by GBIM, the evolution process could work on the basis of genetic operators

designed for the purpose of this thesis, but the evolution would be supported in every nth step by the dispatching rules which could improve the results.

The further research will also include the genetic operators themselves, the settings of the algorithm with newly applied operators and the optimization of the performance in the way of memory management and computational performance. Currently, the algorithm works as a simultaneous algorithm as well as a parallel algorithm in t threads. The future work could deploy the distributed model of the algorithm. Of course, when the algorithm is being developed, the test sets and benchmarking tasks have also to be developed. So, the real data set will be regularly extended on the basis of historical operational data from the logistic environment.

BIBLIOGRAPHY

[1] Leo Liberti and Nelson Maculan, *Global Optimization: From Theory to Implementation*, Springer, February 1 2006, ISBN-13: 978-0387282602.

[2] Thomas Weise, *Global Optimization Algorithms - Theory and Application*, Thomas Wiese, 3rd edition, April 24 2011.

[3] James A. Tompkins, John A. White, Yavuz A. Bozer, and J. M. A. Tanchoco, *Facilities Planning*, Wiley, January 19 2010, ISBN-13: 978-0470444047.

[4] Bernhard Korte and Jens Vygen, *Combinatorial Optimization: Theory and Algorithms*, Spring, 5th edition, January 10 2012, ISBN-13: 978-3642244872.

[5] Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemarechal, and Claudia A. Sagastizábal, *Numerical Optimization: Theoretical and Practical Aspects*, Springer, 2nd edition, November 13 2006, ISBN-13: 978-354035445.

[6] Michael T. Hompel and Thorsten Schmidt, *Warehouse Managmenet - Automation and Organisation of Warehouse and Order Picking Systems*, Springer-Verlag Berlin Heidelberg, 2007, ISBN-13: 971-3-540-35218-1.

[7] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis, "Research on warehouse design and performance evaluation: A comprehensive review," *European Journal of Operational Research*, vol. 203, no. 3, pp. 539–549, 2010.

[8] Franco Caron, Gino Marchet, and Alessandro Perego, "Optimal layout in low-level picker-to-part systems," *International Journal of Production Research*, vol. 38, no. 1, pp. 101–117, 2000.

[9] Kees Jan Roodbergen and Rene de Koster, "Routing methods for warehouses with multiple cross aisles," *International Journal of Production Research*, vol. 39, no. 9, pp. 1865–1883, 2001.

[10] Kees Jan Roodbergen and Rene de Koster, "Routing order pickers in a warehouse with a middle aisle," *European Journal of Operational Research*, vol. 133, no. 1, pp. 32–43, 2001.

[11] John J. Bartholdi and Steven T. Hackman, *Warehouse & Distribution Science*, Georgia Institute of Technology, School of Industrial and Systems Engineering, The Supply Chain and Logistics Institute, August 22 2011, Release: 0.95.

[12] Rene de Koster, Tho Le-Duc, and Kees Jan Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007.

[13] Sunderesh S. Heragu, *Facilities Design*, CRC Press, 3rd edition, June 19 2008, ISBN-13: 978-1420066265.

[14] Kevin R. Gue and Russell D. Meller, "Aisle configurations for unit-load warehouses," *IIE Transactions*, vol. 41, no. 3, pp. 171–182, 2009.

[15] Letitia M. Pohl, Russell D. Meller, and Kevin R. Gue, "An analysis of dual-command operations in common warehouse designs," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 3, pp. 367–379, 2009.

[16] Letitia M. Pohl, Russell D. Meller, and Kevin R. Gue, "Optimizing fishbone aisles for dual-command operations in a warehouse," *Naval Research Logistics*, vol. 56, no. 5, pp. 389–403, 2009.

[17] Kevin R. Gue, Goran Ivanovic̄, and Russell D. Meller, "A unit-load warehouse with multiple pickup and deposit points and non-traditional aisles," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 4, pp. 795–806, 2012.

[18] K. Gue, R. Meller, and J. Skufca, "The effects of pick density on order picking areas with narrow aisles," *IIE Transactions*, vol. 38, no. 10, pp. 859–868, 2006.

[19] M. Napolitano, "Real dc stories: Low cost deep impact," *Logistics Management*, vol. 48, no. 1, pp. 46–49, 2009.

[20] John J. Bartholdi and Loren K. Platzman, "Retrieval strategies for a carousel conveyor," *IIE Transactions*, vol. 18, no. 2, pp. 166–173, 1986.

[21] Jay B. Ghosh and Charles E. Wells, "Optimal retrieval strategies for carousel conveyors," *Math. and Computer Modelling*, vol. 16, no. 10, pp. 59–70, 1992.

[22] Nelly Litvak, "Optimal picking of large orders in carousel systems," *Operations Research Letters*, vol. 34, no. 2, pp. 219–227, 2006.

[23] Jeroen P. Van Den Berg, "Multiple order-pick sequencing in a carousel system: A solvable case of the rural postman problem," *The Journal of the Operational Research Society*, vol. 47, no. 12, pp. 1504–1515, December 1996.

[24] Raymond G. Vickson and Allen Fujimoto, "Optimal storage locations in a carousel storage and retrieval system," *Location Science*, vol. 4, no. 4, pp. 237–245, 1996.

[25] Yavuz A. Bozer and John A. White, "Travel-time models for automated storage/retrieval systems," *IIE Trans.*, vol. 16, no. 4, pp. 329–338, 1984.

[26] Yavuz A. Bozer and John A. White, "Design and performance models for end-of-aisle order picking systems," *Management Science*, vol. 36, no. 7, pp. 852–866, July 1990.

[27] Ya-Hong Hu, Shell Ying Huang, Chuanyu Chen, Wen-Jing Hsu, Ah Cheong Toh, Chee Kit Loh, and Tiancheng Song, "Travel time analysis of a new automated storage and retrieval system," *Computers & Operations Research*, vol. 32, no. 6, pp. 1515–1544, 2005.

[28] Tone Lerher, Iztok Potrč, Matjaž Šraml, and Tomaž Tollazzi, "Travel time models for automated warehouses with aisle transferring storage and retrieval machine," *European Journal of Operational Research*, vol. 205, no. 3, pp. 571–583, 2010.

[29] J. L. Haskett, "Cube-per-order index - a key to warehouse stock location," *Transportation and Distribution Management*, vol. 3, no. 1, pp. 27–31, 1963.

[30] Charles G. Petersen, "An evaluation of order picking routeing policies," *International Journal of Operations & Production Management*, vol. 17, no. 11, pp. 1098–1111, 1997.

[31] H. Brynzer and M. I. Johansson, "Storage location assignment: Using the product structure to reduce order picking times," *International Journal of Production Economics*, vol. 46, no. 1, pp. 595–603, December 1996.

[32] Ronald J. Mantel, Peter C. Schuur, and Sunderesh S. Heragu, "Order oriented slotting: A new assignment strategy for warehouses," *European Journal of Industrial Engineering*, vol. 1, no. 3, pp. 301–316, January 1 2007.

[33] H. D. Ratliff and A. S. Rosenthal, "Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem," *Operations Research*, vol. 31, no. 3, pp. 507–521, May–June 1983.

[34] G. Clarke and J. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Oper. Research*, vol. 12, no. 4, pp. 568–581, 1964.

[35] Rene de Koster, E. S. Van der Poort, and M. Woltersa, "Efficient orderbatching methods in warehouses," *International Journal of Production Research*, vol. 37, no. 7, pp. 1479–1504, 1999.

[36] Inneke Van Nieuwenhuyse and René B.M. de Koster, "Evaluating order throughput time in 2-block warehouses with time window batching," *International Journal of Production Economics*, vol. 121, no. 2, pp. 654–664, 2009.

[37] Herwen de Ruijter, "Improved storage in a book warehouse," M.S. thesis, University of Twente, Enschede – The Netherlands, October 2007.

[38] Tomasz Ambroziak and Konrad Lewczuk, "A method for scheduling the goods receiving process in warehouse facilities," *Total Logistic Management*, vol. 5, no. 1, pp. 7–14, 2008.

[39] Xiaowei Zhu, Samar K. Mukhopadhyay, and Hisashi Kurata, "A review of rfid technology and its managerial applications in different industries," *Journal of Engineering and Technology Management*, vol. 29, no. 1, pp. 152–167, 2012.

[40] Ming K. Lim, Witold Bahr, and Stephen C. H. Leung, "Rfid in the warehouse: A literature analysis (1995–2010) of its applications, benefits, challenges and future trends," *International Journal of Production Economics*, vol. 145, no. 1, pp. 409–430, 2013.

