BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

Ing. Jan Karásek

HIGH-LEVEL OBJECT ORIENTED GENETIC PROGRAMMING IN LOGISTIC WAREHOUSE OPTIMIZATION

VYSOKOÚROVŇOVÉ OBJEKTOVĚ ORIENTOVANÉ GENETICKÉ PROGRAMOVÁNÍ PRO OPTIMALIZACI LOGISTICKÝCH SKLADŮ

SHORTENED VERSION OF PH.D. THESIS

Specialization: Teleinformatics

Supervisor: Ing. Radim Burget, Ph.D.

Opponents:

Date of Defense:

KEYWORDS

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

KLÍČOVÁ SLOVA

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

Dissertation is available at the Science Department of Dean's Office FEEC, Brno University of Technology, Technická 10, Brno, 616 00

Disertační práce je k dispozici na Vědeckém oddělení děkanátu FEKT, Vysoké učení technické v Brně, Technická 10, Brno, 616 00

ISBN 80-214-
ISSN 1213-4198

CONTENTS

1 INTRODUCTION

The proposed thesis 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. Besides, thesis is a part of the project FR-TI1/444 "Research and Development of the System for Manufacturing Optimization".

According to Tompkins et al. [1] 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 %). The most obvious part to optimize 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 etc., 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. The first motivational 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 manager with an automated optimization algorithm. The second motivationl factor is to bring the latest and most effective concepts of optimization in scheduling from the scientific world into the logistics world.

Of course, there are also fully automated logistic warehouses supported by various robots, hoists, automated guided vehicles, automated storage and retrieval systems etc., 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.

1.1 Research Challenges

The first challenge is to design an automated optimization algorithm and support the decision making of operational manager. If the results of algorithm will surpass the results of the manager, the challenge will be to deploy the algorithm as a stand-alone system without any supervision. The second challenge is to develop this algorithm as a general problem solving algorithm with a possibility to define new criteria how to influence the convergence to the solution. This should be done by multi-criteria fitness function which will take into account various parameters such as: the makespan, utilization o employees and equipment, and possible warehouse congestions. The fourth challenge is to reach better warehouse performance and productivity.

1.2 Research Contribution

A comprehensive literature review of the warehouse optimization connected to the scheduling and the vehicle routing was written. This helped to define the hypothesis and goals, and to extend the mathematical model for the warehouse work-flow optimization. The new, extensible, and multi-platform Evolutionary Framework with the computational core based on Genetic Programmin (GP) driven by the Context-Free Grammar (CFG) based on existing Grammar Guided GP (GGGP) approach was developed, implemented, and validated. The new algorithm for the warehouse work-flow optimization problem based on the proposed framework 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. The set of benchmark tests was created as well as the evaluation process, which together give the reference baseline of the results for the optimization.

2 THEORETICAL BACKGROUND

According to [2] 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 Warehouse Optimization

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 [3] is a key component of further optimization tasks and has a significant impact on order-picking and traveling distances in the warehouse. The layout is usually of rectangular shape and based on pallet manipulation [4]. According to [5] and [6] 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. All the equipment is standardized according to the dimensions, but the standardization is mostly only for a specific continent.

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: *random* and *dedicated*. While random strategy allows to store a pallet to an arbitrary empty location with the same probability or to the closest empty location [5], the dedicated strategy allows to store a pallet only to specified locations [7]. 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 [8]. Since the algorithm can be applied only to conventional warehouses, the problem is mostly solved by heuristic methods. The common routing methods were described in [9]. All the methods were primarily developed for single-block warehouses. Modified methods for multiple-block warehouses were proposed in [10].