[41] Christophe Theys, Olli Braysy, Wout Dullaert, and Birger Raa, "Using a tsp heuristic for routing order pickers in warehouses," *European Journal of Operational Research*, vol. 200, no. 3, pp. 755–763, 2010.

[42] Klaus Moeller, "Increasing warehouse order picking performance by sequence optimization," *Social and Behavioral Sciences*, vol. 20, pp. 177–185, 2011.

[43] H. Hwang, Y.H. Oh, and Y.K. Lee, "An evaluation of routing policies for order-picking operations in low-level picker-to-part system," *International Journal of Production Research*, vol. 42, no. 18, pp. 3873–3889, 2004.

[44] M. B. M. de Koster and M. Yu, "Minimizing makespan and throughput times at aalsmeer flower auction," *Journal of Operational Research Society*, vol. 59, no. 9, pp. 1182–1190, September 2008.

[45] S. Hong, A. L. Johnson, and B. A. Peters, "Analysis of picker blocking in narrow-aisle batch picking," in *Proceedings of 2010 International Material Handling Research Colloquium (IMHRC)*, K.P. Ellis, K. Gue, d. R. Koster, R. Meller, B. Montreuil, and M. Oglep, Eds. The Material Handling Institue, Charlotte, NC, USA, 2010.

[46] P. J. Parikh and R. D. Meller, "A note on worker blocking in narrow-aisle order picking systems when pick time is non-deterministic," *IIE Transactions*, vol. 42, no. 6, pp. 392–404, 2010.

[47] Jason Chao-Hsien Pan and Po-Hsun Shih, "Evaluation of the throughput of a multiple-picker order picking system with congestion consideration," *Computers & Industrial Engineering*, vol. 55, no. 2, pp. 379–389, 2008.

[48] Jason Chao-Hsien Pan and Ming-Hung Wu, "Throughput analysis for order picking system with multiple pickers and aisle congestion considerations," *Computers & Operations Research*, vol. 39, no. 7, pp. 1661–1672, 2012.

[49] Jason Chao-Hsien Pan, Po-Hsun Shih, and Ming-Hung Wu, "Storage assignment problem with travel distance and blocking considerations for a picker-to-part order picking system," *Computers & Industrial Engineering*, vol. 62, no. 2, pp. 527–535, 2012.

[50] Soondo Hong, Andrew L. Johnson, and Brett A. Peters, "Batch picking in narrow-aisle order picking systems with consideration for picker blocking," *European Journal of Operational Research*, vol. 221, no. 3, pp. 557–570, 2012.

[51] Fangyu Chen, Hongwei Wang, Chao Qi, and Yong Xie, "An ant colony optimization routing algorithm for two order pickers with congestion consideration," *Computers & Industrial Engineering*, vol. 66, no. 1, pp. 77–85, 2013.

[52] Ling-Feng Hsieh and Yi-Chen Huang, "New batch construction heuristics to optimise the performance of order picking systems," *International Journal of Production Economics*, vol. 131, no. 2, pp. 618–630, 2011.

[53] Jose I. U. Rubrico, Toshimitu Higashi, Hirofumi Tamura, and Jun Ota, "Online rescheduling of multiple picking agents for warehouse management," *Robot. Comput.-Integr. Manuf.*, vol. 27, no. 1, pp. 62–71, February 2011.

[54] Yossi Bukchin, Eugene Khmelnitsky, and Pini Yakuel, "Optimizing a dynamic order-picking process," *European Journal of Operational Research*, vol. 219, no. 2, pp. 335–346, 2012.

[55] Sebastian Henn, "Algorithms for on-line order batching in an order picking warehouse," *Computers & Oper. Res.*, vol. 39, no. 11, pp. 2549–2563, 2012.

[56] Sebastian Henn and Verena Schmid, "Metaheuristics for order batching and sequencing in manual order picking systems," *Computers & Industrial Engineering*, vol. 66, no. 2, pp. 338–351, 2013, ISSN: 0360–8352.

[57] Marek Matusiak, René Koster, Leo Kroon, and Jari Saarinen, "A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse," *European Journal of Oper. Res.*, vol. ?, no. ?, pp. ?, 2013.

[58] Ann E. Gray, Uday S. Karmarkar, and Abraham Seidmann, "Design and operation of an order-consolidation warehouse: Models and application," *European Journal of Operational Research*, vol. 58, pp. 14–36, 1992.

[59] G. Mosheiov, "Vehicle routing with pick-up and delivery: Tour-partitioning heuristics," *Computers and Industrial Eng.*, vol. 34, no. 3, pp. 669–684, 1998.

[60] G. Barbarosoglu and D. Ozgur, "A tabu search algorithm for the vehicle routing problem," *Comp. and Op. Research*, vol. 26, no. 3, pp. 255–270, 1999.

[61] B. Vahdani and M. Zandieh, "Scheduling trucks in cross-docking systems: Robust meta-heuristics," *Comp. & Ind. Eng.*, vol. 58, no. 1, pp. 12–24, 2010.

[62] Hong Yan and Shao-long Tang, "Pre-distribution and post-distribution cross-docking operations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 45, no. 6, pp. 843–859, 2009.

[63] Michael L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, 4th edition, January 6 2012, ISBN-13: 978-1461419860.

[64] J. R. Jackson, "Scheduling a production line to minimize maximum tardiness," Tech. Rep., University of California, Los Angeles, 1955.

[65] Wayne E. Smith, "Various optimizers for single-stage production," *Naval Research Logistics Quarterly*, vol. 3, no. 1, pp. 59–66, 1956.

[66] Edward Ignall and Linus Schrage, "Application of the branch and bound technique to some flow-shop scheduling problems," *Operations Research*, vol. 13, no. 3, pp. 400–412, May–June 1965.

[67] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds., pp. 85–103. Plenum Press, 1972.

[68] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems*, P. L. Hammer, E. L. Johnson, and B. H. Korte, Eds., vol. 5, pp. 287–326. Elsevier, 1979.

[69] J. C. Gittins, "Bandit processes and dynamic allocation indices," *Journal of the Royal Statistical Society*, vol. 41, no. 2, pp. 148–177, 1979.

[70] Richard Weber, "On the gittins index for multiarmed bandits," *The Annals of Applied Probability*, vol. 2, no. 4, pp. 1024–1033, November 1992.

[71] Teofilo Gonzalez and Sartaj Sahni, "Open shop scheduling to minimize finish time," *Journal of the ACM*, vol. 23, no. 4, pp. 665–679, October 1976.

[72] Teofilo Gonzalez and Sartaj Sahni, "Flowshop and jobshop schedules: Complexity and approximation," *Op. Research*, vol. 26, no. 1, pp. 36–52, 1978.

[73] C. N. Potts and L. N. van Wassenhove, "Algorithms for scheduling a single machine to minimize the weighted number of late jobs," *Management Science*, vol. 34, pp. 843–858, 1988.

[74] Joseph Adams, Egon Balas, and Daniel Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, March 1988.

[75] Fred Glover, "Tabu search - part i," *Journal on Computing*, vol. 1, no. 3, pp. 190–206, Summer 1989.

[76] Chung-Yee Lee, Lei Lei, and Michael Pinedo, "Current trends in deterministic scheduling," *Annals of Operations Research*, vol. 70, pp. 1–41, 1997.

[77] R. Linn and W. Zhang, "Hybrid flow shop scheduling: A survey," *Computers & Industrial Engineering*, vol. 37, no. 1–2, pp. 57–61, October 1999.

[78] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operations Research*, vol. 113, no. 2, pp. 390–434, March 1 1999.

[79] H. L. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 375–382.

[80] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, no. 1, pp. 113–125, January–February 1992.

[81] M. Ben-Daya and M. Al-Fawzan, "A tabu search approach for the flow shop scheduling problem," *European Journal of Operational Research*, vol. 109, no. 1, pp. 88–95, August 16 1998.

[82] L. Wang and D. Z. Zheng, "An effective hybrid optimization strategy for job-shop scheduling problems," *Computers & Operations Research*, vol. 28, no. 6, pp. 585–596, May 2001.

[83] L. Wang and D. Z. Zheng, "An effective hybrid heuristic for flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 21, no. 1, pp. 38–44, 2003.

[84] C. F. Liaw, "A hybrid genetic algorithm for the open shop scheduling problem," *European Journal of Op. Res.*, vol. 124, no. 1, pp. 28–42, July 1 2000.

[85] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 32, no. 1, pp. 1–13, February 2002.

[86] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 245–276, September 30 2002.

[87] C. Blum, "Beam-aco - hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Computers & Operations Research*, vol. 32, no. 6, pp. 1565–1591, June 2005.

[88] M. Zandieh, S. M. T. Fatemi Ghomi, and S. M. Moattar Husseini, "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 111–127, SEP 1 2006.