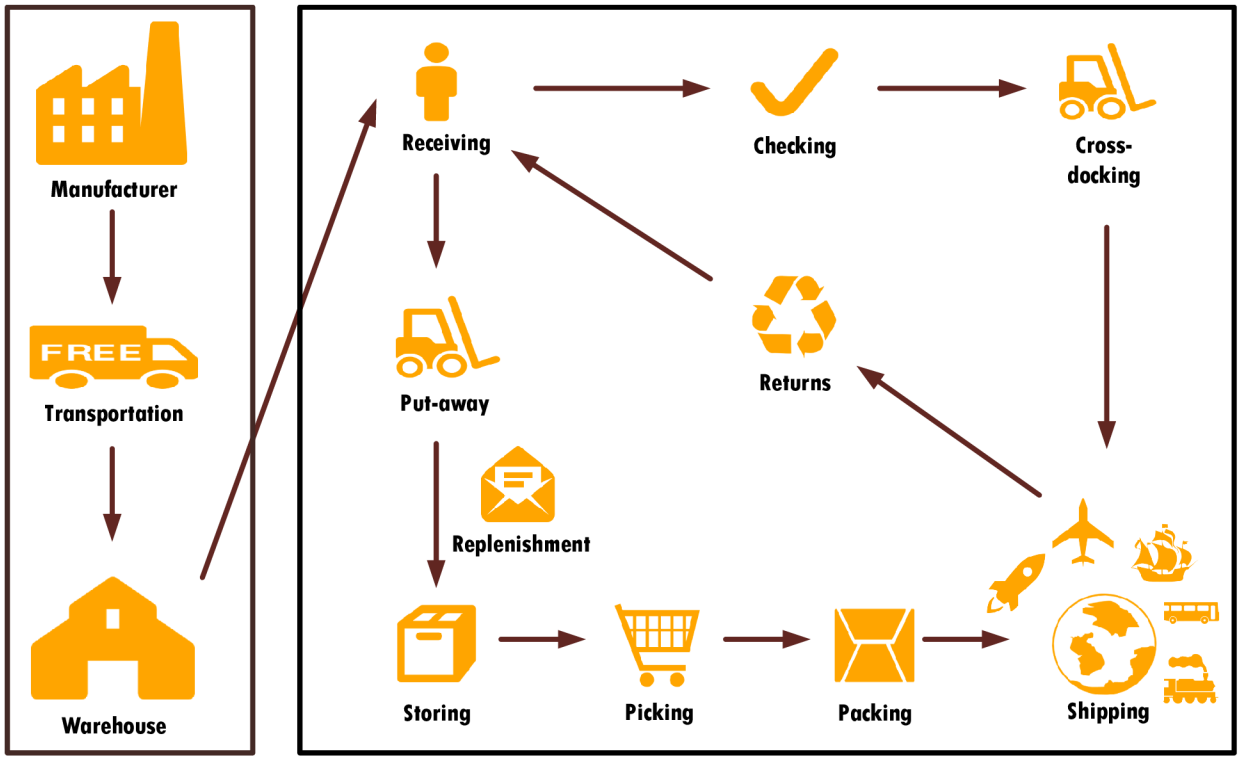

Fig. 2.1: An example of the logistic flow from the manufacturer to the warehouse.

The basic processes in the warehouse are receiving, storing, putting-away, picking/retrieving and shipping goods (see Fig.2.1). 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.

Receiving 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. **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. **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 operating costs [4]. **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. textbfCross-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. 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 [11] 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 [11].

The relationship between pick density and throughput, which has demonstrated the significance of blocking, was determined in [12]. The throughput analysis for order-picking with multiple pickers and aisle congestion is investigated in [13]. The batching in the narrow-aisle order-picking system with picker blocking consideration was investigated in [14]. The alorithm for two order pickers with consideration of congestions was proposed in [15]. 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. Also the online version of the multiple picking agents for warehouse was studied in [16]. The dynamic order-picking and cost reduction generated by optimal policies is discussed in [17]. Another heuristic approach for online batching based on offline batching is introduced in [18] and [19]. 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 makespan. A*-algorithm for routing and Simulated Annealing for batching were proposed in [20].

2.2 The Job Shop 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. This thesis is mostly inspired by *Job Shop Scheduling (JSS)* which 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. The general JSS problem is very hard to solve optimally, the 10-job 10-machine problem formulated in 1963 [21] was solved for the first time in [22]. 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 JSS is possible by disjunctive graph models. The special emphasis is given on machine blocking and no-wait constraints.

The JSS was often solved by linear programming, constrained programming, disjunctive programming, shifting bottleneck approaches algorithms. The problem was also solved by many algorithms from are of artificial intelligence and machine learning, such as: Tabu Search [23], Simulated Annealing and Genetic Algorithms [24], Particle Swarm Optimization [25], Shifting Bottleneck Algorithms [26], Biogeography-Based Optimization [27] and many other algorithms and their combinations.

One of the first integration of GP into JSS was introduced in [28] (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 [29] (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 [30] (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 [31] (2013) also discussed dispatching rules, in particular the iterative learning.

2.3 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 chapter 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.

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 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. The proposed GP algorithm as well as the Evolutionary Framework is described in [37], [38]. The algorithm was then adjusted to the JSS applied in logistics, which is described in [39]. The proposed GP algorithm was tested also on other examples, such as text processing and the emotion recognition [40], [41], and image processing and image mining [42], [43], [44], and [45].