[89] Seyed Habib A. Rahmati and M. Zandieh, "A new biogeography-based optimization (bbo) algorithm for the flexible job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 58, no. 9–12, pp. 1115–1129, February 2012.

[90] Chung-Yee Lee, *Handbook of Scheduling*, chapter Machine Scheduling with Availability Constraints, p. Chapter 22, Chapman and Hall, 2004.

[91] Selmer M. Johnson, "Optimal two- and three-stage production schedules and setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, March 1954.

[92] Michael Andresen and Tanka Nath Dhamala, "New algorithms and complexity status of the reducibility problem of sequences in open shop scheduling minimizing the makespan," *Annals of Operations Research*, vol. 196, no. 1, pp. 1–26, July 2012.

[93] Y. Cho and S. Sahni, "Preemptive scheduling of independent jobs with release and due times on open, flow and job shops," *Operations Research*, vol. 29, no. 3, pp. 511–522, 1981.

[94] Danyu Bai and Lixin Tang, "Open shop scheduling problem to minimize makespan with release dates," *Applied Mathematical Modelling*, vol. 37, no. 4, pp. 2008–2015, February 15 2013.

[95] B. Naderi, S. M. T. Fatemi Ghomi, M. Aminnayeri, and M. Zandieh, "Scheduling open shops with parallel machines to minimize total completion time," *Journal of Computational and Applied Mathematics*, vol. 235, no. 5, pp. 1275–1287, January 1 2011.

[96] C. Y. Liu and R. L. Bulfin, "On the complexity of preemptive open-shop scheduling problems," *Op. Res. Letters*, vol. 4, no. 2, pp. 71–74, 1985.

[97] Ming Liu, Chengbin Chu, Yinfeng Xu, and Feifeng Zheng, "An optimal online algorithm for two-machine open shop preemptive scheduling with bounded processing times," *Optimization Letters*, vol. 4, no. 2, pp. 227–237, May 2010.

[98] H. Ishii and T. Nishida, "2 machine open shop scheduling problem with controllable machine speeds," *Journal of the Operations Research Society of Japan*, vol. 29, no. 2, pp. 123–132, June 1986.

[99] V. A. Strusevich, "A heuristic for the two-machine open-shop scheduling problem with transportation times," *Discrete Applied Mathematics*, vol. 93, no. 2–3, pp. 287–304, July 20 1999.

[100] J. Breit, G. Schmidt, and V. A. Strusevich, "Two-machine open shop scheduling with an availability constraint," *Operations Research Letters*, vol. 29, no. 2, pp. 65–77, September 2001.

[101] C. A. Glass, C. N. Potts, and V. A. Strusevich, "Scheduling batches with sequential job processing for two-machine flow and open shops," *INFORMS Journal on Computing*, vol. 13, no. 2, pp. 120–137, Spring 2001.

[102] Rui Zhang and Cheng Wu, "A simulated annealing algorithm based on bottleneck jobs for the open shop scheduling problem," in *2008 7th World Congess on Intelligent Control and Automation, Vols 1–23*, 2008, pp. 4453–4457.

[103] S. J. Louis and Z. J. Xu, "Genetic algorithms for open shop scheduling and rescheduling," in *Computers and Their Applications – Proceedings of the ICSA 11th Int. Conf.*, M. E. Cohen and D. L. Hudson, Eds., 1996, pp. 99–102.

[104] C. F. Liaw, "A tabu search slgorithm for the open shop scheduling problem," *Computers & Operations Research*, vol. 26, no. 2, pp. 109–126, February 1999.

[105] C. F. Liaw, "An efficient tabu search approach for the two-machine preemptive open shop scheduling problem," *Computers & Operations Research*, vol. 30, no. 14, pp. 2081–2095, December 2003.

[106] C. F. Liaw, "Applying simulated annealing to the open shop scheduling problem," *IIE Transactions*, vol. 31, no. 5, pp. 457–465, May 1999.

[107] C Prins, "Competitive genetic algorithms for the open-shop scheduling problem," *Math. Methods of Op. Research*, vol. 52, no. 3, pp. 389–411, 2000.

[108] U. Dorndorf, E. Pesch, and T. Phan-Huy, "Solving the open shop scheduling problem," *Journal of Scheduling*, vol. 4, no. 3, pp. 157–174, May–June 2001.

[109] T. Lorigeon, J. C. Billaut, and J. L. Bouquard, "A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint," *Journal of the Operational Research Society*, vol. 53, no. 11, pp. 1239–1246, November 2002.

[110] Rui Zhang and Cheng Wu, "An immune mechanism for the open shop scheduling problem with application to genetic algorithm," in *2008 Chienese Control and Decision Conference, Vols 1–11*, 2008, pp. 159–164.

[111] D. Y. Sha and Cheng-Yu Hsu, "A new particle swarm optimization for the open shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3243–3261, October 2008.

[112] D. Y. Sha, Hsing-Hung Lin, and C. Y. Hsu, "A modified particle swarm optimization for multi-objective open shop scheduling," in *International Multiconference of Engineers and Computer Scientists (IMECS 2010) Vols I-III*, S. I. Ao, O. Castillo, C. Douglas, D. D. Feng, and J. A. Lee, Eds., 2010, Lecture Notes in Engineering and Computer Science, pp. 1844–1848.

[113] Y. Zhan, Y. G. Zhong, and H. T. Zhu, "Preemptive open-shop scheduling: Network flow based algorithm," in *Digital Design and Manufacturing Technology II*, C. Lu, Ed., 2011, vol. 215, pp. 111–114.

[114] Tadeusz Witkowski, Pawel Antczak, and Arkadiusz Antczak, "Hybrid method for solving flexible open shop scheduling problem with simulated annealing algorithm and multi-agent approach," in *Manufacturing Science and Technology*, W. Fan, Ed., 2012, vol. 383–390, pp. 4612–4619.

[115] Fardin Ahmadizar and Mehdi Hosseinabadi Farahani, "A novel hybrid genetic algorithm for the open shop scheduling problem," *Int. Journal of Advanced Manufacturing Technology*, vol. 62, no. 5–8, pp. 775–787, September 2012.

[116] M. Emin Baysal, Taha Durmaz, Ahmet Sarucan, and Orhan Engin, "To solve the open shop scheduling problems with the parallel kangaroo algorithm," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 27, no. 4, pp. 855–864, December 2012.

[117] Jianming Dong, An Zhang, Yong Chen, and Qifan Yang, "Approximation algorithms for two-machine open shop scheduling with batch and delivery coordination," *Theoretical Comp. Science*, vol. 491, pp. 94–102, June 17 2013.

[118] Yong Chen, An Zhang, Guangting Chen, and Jianming Dong, "Approximation algorithms for parallel open shop scheduling," *Information Processing Letters*, vol. 113, no. 7, pp. 220–224, April 15 2013.

[119] Yong Zhan, Yuguang Zhong, and Haitao Zhu, "Research on open shop scheduling based on genetic algorithm," in *Engineering Solutions for Manufacturing Processes*, Z. Y. Jiang, X. H. Liu, S. H. Jiao, and J. T. Han, Eds., 2013, vol. 655–657 of *Advanced Materials Research*, pp. 1670–1674.

[120] Chang Sup Sung and Sang Hum Yoon, "Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines," *International Journal of Production Economics*, vol. 54, no. 3, pp. 247–255, 1998.

[121] Thomas A. Roemer, "A note on the complexity of the concurrent open shop problem," *Journal of Scheduling*, vol. 9, no. 4, pp. 389–396, 2006.

[122] G. B. McMahon and P. G. Burton, "Flow-shop scheduling with branch-and-bound method," *Operations Research*, vol. 15, no. 3, pp. 473–481, 1967.

[123] B. D. Corwin and A. O. Esogbue, "2 machine flow shop scheduling problems with sequence dependent setup times – dynamic-programming approach," *Naval Research Logistics*, vol. 21, no. 3, pp. 515–524, 1974.

[124] E. Nowicki and S. Zdrzalka, "A 2-machine flow-shop scheduling problem with controllable job processing times," *European Journal of Operational Research*, vol. 34, no. 2, pp. 208–220, March 1988.

[125] K. R. Baker, "Scheduling groups of jobs in the 2-machine flow-shop," *Mathematical and Computer Modeling*, vol. 13, no. 3, pp. 29–36, 1990.

[126] R. G. Vickson and B. E. Alfredsson, "2-machine and 3-machine flow-shop scheduling problems with equal sized transfer batches," *International Journal of Production Research*, vol. 30, no. 7, pp. 1551–1574, July 1992.