3 THE OBJECTIVES OF DISSERTATION

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 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** enhances the standard mathematical model, 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.
2. **Involvement of the multi-criteria fitness function for 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 collisions, as well as other factors such as the balanced workload of employees.
3. **The variability of time planning and simulation (minutes/hours/shift).** 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.
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.
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 GP algorithm driven by CFG. 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 has to be validated, and then adapted in the way that they could work with a defined JSS problem applied to the logistic warehouses.
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 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 is also described in the next 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:

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

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

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

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

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

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

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

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

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

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

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

$$x_{i,j,h,k} \; \in \; \{0, 1\} \quad (\forall \; i, \; j, \; h, \; k); \quad y_{i,j,h} \; \in \; \{0, 1\} \quad (\forall \; i, \; j, \; h); \tag{4.12}$$

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, the processing time of all operations is not a negative value, and the operation which is assigned to a machine can start immediately, if the machine is in idle time. Eq. 4.12 describes that $O_{j,h}$ is performed on machine i with priority k and machine i which is selected for this operation.

The JSS model described in the previous text 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:**

$$z_{e,j,h} \leq b_{e,j,h} \quad (\forall\, e,\ j,\ h); \tag{4.13}$$

$$c_{e,i} = 1 \quad (\forall\, e,\ i); \tag{4.14}$$

$$y_{i,j,h} \cdot z_{e,j,h} = c_{e,i} \quad (\forall\, e,\ i,\ j,\ h); \tag{4.15}$$

$$\sum_{e} z_{e,j,h} = 1 \quad (\forall\, j,\ h); \tag{4.16}$$

$$D_j \geq C_j \quad (\forall\, j); \tag{4.17}$$

$$\bar{D}_j \geq D_j \quad (\forall\, j); \tag{4.18}$$

$$z_{e,j,h} \in \{0, 1\} \quad (\forall\, e,\ j,\ h); \tag{4.19}$$

Constraint Eq. 4.13 declares a suitable employee for each operation. Constraint Eq. 4.14 ensures that employee e is authorized for machine i. Constraint Eq. 4.15 ensures that the selected employee is authorized for the selected machine. Constraint Eq. 4.16 ensures that all operations will be performed only by one employee. Constraint Eq. 4.17 ensures that the due date is less or equal to the completion time of the specific job. Constraint Eq. 4.18 ensures that the deadline for job j has to be fulfilled, and the Eq. 4.19 describes the parameter interval of the selected employee for operation $O_{h,j}$.

5 THE COLLISION PREDICTION ALGORITHM

Collision is a phenomenon, limited in time and space, in which two or more objects mutually affect each other [32]. 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. This section is based on the paper [33].

5.1 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.

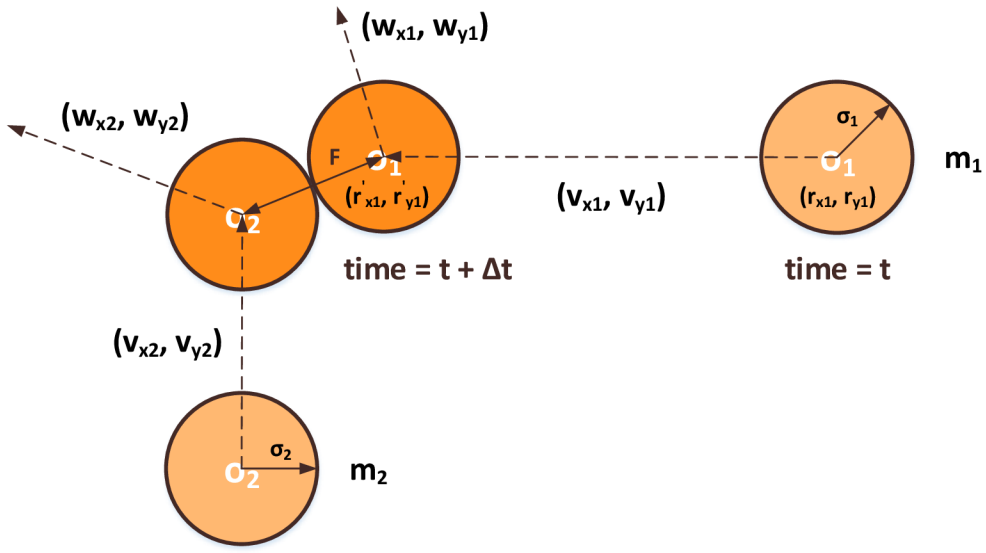

Fig. 5.1: Another collision in a 2-dimensional environment.