[127] C. N. Potts, "Analysis of a linear-programming heuristic for scheduling unrelated parallel machines," *Discrete Applied Mathematics*, vol. 10, no. 2, pp. 155–164, 1985.

[128] C. Y. Liu and S. C. Chang, "Scheduling flexible flow shops with sequence-dependent setup effects," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, pp. 408–419, August 2000.

[129] P Kouvelis, RL Daniels, and G Vairaktarakis, "Robust scheduling of a two-machine flow shop with uncertain processing times," *IIE Transactions*, vol. 32, no. 5, pp. 421–432, 2000.

[130] R. Aggoune, "Minimizing the makespan for the flow shop scheduling problem with availability constraints," *European Journal of Operational Research*, vol. 153, no. 3, pp. 534–543, March 16 2004.

[131] A. Soukhal, A. Oulamara, and P. Martineau, "Complexity of flow shop scheduling problems with transportation constraints," *European Journal of Operational Research*, vol. 161, no. 1, pp. 32–41, February 16 2005.

[132] L. G. Mitten, "Sequencing n jobs on two machines with arbitrary time lags," *Management Science*, vol. 5, no. 3, pp. 293–298, 1959.

[133] Selmer M. Johnson, "Discussion sequencing n jobs on two machines with arbitrary time lags," *Management Science*, vol. 5, no. 3, pp. 299–303, 1959.

[134] J. K. Lenstra and A. H. G. Rinnooy Kan, "Some simple applications of the travelling salesman problem," *Operational Research Quarterly*, vol. 26, no. 4, pp. 717–733, November 1975.

[135] John M. Van Deman and Kenneth R. Baker, "Minimizing mean flowtime in the flow shop with no intermediate queues," *AIIE Transactions*, vol. 6, no. 1, pp. 28–34, 1974.

[136] Nicholas G. Hall and Chelliah Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations Research*, vol. 44, no. 3, pp. 510–525, May–June 1996.

[137] L. A. Hall, "Approximability of flow shop scheduling," *Mathematical Programming*, vol. 82, no. 1–2, pp. 175–190, June 1 1998.

[138] H. Ishibuchi, N. Yamamoto, S. Misaki, and H. Tanaka, "Local search algorithms for flow-shop scheduling with fuzzy due-dates," *International Journal of Production Economics*, vol. 33, no. 1–3, pp. 53–66, January 1994.

[139] C. A. Glass and C. N. Potts, "A comparison of local search methods for flow shop scheduling," *Annals of Operations Research*, vol. 63, pp. 489–509, 1996.

[140] L. F. Gelders and N. Sambandam, "4 simple heuristics for scheduling a flow-shop," *Int. Journal of Production Research*, vol. 16, no. 3, pp. 221–231, 1978.

[141] B. Chen, "Analysis of classes of heuristics for scheduling a 2-stage flow-shop with parallel machines at one-stage," *Journal of the Operational Research Society*, vol. 46, no. 2, pp. 234–244, February 1995.

[142] G. A Clevelland and S. F. Smith, "Using genetic algorithms to schedule flow-shop releases," in *Proceedings of the Third Interantional Conference on Genetic Algorithms*, J. D. Schaffer, Ed., San Mateo, 1989, pp. 160–169.

[143] J. P. Watson, L. Barbulescu, L. D. Whitley, and A. E. Howe, "Contrasting structured and random permutation flow-shop scheduling problems: Search-space topology and algorithm performance," *INFORMS Journal on Computing*, vol. 14, no. 2, pp. 98–123, Spring 2002.

[144] S. H. Yoon and J. A. Ventura, "An application of genetic algorithms to lot-streaming flow shop scheduling," *IIE Trans.*, vol. 34, no. 9, pp. 779–787, 2002.

[145] S. G. Ponnambalam, H. Jagannathan, M. Kataria, and A. Gadicherla, "A tsp-ga multi-objective algorithm for flow-shop scheduling," *Int. Journal of Adv. Manufacturing Technology*, vol. 23, no. 11–12, pp. 909–915, June 2004.

[146] S. Bertel and J. C. Billaut, "A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation," *European Journal of Operational Research*, vol. 159, no. 3, pp. 651–662, December 16 2004.

[147] C. Low, "Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines," *Computer & Operations Research*, vol. 32, no. 8, pp. 2013–2025, August 2005.

[148] J. B. Wang and Z. Q. Xia, "Flow-shop scheduling with a learning effect," *Journal of the Op. Research Society*, vol. 56, no. 11, pp. 1325–1330, 2005.

[149] G. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, June 1 2006.

[150] L. Wang, L. Zhang, and D. Z. Zheng, "An effective hybrid genetic algorithm for flow shop scheduling with limited buffers," *Computers & Operations Research*, vol. 33, no. 10, pp. 2960–2971, October 2006.

[151] Bo Liu, Ling Wang, and Yi-Hui Jin, "An effective hybrid particle swarm optimization for no-wait flow shop scheduling," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 9–10, pp. 1001–1011, January 2007.

[152] Bo Liu, Ling Wang, and Yi-Hui Jin, "An effective pso-based memetic algorithm for flow shop scheduling," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 1, pp. 18–27, February 2007.

[153] Bin-Bin Li and Ling Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 37, no. 3, pp. 576–591, June 2007.

[154] Reza Tavakkoli-Moghaddam, Alireza Rahimi-Vahed, and Ali Hossein Mirzaei, "A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness," *Inf. Sciences*, vol. 177, no. 22, pp. 5072–5090, November 15 2007.

[155] Bo Liu, Ling Wang, and Yi-Hui Jin, "An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers," *Computers & Operations Research*, vol. 35, no. 9, pp. 2791–2806, September 2008.

[156] Quan-Ke Pan, Ling Wang, and Bin Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Computers & Operations Research*, vol. 36, no. 8, pp. 2498–2511, AUG 2009.

[157] Ling Wang, Quan-Ke Pan, P. N. Suganthan, Wen-Hong Wang, and Ya-Min Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems," *Computers & Operations Research*, vol. 37, no. 3, pp. 509–520, March 2010.

[158] Hui-Mei Wang, Fuh-Der Chou, and Ful-Chiang Wu, "A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan," *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 5–8, pp. 761–776, March 2011.

[159] Quan-Ke Pan, M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Inf. Sciences*, vol. 181, no. 12, pp. 2455–2468, June 15 2011.

[160] Imma Ribas, Rainer Leisten, and Jose M. Framinan, "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective," *Computers & Operations Research*, vol. 37, no. 8, pp. 1439–1454, August 2010.

[161] Ruben Ruiz and Jose Antonio Vazquez-Rodriguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, August 16 2010.

[162] Tarek Chaari, Sondes Chaabane, Taicir Loukil, and Damien Trentesaux, "A genetic algorithm for robust hybrid flow shop scheduling," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. 9, pp. 821–833, 2011.

[163] Fuh-Der Chou, "Particle swarm optimization with coctail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks," *Int. Journal of Production Economics*, vol. 141, no. 1, pp. 137–145, January 2013.

[164] Min Dai, Dunbing Tang, Adriana Giret, Miguel A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, pp. 418–429, October 2013.

[165] P. Mellor, "A review of job shop scheduling," *Operational Research Quarterly*, vol. 17, no. 2, pp. 161–171, 1966.

[166] L. Gelders and P. R. Kleindorfer, "Coordinating aggregate and detailed scheduling decisions in one-machine job shop: 1. theory," *Operations Research*, vol. 22, no. 1, pp. 46–60, 1974.

[167] L. Gelders and P. R. Kleindorfer, "Coordinating aggregate and detailed scheduling in one-machine job shop: 2. computation and structure," *Operations Research*, vol. 23, no. 2, pp. 312–324, 1975.

[168] Jeffrey R. Barker and Graham B. McMahon, "Scheduling the general job-shop," *Management Science*, vol. 31, no. 5, pp. 594–598, 1985.

[169] R. Ramasesh, "Dynamic job shop scheduling - a survey of simulation research," *Omega-Int. Journal of Managmenet Science*, vol. 18, no. 1, pp. 43–57, 1990.

[170] V. J. Leon, S. D. Wu, and R. H. Storer, "Robustness measures and robust scheduling for job shops," *IIE Trans.*, vol. 26, no. 5, pp. 32–43, Sep. 1994.

[171] J. R. Jackson, "An extension of johnson's results on job lot scheduling," *Naval Research Logistics Quarterly*, vol. 3, pp. 201–203, 1956.

[172] John F. Muth and Gerald L. Thompson, *Industrial Scheduling*, Prentice-Hall, Englewood Clifs, 1963.

[173] Jacques Carlier, "The one-machine sequencing problem," *European Journal of Operational Research*, vol. 11, no. 1, pp. 42–47, 1982.