The situation is depicted in Fig. 5.1, 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 $\sigma = \sigma_1 + \sigma_2$:

$$\sigma^2 = (r'_{x1} - r'_{x2})^2 + (r'_{y1} - r'_{y2})^2 \ . \tag{5.1}$$

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

$$\begin{aligned} r'_{x1} &= r_{x1} + \Delta t \times v_{x1} \ , \\ r'_{y1} &= r_{y1} + \Delta t \times v_{y1} \ , \\ r'_{x2} &= r_{x2} + \Delta t \times v_{x2} \ , \\ r'_{y2} &= r_{y2} + \Delta t \times v_{y2} \ . \end{aligned} \tag{5.2}$$

Substituting Eq. 5.2 with Eq. 5.1 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.5.3:

$$\Delta t = \begin{cases} \infty & \text{if } \Delta v \times \Delta r \geq 0, \\ \infty & \text{if } d < 0, \\ -\frac{\Delta v \times \Delta r + \sqrt{d}}{\Delta v \times \Delta v} & \text{otherwise,} \end{cases} \tag{5.3}$$

where

$$\begin{aligned} d &= (\Delta v \times \Delta r)^2 - (\Delta v \times \Delta v) \times (\Delta r \times \Delta r - \sigma^2) \ , \\ \Delta r &= (\Delta r_x, \Delta r_y) = (r_{x2} - r_{x1}, r_{y2} - r_{y1}) \ , \\ \Delta v &= (\Delta v_x, \Delta v_y) = (v_{x2} - v_{x1}, v_{y2} - v_{y1}) \ , \\ \Delta r \times \Delta r &= (\Delta r_x)^2 + (\Delta r_y)^2 \ , \\ \Delta v \times \Delta v &= (\Delta v_x)^2 + (\Delta v_y)^2 \ , \\ \Delta v \times \Delta r &= (\Delta v_x) \times (\Delta r_x) + (\Delta v_y) \times (\Delta r_y). \end{aligned} \tag{5.4}$$

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 $\Delta t > 0$.

5.2 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 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 racks.

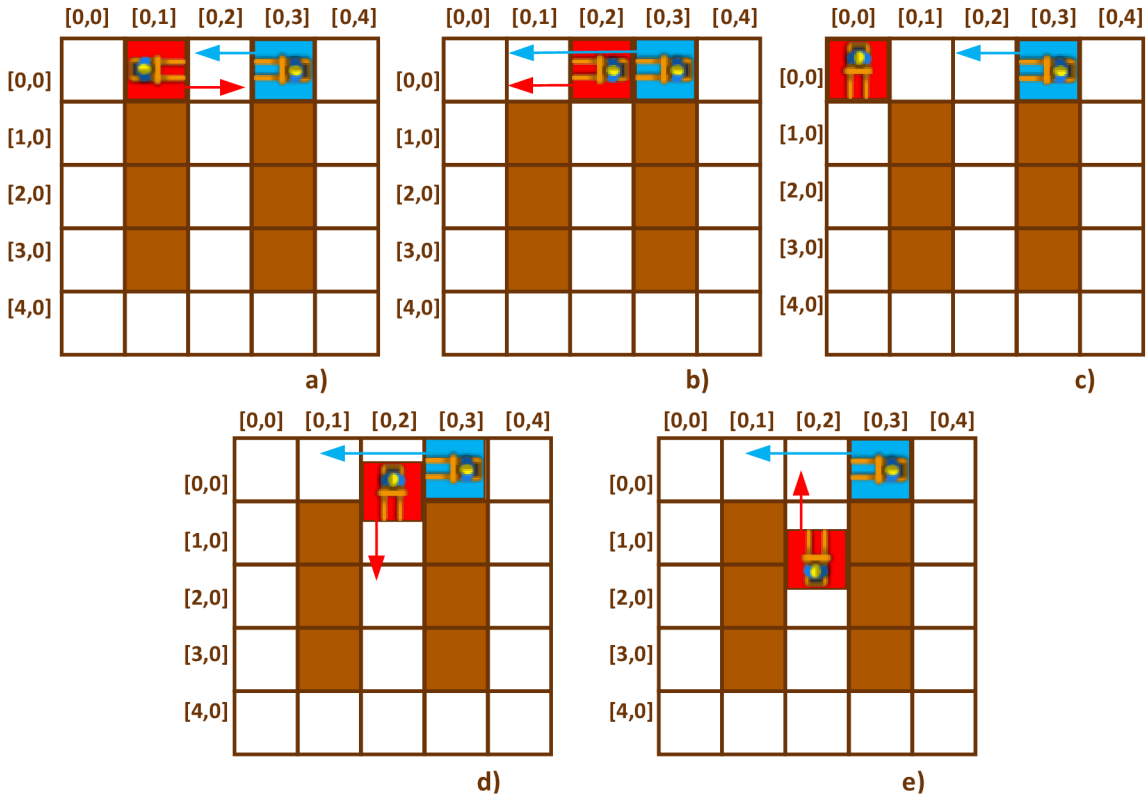

Fig. 5.2: Types of collisions in the warehouse environment.

Collisions can be divided into two basic categories. The first category is the collision in *a straight direction*, depicted in Fig. 5.2. The trucks in this category are moving in the opposite directions (see Fig. 5.2a), in the same direction (see Fig. 5.2b), or only one truck is moving (see Fig. 5.2c). 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. 5.2d), or directly, i.e. both trucks are moving towards the cell of collision (see Fig. 5.2e). Off course, the other parameters such as the type of aisle (wide aisle or narrow aisle) have to be considered.

5.3 A Numerical Example of Truck Collision

The following text describes a numerical example of collision prediction. Fig. 5.3 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.

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. 5.3. 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. 5.3 in orange color. The process of collision calculation is present in the next paragraph.

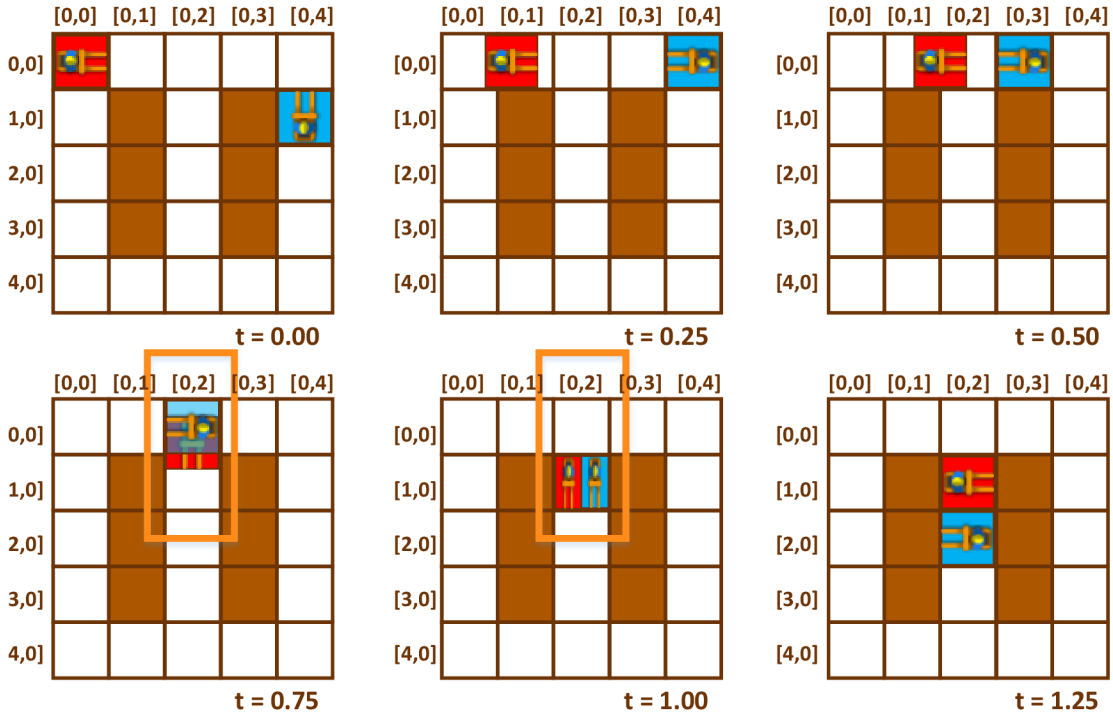

Fig. 5.3: A numerical example of the collision detection.

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. 5.3. 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 %

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.

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.

6.1 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, 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 [34] and [35]. 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. 6.1 is used in this section to clarify the examples and it is inspired by the grammar defined in [34] and [35]. An example of syntactical tree generated with respect to the defined grammar is also depicted in Fig. 6.1.

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 [34].

6.1.1 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 [35] 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.