[174] A. Mascis and D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints," *European Journl of Operational Research*, vol. 143, no. 3, pp. 498–517, December 16 2002.

[175] Alan S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. 2, pp. 219–223, March–April 1960.

[176] Graham McMahon and Michael Florian, "On scheduling with ready times and due dates to minimize maximum lateness," *Operations Research*, vol. 23, no. 3, pp. 475–482, May-June 1975.

[177] B. J. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Job-shop scheduling by implicit enumeration," *Mgmt Science*, vol. 24, no. 4, pp. 441–450, 1977.

[178] John H. Blackstone, Don T. Phillips, and Gary L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *International Journal of Production Research*, vol. 20, no. 1, pp. 27–45, 1982.

[179] O. Holthaus and C. Rajendran, "Efficient dispatching rules for scheduling in a job shop," *International Journal of Production Economics*, vol. 48, no. 1, pp. 87–105, January 10 1997.

[180] A. P. Muhlemann, A. G. Lockett, and C. K. Farn, "Job shop scheduling heuristics and frequency of scheduling," *International Journal of Production Research*, vol. 20, no. 2, pp. 227–241, 1982.

[181] K. N. McKay, F. R. Safayeni, and J. A. Buzacott, "Job-shop scheduling theory – what is relevant," *Interfaces*, vol. 18, no. 4, pp. 84–90, JUL-AUG 1988.

[182] S. V. Mehta and R. M. Uzsoy, "Predictable scheduling of a job shop subject to breakdowns," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 365–378, June 1998.

[183] P. Brucker and R. Schlie, "Job-shop scheduling with multipurpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.

[184] N. Nasr and E. A. Elsayed, "Job shop scheduling with alternative machines," *Int. Journal of Production Research*, vol. 28, no. 9, pp. 1595–1609, 1990.

[185] R. H. Storer, S. D. Wu, and R. Vaccari, "New search spaces for sequencing problems with application to job shop scheduling," *Management Science*, vol. 38, no. 10, pp. 1495–1509, October 1992.

[186] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 1–13, 1993.

[187] E. Pinson, *Scheduling Theory and Applications*, chapter The Job Shop Scheduling Problem: A Concise Survey and Some Recent Developments, pp. 177–293, John Wiley, New York, 1995.

[188] I. Sabuncuoglu and M. Bayiz, "Job shop scheduling with beam search," *European Journal of Op. Research*, vol. 118, no. 2, pp. 390–412, October 16 1999.

[189] P. Brucker, B. Jurisch, and B. Sievers, "A branch-and-bound algorithm for the job-shop scheduling problem," *Discrete Applied Mathematics*, vol. 49, no. 1–3, pp. 107–127, March 30 1994.

[190] E Balas and A Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Mgmt Science*, vol. 44, no. 2, pp. 262–275, 1998.

[191] D. Dubois, H. Fargier, and H. Prade, "Fuzzy constraints in job-shop scheduling," *Journal of Int. Manufacturing*, vol. 6, no. 4, pp. 215–234, August 1995.

[192] M. Sakawa and R. Kubota, "Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms," *European Journal of Operational Research*, vol. 120, no. 2, pp. 393–407, January 16 2000.

[193] J. E. Biegel and J. J. Davern, "Genetic algorithms and job shop scheduling," *Computers & Industrial Engineering*, vol. 19, no. 1–4, pp. 81–91, 1990.

[194] U. Dorndorf and E. Pesch, "Evolution based learning in a job-shop scheduling environment," *Computers & Op. Research*, vol. 22, no. 1, pp. 25–40, 1995.

[195] C. Bierwirth, "A generalized permutation approach to job-shop scheduling with genetic algorithms," *OR Spektrum*, vol. 17, no. 2–3, pp. 87–92, 1995.

[196] R. W. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms - i. representation," *Computers & Industrial Engineering*, vol. 30, no. 4, pp. 983–997, September 1996.

[197] R. W. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: Hybrid genetic search strategies," *Comp. & Industrial Eng.*, vol. 36, no. 2, pp. 343–364, April 1999.

[198] W. J. Xia and Z. M. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, March 2005.

[199] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Computers & Operations Research*, vol. 30, no. 8, pp. 1151–1171, July 2003.

[200] B. J. Park, H. R. Choi, and H. S. Kim, "A hybrid genetic algorithm for the job shop scheduling problems," *Computers & Industrial Engineering*, vol. 45, no. 4, pp. 597–613, December 2003.

[201] Imen Essafi, Yazid Mati, and Stephane Dauzere-Peres, "A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem," *Computers & Op. Research*, vol. 35, no. 8, pp. 2599–2616, 2008.

[202] Jie Gao, Linyan Sun, and Mitsuo Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers & Operations Research*, vol. 35, no. 9, pp. 2892–2907, September 2008.

[203] Guohui Zhang, Liang Gao, and Yang Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, April 2011.

[204] D. N. Zhou, V. Cherkassky, T. R. Baldwin, and D. E. Olson, "A neural network approach to job-shop scheduling," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 175–179, January 1991.

[205] A. S. Jain and S. Meeran, "Job-shop scheduling using neural networks," *International Conference Journal of Production Research*, vol. 36, no. 5, pp. 1249–1272, May 1998.

[206] M. Kolonko, "Ssome new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, vol. 113, no. 1, pp. 123–136, February 16 1999.

[207] F. Pezzella and E. Merelli, "A tabu search method guided by shifting bottleneck for the job shop scheduling problem," *European Journal of Operational Research*, vol. 120, no. 2, pp. 297–310, January 16 2000.

[208] Jun-Qing Li, Quan-Ke Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 52, no. 5–8, pp. 683–697, February 2011.

[209] Mohammad Saidi-Mehrabad and Parviz Fattahi, "Flexible job shop scheduling with tabu search algorithms," *International Journal of Advanced Manufacturing Technology*, vol. 32, no. 5-6, pp. 563–570, MAR 2007.

[210] Chao Yong Zhang, Pei Gen Li, Zai Lin Guan, and Yun Qing Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Computers & Operations Research*, vol. 34, no. 11, pp. 3229–3242, November 2007.

[211] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, October 2008.

[212] Kuo-Ling Huang and Ching-Jong Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1030–1046, April 2008.

[213] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Systems with Applications*, vol. 37, no. 1, pp. 678–687, January 2010.

[214] Lele Zhang and Andrew Wirth, "On-line scheduling of two parallel machines with a single server," *Computers & Operations Research*, vol. 36, no. 5, pp. 1529–1553, May 2009.

[215] A. Bagheri, M. Zandieh, Iraj Mahdavi, and M. Yazdani, "An artificial immune algorithm for the flexible job-shop scheduling problem," *Future Generation Computer Systems - The International Journal of Grid Computing and eScience*, vol. 26, no. 4, pp. 533–541, April 2010.

[216] S. Q. Liu and E. Kozan, "A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem," *Journal of the Operational Research Society*, vol. 63, no. 2, pp. 168–182, February 2012.

[217] J. F. Chin and S. Meeran, "Integrating genetic programming into job shop scheduling problem," in *Advances in Manufacturing Technology - XVII*, Y. Qin and N. Juster, Eds., 2003, pp. 415–421.

[218] Shaohua Lu and Yun Xia, "Application of genetic programming on makespan optimization of job shop scheduling problem," in *Proceedings of the 6th International Conference on Innovation and Management, Vols I and II*, A. DeHoyos, Ed., 2009, pp. 1284–1291.

[219] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan, "A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems," in *2012 IEEE Congress on Evolutionary Computation*, 2012.

[220] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan, "Learning iterative dispatching rules for job shop scheduling with genetic programming," *International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1–4, pp. 85–100, July 2013.

[221] N. G. Hall, H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: Complexity and steady state analysis," *European Journal of Operational Research*, vol. 109, no. 1, pp. 43–65, August 16 1998.

[222] Y. Crama and J. Van de Klundert, "Cyclic scheduling of identical parts in a robotic cell," *Operations Research*, vol. 45, no. 6, pp. 952–965, 1997.

[223] M. Dawande, H. N. Geismar, S. P. Sethi, and C. Sriskandarajah, "Sequencing and scheduling in robotic cells: Recent developments," *Journal of Scheduling*, vol. 8, no. 5, pp. 387–426, October 2005.

[224] N. G. Hall, H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: Classification, two and three machine cells," *Operations Research*, vol. 45, no. 3, pp. 421–439, May–June 1997.

[225] M. H. Fazel Zarandi, H. Mosadegh, and M. Fattahi, "Two-machine robotic cell scheduling problem with sequence-dependent setup times," *Computers & Operations Research*, vol. 40, no. 5, pp. 1420–1434, May 2013.