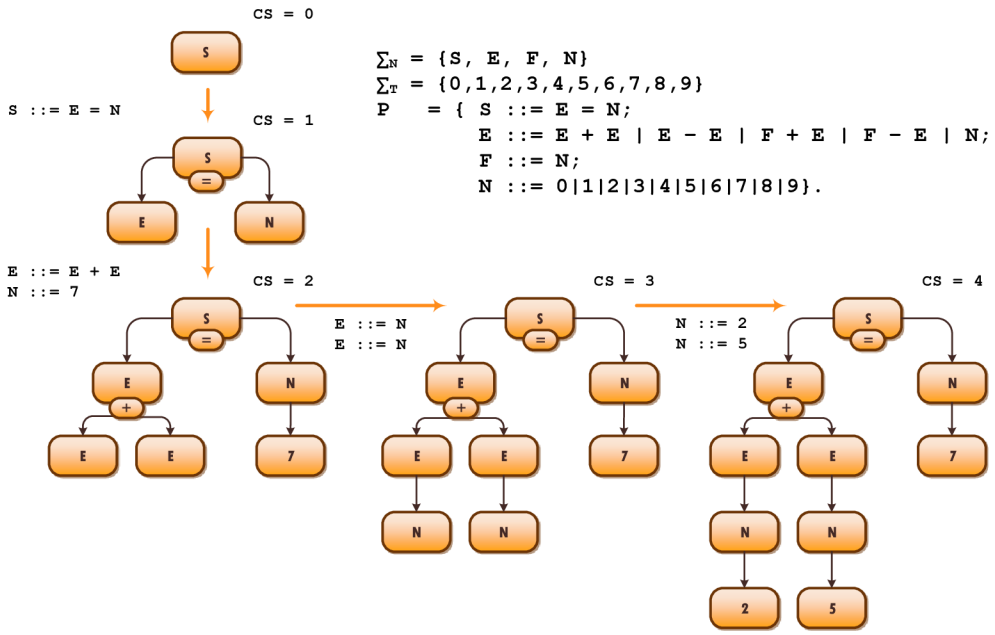

Fig. 6.1: Definition of the Context-free Grammar and an example of the process generating syntactical tree for the sentence $(2 + 5 = 7)$.

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 [34] and [35] 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 [34]:

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. 6.1 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. 6.1, 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.

6.1.2 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 [35] and slightly simplified in few steps. The process of how the operator works is described below (see Fig. 6.2):

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. 6.2.
3. The parent of $CN1$ regarding to given CFG in Fig. 6.1 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 [35] 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.

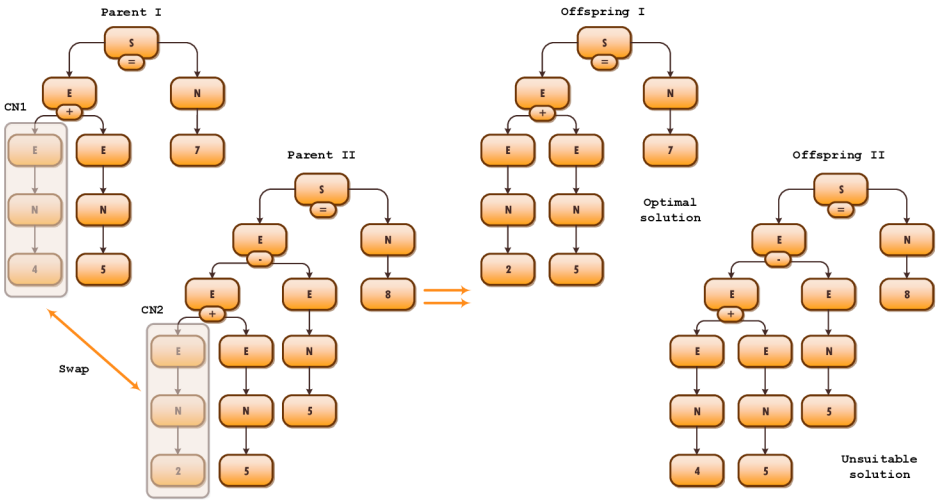

Fig. 6.2: An example of the Grammar Driven Crossover.

6.1.3 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):

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.

6.2 Framework Validation by Use Cases

The fist 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 [36], 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. The use case is described in more details in [46] 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 second 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. The success of the CCA localization in images of the testing set was approximately 75 %. More about the experiments using ultrasound images as inputs is described in [47], [48]. The experiment was further developed and performed on ultrasound video sequences with much better results. The resulting success of the proposed solution was 82.7 %, which exceeded the current state of the art by 4 % [49].

7 THE WAREHOUSE 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. [50], [51]

7.1 The Terminals and Non-Terminals

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, were stated as follows.

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 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.

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.

7.2 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:

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

7.3 The Run Controlling and the Termination

Thanks to the grammar defined in section 7.4 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: population size, evolution size (the number of generations), simulation time, elitism rate, path mutation (PA) rate, job order (JO) mutation rate, swap job (SJ) mutation rate, swap work-plan (SW) mutation rate, and split job (SP) mutation rate.

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 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.

7.4 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 1, and a descriptive example is shown in Fig. 7.1.

Algorithm 1 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.

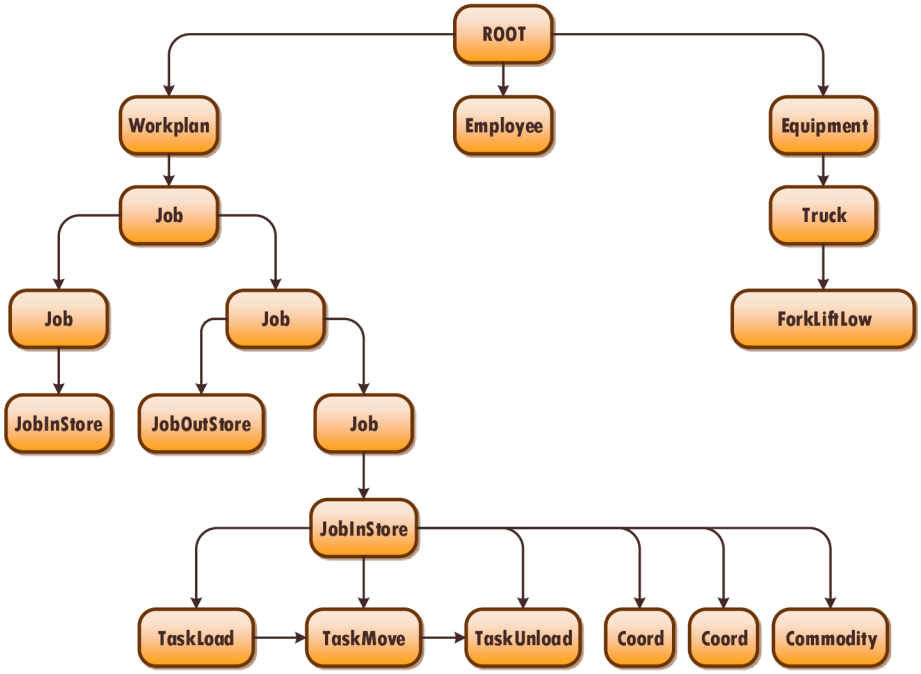

Fig. 7.1: An example of the tree generated by the context-free grammar.

7.5 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 (PA), Job Order Mutation (JO), Swap Job Mutation (SJ), Swap Work-Plan Mutation (SW), and Split Job Mutation (SP). 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.

The **Path Mutation** 7.2 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. 7.2 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 [52].

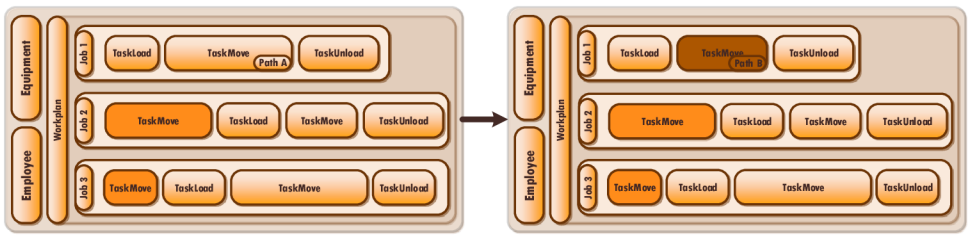

Fig. 7.2: An example of the Path Mutation operator.

The **Job Order Mutation** 7.3 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. 7.3 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 [52].

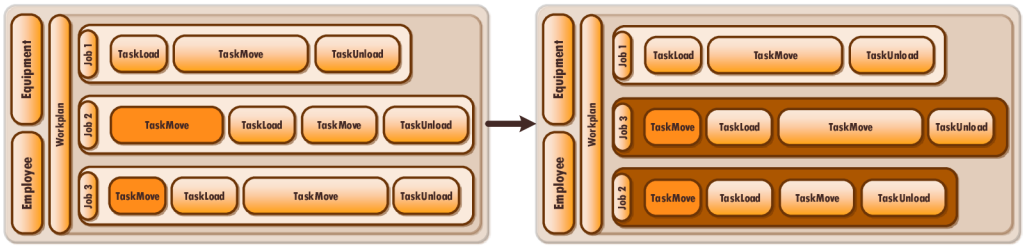

Fig. 7.3: An example of the Job Order Mutation operator.