[226] M. A. Manier and C. Bloch, "A classification for hoist scheduling problems," *International Journal of Flexible Manufacturing Systems*, vol. 15, no. 1, pp. 37–55, January 2003.

[227] Adnen El Amraoui, Marie-Ange Manier, Abdellah El Moudni, and Mohamed Benrejeb, "A linear optimization approach to the heterogeneous r-cyclic hoist scheduling problem," *Computers & Industrial Engineering*, vol. 65, no. 3, pp. 360–369, July 2013.

[228] Adnen El Amraoui, Marie-Ange Monier, Abdellah El Moudni, and Mohamed Benrejeb, "A genetic algorithm approach for a single hoist scheduling problem with time windows constraints," *Engineering Applications of Artificial Intellingence*, vol. 26, no. 7, pp. 1761–1771, August 2013.

[229] A. Che and C. Chu, "Single-track multi-hoist scheduling problem: A collision-free resolution based on a branch-and-bound approach," *International Journal of Production Research*, vol. 42, no. 12, pp. 2435–2456, June 15 2004.

[230] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, May 1 2006.

[231] T. Le-Anh and M. B. M. de Koster, "A review of design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 171, no. 1, pp. 1–23, May 16 2006.

[232] B. S. Manda and U. S. Palekar, "Collision-free routing in automated guided vehicle systems," in *2nd Industrial Engineering Research Conference Proceedings*, D. A. Mitta, L. I. Burke, J. R. English, J. Gallimore, G. A. Klutke, and G. L. Tonkay, Eds., 1993, pp. 510–514.

[233] P. S. Lee and L. L. Wang, "Collision-avoidance by fuzzy-logic control for automated guided vehicle navigation," *Journal of Robotic Systems*, vol. 11, no. 8, pp. 743–760, December 1994.

[234] N. Q. Wu and M. C. Zhou, "Modeling and deadlock control of automated guided vehicle systems," *IEEE-ASME Transactions on Mechatronics*, vol. 9, no. 1, pp. 50–57, March 2004.

[235] N. Q. Wu and M. C. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, vol. 35, no. 6, pp. 1193–1202, December 2005.

[236] Chang W. Kim and J. M. A. Tanchoco, "Conflict-free shortest-time bidirectional agv routeing," *International Journal of Production Research*, vol. 29, no. 12, pp. 2377–2391, 1991.

[237] Laiguang Zeng, Hsu-Pin Wang, and Song Jin, "Conflict detection of automated guided vehicles: a petri net approach," *International Journal of Production Research*, vol. 29, no. 5, pp. 866–879, 1991.

[238] Sophie Parragh, Karl Doerner, and Richard Hartl, "A survey on pickup and delivery problems: Part i: Transportation between customers and depot," *Journal für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, April 2008.

[239] Sophie Parragh, Karl Doerner, and Richard Hartl, "A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations," *Journal für Betriebswirtschaft*, vol. 58, no. 2, pp. 81–117, June 2008.

[240] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins, "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis," *European Journal of Op. Research*, vol. 231, no. 1, pp. 1–21, 2013.

[241] M. Gendreau and C. Tarantilis, "Solving large-scale vehicle routing problem with time windows: The stat-of-the-art," Tech. Rep., CIRRELT, 2010.

[242] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti, "New route relaxation and pricing strategies for the vehicle routing problem," *Operations Research*, vol. 59, no. 5, pp. 1269–1283, 2011.

[243] P. P. Repoussis, C. D. Tarantilis, and G. Loannou, "Arc-guided evolutionary algorithm for the vehicle routing problem with time windows," *Evolutionary Computation, IEEE Transactions on*, vol. 13, no. 3, pp. 624–647, 2009.

[244] Yuichi Nagata, Olli Braysy, and Wout Dullaert, "A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 37, no. 4, pp. 724–737, 2010.

[245] Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin, "Vehicle dispatching with time-dependent travel times," *European Journal of Operational Research*, vol. 144, no. 2, pp. 379–396, 2003.

[246] S. R. Balseiro, I. Loiseau, and J. Ramonet, "An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows," *Computers & Operations Research*, vol. 38, no. 6, pp. 954–966, June 2011.

[247] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, MA, USA, 1st edition edition, December 1992.

[248] John R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, The MIT Press, Cambridge, MA, USA, May 1994.

[249] John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane, *Genetic Programming III: Darwinian Invention & Problem Solving*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, May 1999.

[250] Julian F. Miller, "An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach," in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO 1999)*, Orlando, Florida, USA, July 14-17 1999, pp. 1135–1142.

[251] Markus F. Brameier and Wolfgang Banzhaf, *Linear Genetic Programming*, Springer, New York, NY, USA, 1st edition, 2007.

[252] Ajith Abraham, Nadia Nedjah, and Luiza Mourelle, *Evolutionary Computation: from Genetic Algorithms to Genetic Programming*, vol. 13, chapter 1, pp. 1–20, Springer Berlin Heidelberg, 2006.

[253] Jarmo T. Alander, "An indexed bibliography of genetic programming," Report Series no 2008-1-GP, Department of Information Technology and Industrial Management, University of Vaasa, Finland, 2008.

[254] William B. Langdon, *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!*, vol. 1, Springer, Kluwer, Boston, USA, 1st edition, April 1998.

[255] David J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, pp. 199–230, June 1994.

[256] Conor Ryan and Michael O'Neill, "Grammatical evolution: A steady state approach," *Late Breaking Papers, Genetic Prog.*, vol. 2, pp. 180–185, 1998.

[257] Michael O'Neill and Conor Ryan, "Grammatical evolution," *IEEE Evolutionary Computation*, vol. 5, pp. 349–359, August 2001.

[258] Michael O'Neill, John Mark Swafford, James McDermott, Jonathan Byrne, Anthony Brabazon, Elizabeth Shotton, Ciaran McNally, and Martin Hemberg, "Shape grammars and grammatical evolution for evolutionary design," in *GECCO'09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. 2009, pp. 1035–1042, NY, USA.

[259] Peter A. Whigham, "Grammatically-based genetic programming," in *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Tahoe City, CA, USA, July 1995, pp. 33–41, Morgan Kaufmann.

[260] Nguyen Xuan Hoai, R. I. McKay, and H. A. Abbass, "Tree adjoining grammars, language bias, and genetic programming," in *Genetic Programming, Proceedings of EuroGP'2003*, Essex, April 14–16 2003, vol. 2610 of *LNCS*, pp. 335–344, Springer-Verlag.

[261] Nguyen Xuan Hoai, R. I. (Bob) McKay, and Daryl Essam, "Representation and structural difficulty in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 157–166, April 2006.

[262] Hitoshi Iba, "Random tree generation for genetic programming," in *Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation*, Germany, 1996, vol. 1141 of *LNCS*, pp. 144–153, Springer-Verlang.

[263] Walter Bohm and Andreas Geyer-Schulz, *Exact Uniform Initialization For Genetic Programming*, pp. 379–407, Morgan Kaufman, August 1996.

[264] Kumar Chellapilla, "Evolving computer programs without subtree crossover," *IEEE Tran. on Evolutionary Computation*, vol. 1, no. 3, pp. 209–216, 1997.

[265] Sean Luke, "Two fast tree-creation algorithms for genetic programming," *IEEE Trans. on Evolutionary Comp.*, vol. 4, no. 3, pp. 274–283, Sep. 2000.

[266] Peter A. Whigham, "Inductive bias and genetic programming," in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, September 1995, vol. 414, pp. 461–466.

[267] Marc García-Arnou, Daniel Manrique, Juan Rios, and Alfonso Rodríguez-Patón, "Initialization method for grammar-guided genetic programming," *Knowledge-Based Systems, Elsevier*, vol. 20, no. 2, pp. 127–133, March 2007.

[268] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, 1st edition, November 2008.

[269] Sean Luke and Lee Spector, "A comparison of crossover and mutation in genetic programming," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. July 1997, pp. 240–248, Morgan Kaufmann.

[270] Sean Luke and Lee Spector, "A revised comparison of crossover and mutation in genetic programming," in *Genetic Programming 1998: Proceedings of the Third Annual Conference*. July 1998, pp. 208–213, Morgan Kaufmann.

[271] William M. Spears and Kenneth A. DeJong, "An analysis of multi-point crossover," in *Foundations of Genetic Algorithms*, pp. 301–315. Morgan Kaufmann, San Mateo, CA, USA, 1991.

[272] Kit Yan Chan and Terence C. Fogarty, "Experimental design based multi-parent crossover operator," in *Genetic Programming, Proceedings of EuroGP'2003*, Essex, April 14–16 2003, vol. 2610 of *LNCS*, pp. 297–306.

[273] A. E. Eiben and C. H. M. Van Kemenade, "Diagonal crossover in genetic algorithms for numerical optimization," *Journal of Control and Cybernetics*, vol. 26, no. 3, pp. 447–465, 1997.

[274] Riccardo Poli and William B. Langdon, "Genetic programming with one-point crossover," in *Soft Computing in Engineering Design and Manufacturing*, Godalming, UK, June 1997, pp. 180–189, Springer-Verlang London.

[275] Riccardo Poli and William B. Langdon, "Schema theory for genetic programming with one-point crossover and point mutation," *Evolutionary Computation*, vol. 6, no. 3, pp. 231–252, 1998.

[276] William B. Langdon, "Size fair and homologous tree crossovers for tree genetic programming," *Gen. Prog. and Evo. Machines*, vol. 1, pp. 95–119, April 2000.

[277] Lee Spector, *Advances in Genetic Programming 3*, The MIT Press, Cambridge, MA, USA, June 1999.

[278] Patrik D'haeseleer, "Context preserving crossover in genetic programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, Orlando, Florida, USA, 1994, vol. 1, pp. 256–261, IEEE Press.

[279] Daniel Manrique, Fernando Marquez, Juan Rios, and Alfonso Rodríguez-Patón, "Grammar based crossover operator in genetic programming," in *Artificial Intelligence and Knowledge Engineering Applications*, José Mira and José R. Álvarez, Eds. June 2005, vol. 3562 of *Lecture Notes in Computer Science*, pp. 252–261, Springer-Verlag Berlin Heidelberg.

[280] Jorge Couchet, Daniel Manrique, Juan Rios, and Alfonso RodrígueePatón, "Crossover and mutation operators for grammar-guided genetic programming," *Soft Computing*, vol. 11, no. 10, pp. 943–955, August 2007.

[281] Nguyen Quang Uy, Nguyen Xuan Hoai, and Michael O'Neill, "Semantic aware crossover for genetic programming: The case for real-valued function regression," in *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, Taubingen, April 2009, EvoStar, vol. 5481 of *Lecture Notes in Computer Science*, pp. 292–302, Springer Berlin Heidelberg.

[282] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, R. I. McKay, and Edgar Galvan-Lopez, "Semantically-based crossover in genetic programming: Application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, June 2010.

[283] Peter J. Angeline, "Subtree crossover: Building block engine or macromutation?," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. July 13-16 1997, pp. 9–17, Morgan Kaufmann.

[284] S. Openshaw and I. Turton, "Building new spatial interaction models using genetic programming," in *Evolutionary Computing, AISB workshop*, Leeds, UK, April 11–13 1994, AISB, LNCS, p. 10, Springer-Verlag.

[285] Lawrence Beadle and Colin G. Johnson, "Semantically driven mutation in genetic programming," in *2009 IEEE Congress on Evolutionary Computation*, Norway, 18-21 May 2009, IEEE Comp. Intelligence Society, pp. 1336–1342.

[286] Riccardo Poli and Nicholas Freitag McPhee, "General schema theory for genetic programming with subtree-swapping crossover: Part i," *Evolutionary Computation*, vol. 11, no. 1, pp. 53–66, Mar. 2003.

[287] Riccardo Poli and Nicholas Freitag McPhee, "General schema theory for genetic programming with subtree-swapping crossover: Part ii," *Evolutionary Computation*, vol. 11, no. 2, pp. 169–206, May 2003.

[288] Peter J. Angeline and Jordan B. Pollack, "Evolutionary induction of subroutines," in *Proceedings of the 14th Annual Cognitive Science Conference*, Indiana, USA, 1992, pp. 236–241.

[289] Stephen Dignum and Riccardo Poli, "Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO'2007)*, London, UK, July 7–11 2007, vol. 2, pp. 1588–1595, ACM Press.

[290] Riccardo Poli, William B. Langdon, and Stephen Dignum, "On the limiting distribution of program sizes in tree-based genetic programming," in *Genetic Programming, 10th European Conference, EuroGP 2007*, April, 11-13 2007, vol. 4445 of *Lecture Notes in Computer Science*, pp. 193–204.

[291] Terence Soule and Robert B. Heckendorn, "An analysis of the causes of code growth in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 3, no. 3, pp. 283–309, 2002.

[292] Sean Luke, "Code growth is not caused by introns," in *In Whitley, D. (Ed.), Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference (pp. 228–235). Las Vegas.* 2000, pp. 228–235, Morgan Kaufmann.

[293] Chris Gathercole and Peter Ross, "Small populations over many generations can beat large populations over few generations in genetic programming," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*, San Francisco, CA, USA, July 13–16 1997, pp. 111–119, Morgan Kaufmann.

[294] Cemal Ozguven, Lale Ozbakir, and Yasemin Yavuz, "Mathematical models for job-shop scheduling problems with routing and process plan flexibility," *Applied Mathematical Modelling*, vol. 34, no. 6, pp. 1539–1548, 2010.

[295] Onur Ozturk, Marie-Laure Espinouse, Maria Di Mascolo, and Alexia Gouin, "Makespan minimisation on parallel batch processing machines with non-identical job sizes and release dates," *International Journal of Production Research*, vol. 50, no. 20, pp. 6022–6035, 2012.

[296] Parviz Fattahi, Mohammad Saidi Mehrabad, and Fariborz Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 18, no. 3, pp. 331–342, 2007.

[297] Amir Jalilvand-Nejad and Parviz Fattahi, "A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem," *Journal of Intelligent Manufacturing*, pp. 1–14, 2013.

[298] T. I. Kopaleishvili, *Collision Theory: A Short Course*, World Scientific Publishing Co Pte Ltd, December 1994.

[299] Marvin L. Goldberger and Kenneth M. Watson, *Collision Theory*, Dover Publications Inc., September 2004.

[300] Robert Sedgewick and Wayne Kevin, *Algorithms*, Addison-Wesley Professional, March 2011.

[301] César Estébanez, Julio César Hernández-Castro, Arturo Ribagorda, and Pedro Isasi, "Evolving hash functions by means of genetic programming," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2006, GECCO '06, pp. 1861–1862, ACM.

[302] César Estébanez, Julio César Hernández-Castro, Arturo Ribagorda, and Pedro Isasi, "Finding state-of-the-art non-cryptographic hashes with genetic programming," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*. 2006, pp. 818–827, Springer-Verlag.

BIBLIOGRAPHY OF AUTHOR

[303] Jan Karásek, "Portfolio selection based on analytic hierarchy process and evolutionary computation," in *Proceedings of the 17th Conference Student EEICT 2011*, Brno, Czech Republic, 2011, vol. 3, pp. 595–599.

[304] Jan Karásek and Lubomír Cvrk, "Stav vědy a techniky v oblasti genetického programování," *Elektrorevue*, vol. 15, no. 2, pp. 147–155, 2013.

[305] Jan Karásek, Radim Burget, Jakub Mūller, and Victor Manuel Peris Montalt, "Automatization of vehicle routing in warehouse," *Elektrorevue*, vol. 2, no. 1, pp. 1–8, April 2011.

[306] Radim Burget, Jan Karásek, and Zdeněk Smékal, "Classification and detection of emotions in czech news headlines," in *Proceedings of the 33rd International Conference on Telecommunication and Signal Processing*, 2010, pp. 64–69.

[307] Radim Burget, Jan Karasek, and Zdeněk Smékal, "Recognition of emotions in czech newspaper headlines," *Radioengineering*, vol. 20, no. 1, pp. 31–39, April 2011.

[308] Radim Burget, Jan Karásek, Zdeněk Smékal, Václav Uher, and Otto Dostál, "Rapidminer image processing extension: A platform for collaborative research," in *Proceedings of the 33rd International Conference on Telecommunication and Signal Processing*, 2010, pp. 114–118.

[309] Radim Burget, Jan Karásek, Václav Uher, and Martin Zukal, "Semi–automatic image data analysis," *Elektrorevue*, vol. 2010, no. 1, pp. 61–67, 2010.

[310] Jan Mašek, Radim Burget, Jan Karásek, Václav Uher, and Selda Guney, "Evolutionary improved object detector for ultrasound images," in *36th International Conference on Telecommunications and Signal processing (TSP 2013)*, July, pp. 586–590.

[311] Jan Karásek, Radim Burget, Jan Mašek, and Malay Kishore Dutta, "Genetic programming based classifier in viola-jones rapidminer image mining extension," in *36th International Conference on Telecommunications and Signal Processing (TSP 2013)*, July 2013, pp. 872–876.

[312] Michal Jurčík and Jan Karásek, "Návrh metody pro výpočet predikce kolizí vozíků v logistickém skladu založena na simulaci neelastických kolizí," *Elektrorevue*, vol. 15, no. 3, pp. 194–204, 2013.

[313] Jan Karásek, Radim Burget, Malay Kishore Dutta, and Anushikha Singh, "Java evolutionary framework based on genetic programming," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, February 2014, pp. ?–?

[314] Jan Karásek, Radim Burget, and Ondřej Morský, "Towards an automatic design of non-cryptographic hash function," in *34th International Conference on Telecommunications and Signal Processing (TSP 2011)*, 2011, pp. 19–23.

[315] Jan Karásek and Radek Beneš, "Image filter design based on evolution," in *Proceedings of the 16th Conference Student EEICT 2010*, Brno, Czech Republic, 2010, vol. 5, pp. 251–255.

[316] Jan Karásek, Radim Burget, Radek Beneš, and Luboš Nagy, "Grammar guided genetic programming for automatic image filter design," in *The Conference Proceedings of the Knowledge in Telecommunication Technologies and Optics*, Miroslav Vozňák and Jan Skapa, Eds., Ostrava, Czech Rep., December 2010, pp. 205–210, VŠB–Technical University of Ostrava.

[317] Radek Beneš, Jan Karásek, Radim Burget, and Kamil Říha, "Automatically designed machine vision system for the localization of cca transverse section in ultrasound images," *Computer Methods and Programs in Biomedicine*, vol. 109, no. 1, pp. 92–103, 2013.

[318] Jan Karásek, Radim Burget, and Václav Uher, "Optimization of warehouse processes – benchmark definition," in *MENDEL 2013 19th International Conference on Soft Computing*, June 2013, vol. 1, pp. 45–50.

[319] Jan Karásek, Radim Burget, Václav Uher, Malay Kishore Dutta, and Yogesh Kumar, "Optimization of logistic distribution centers process planning and scheduling," in *2013 Sixth International Conference on Contemporary Computing (IC3-2013)*, August 2013, pp. 343–348.

[320] Jan Karásek, Radim Burget, and Lukáš Povoda, "Logistic warehouse process optimization through genetic programming algorithm," in *Advances in Intelligent Systems and Computing*, April 2014, pp. ?–?, ISSN 2194-5357.

LIST OF ABBREVIATIONS

AAO Architecture-Altering Operations

ACO Ant Colony Optimization

ADF Automatically Defined Functions

AGV Automated Guided Vehicles

ASRS Automated Storage and Retrieval System

BB Branch and Bound

BBO Biogeography-Based Optimization

CCA Common Carotid Artery

CFG Context-Free Grammar

CGP Cartesian Genetic Programming

CPC Context Preserving Crossover

CPM Critical Path Method

DARP Dial-A-Ride Problem

EP Evolutionary Programming

ERP Enterprise Resource Planning

ES Evolution Strategies

FIFO First In First Out

FS Flow Shop

FSS Flow Shop Scheduling

GA Genetic Algorithm

GBIM Grammar-based Initialization Method

GBM Grammar-based Mutation

GBX Grammar-based Crossover

GGGP Grammar Guided Genetic Programming

GP Genetic Programming

JS Job Shop

JSS Job-Shop Scheduling Problem

LGP Linear Genetic Programming

LPT Longest Processing Path first

LR Lagrangian Relaxation

MIP Mixed Integer Programming

NN Neural Network

NS Neighborhood Search

OM Operational Manager

OS Open Shop

OSS Open Shop Scheduling

PDP Pickup and Delivery Problem

PERT Program Evaluation and Review Technique

PMM Parallel Machine Model

PSO Particle Swarm Optimization

PTC Probabilistic Tree Creation

RPT Reverse Processing Path first (Reverse SPT)

SA Simulated Annealing

SAC Semantics Aware Crossover

SDM Semantically Driven Mutation

SKU Stock Keeping Unit

SMM Single Machine Model

SPT Shortest Processing Path first

SSC Semantic Similarity-based Crossover

STGP Strongly Typed Genetic Programming

TDVRP Time Dependent Vehicle Routing Problem

TGP Tree Genetic Programming

TS Tabu Search

TSP Traveling Salesman Problem

VRP Vehicle Routing Problem

VRPB Vehicle Routing Problem with Backhauls

VRPPD Vehicle Routing Problem with Pickups and Deliveries

VRPTW Vehicle Routing Problem with Time Windows

WMS Warehouse Management System

WX Whigham's Crossover

LIST OF APPENDICES

CD

- The Java Evolutionary Framework (JEF)
- The Basic Instructions for JEF

Curriculum Vitae

Jan Karásek

Personal information:

Address:	Rosice, Czech Republic
E-mail:	karasekj@gmail.com
GSM:	+420 777 933 554

School and education:

09/2009-present	Brno University of Technology, Ph.D. Faculty of Electrical Engineering and Communication Specialization: Teleinformatics Theme of thesis: High-Level Object Oriented Genetic Programming in Logistic Warehouse Optimization
09/2011-01/2012	University of Eastern Finland, Ph.D. exchange stay School of Computing Specialization: Teleinformatics Theme of thesis: High-Level Object Oriented Genetic Programming in Logistic Warehouse Optimization
07/2008-06/2010	Brno University of Technology, MSc. Faculty of Business and Management Specialization: Economics and Management Theme of thesis: The Application of Evolution Algorithm for the Rating of Supplier of the Firm
07/2007-06/2009	Brno University of Technology, MSc. Faculty of Electrical Engineering and Communication Specialization: Communications and Informatics Theme of thesis: Hash Functions - Characteristics, Implementation and Collisions
07/2004-06/2007	Brno University of Technology, BSc. Faculty of Electrical Engineering and Communication Specialization: Teleinformatics Theme of thesis: Multi–Layer Desktop Applications

Work experience:

01/2013-present	Junior Researcher, Brno University of Technology
05/2011-present	Project Manager, Brno University of Technology
06/2005-01/2009	Programmer – Analyst of Logistic Inf. System Prystanis

Project participation

04/2012-12/2015	FR-TI4/151: Research and Development of Technology for Machine Emotion Detection in Unstructured Data Principal investigator: Prof. Ing. Zdeněk Smékal, CSc.
01/2011-12/2014	FEKT-S-11-17: Research of Sophisticated Methods for Digital Audio and Image Signal Processing Principal investigator: Prof. Ing. Zdeněk Smékal, CSc.
05/2011-05/2014	EE2.3.20.0094: Support for Incorporating R&D Teams in International Cooperation in the Area of Image and Audio Signal Processing Principal investigator: Ing. Jan Karásek
01/2013-12/2013	ED2.1.00/03.0072, Center of Sensor, Information and Communication Systems, Principal investigator: Prof. Dr. Ing. Zbyněk Raida
08/2009-07/2013	FR-TI1/444: Research and Development of the System for Manufacturing Optimization, Principal investigator: Prof. Ing. Zdeněk Smékal, CSc.
01/2012-12/2012	2272/2012/F1: Integration of Genetic Algorithms and Genetic Prog. in Teaching, Principal investigator: Ing. Radim Burget, Ph.D.
01/2011-12/2011	2076/2011/G1: Integration of Evolutionary Algorithms in Teaching Principal investigator: Ing. Jan Karásek
01/2010-12/2010	FEKT-S-10-16: Research of Communication Systems and Networks Principal investigator: Prof. Ing. Kamil Vrba, CSc.

Designated reviewer:

- 6th International Conference on Teleinformatics (ICT 2011)
- 35th International Conference on Telecommunications and Signal Processing (TSP 2012)
- 36th International Conference on Telecommunications and Signal Processing (TSP 2013)
- 37th International Conference on Telecommunications and Signal Processing (TSP 2014)
- World Symposium on Computer Applications & Research (WSCAR' 2014)
- International Conference on Signal Processing & Integrated Networks (SPIN-2014)
- Elektrorevue – Internet Electrotechnics Magazine, ISSN 1213-1539
- IJATES – Interntet All-Electronic Journal, ISSN 1805-5443

Invited Presentations:

- 2013, Amity University, Noida, New Delhi, India
- 2014, University of Stirling, Stirling, Great Britain
- 2014, University of Gran Canaria, Gran Canaria, Spain

Publication activities:

- Papers published in impact factor journals: **2**
- Papers published in other journals: **9**
- Papers published in international conferences: **18**
- Papers published in domestic conferences: **3**
- Papers indexed in WoS: **5**
- Papers indexed in Scopus: **7**
- H-index according to WoS: **2**
- H-index according to Scopus: **2**
- Number of released products: **5**

Last Actualization: March 21, 2014