The **Swap Jobs Mutation** 7.4 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. 7.4 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.

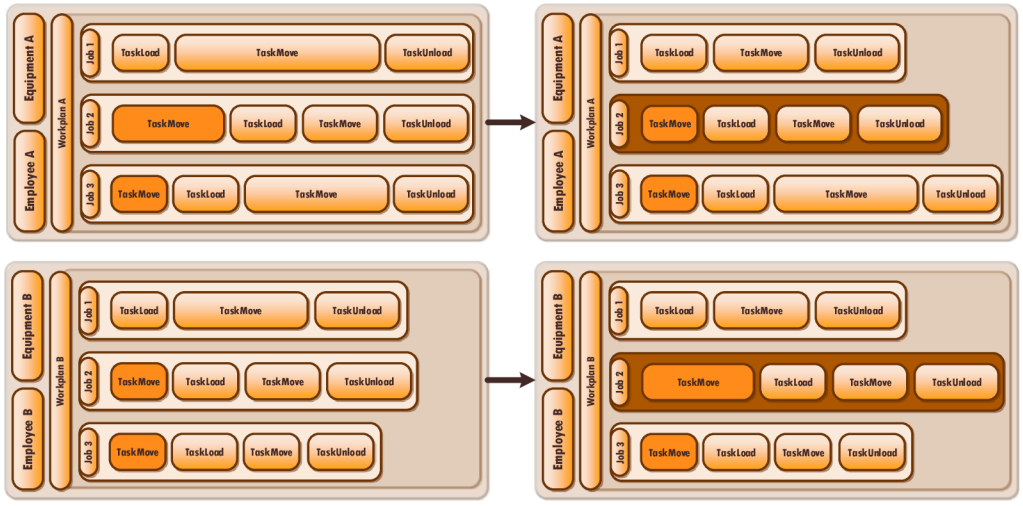

Fig. 7.4: An example of the Swap Job Mutation operator.

The **Swap Work-Plan Mutation** 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 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.

The **Split Job Mutation** is depicted in Fig. 7.5 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. 7.5. 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*.

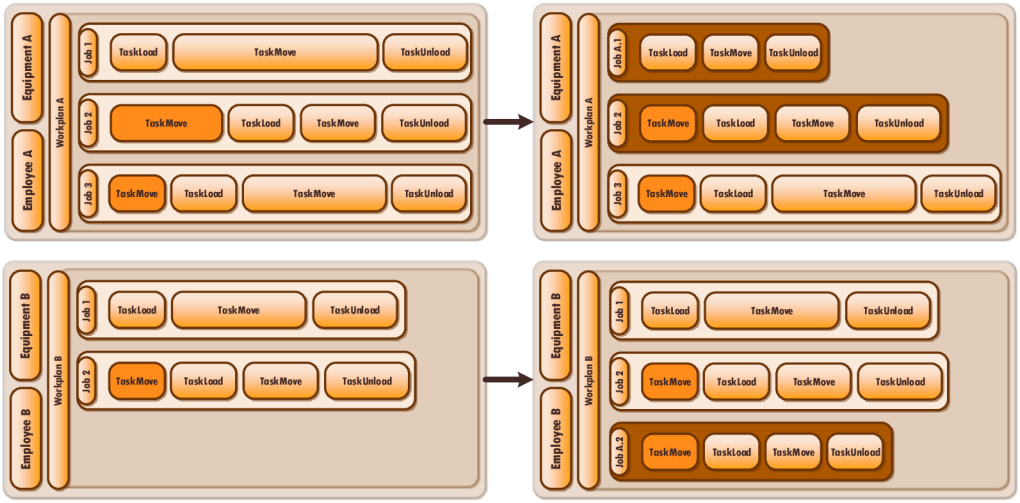

Fig. 7.5: An example of the Split Job Mutation operator.

7.6 Maintaining Mechanisms of Algorithm

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.

8 BENCHMARKING AND TEST SETS

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.

8.1 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 operations and normalized time of these operations are described in Tab. 8.1.

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

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

8.2 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.

The first set of scenarios (01–10) 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.

The second set of scenarios (11–20) also includes realistic scenarios of the warehouse work-flow. 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.

The third set of scenarios (21–30) 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.

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 fourth set of scenarios (31–40) 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.

The last set of scenarios (41–60) 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.

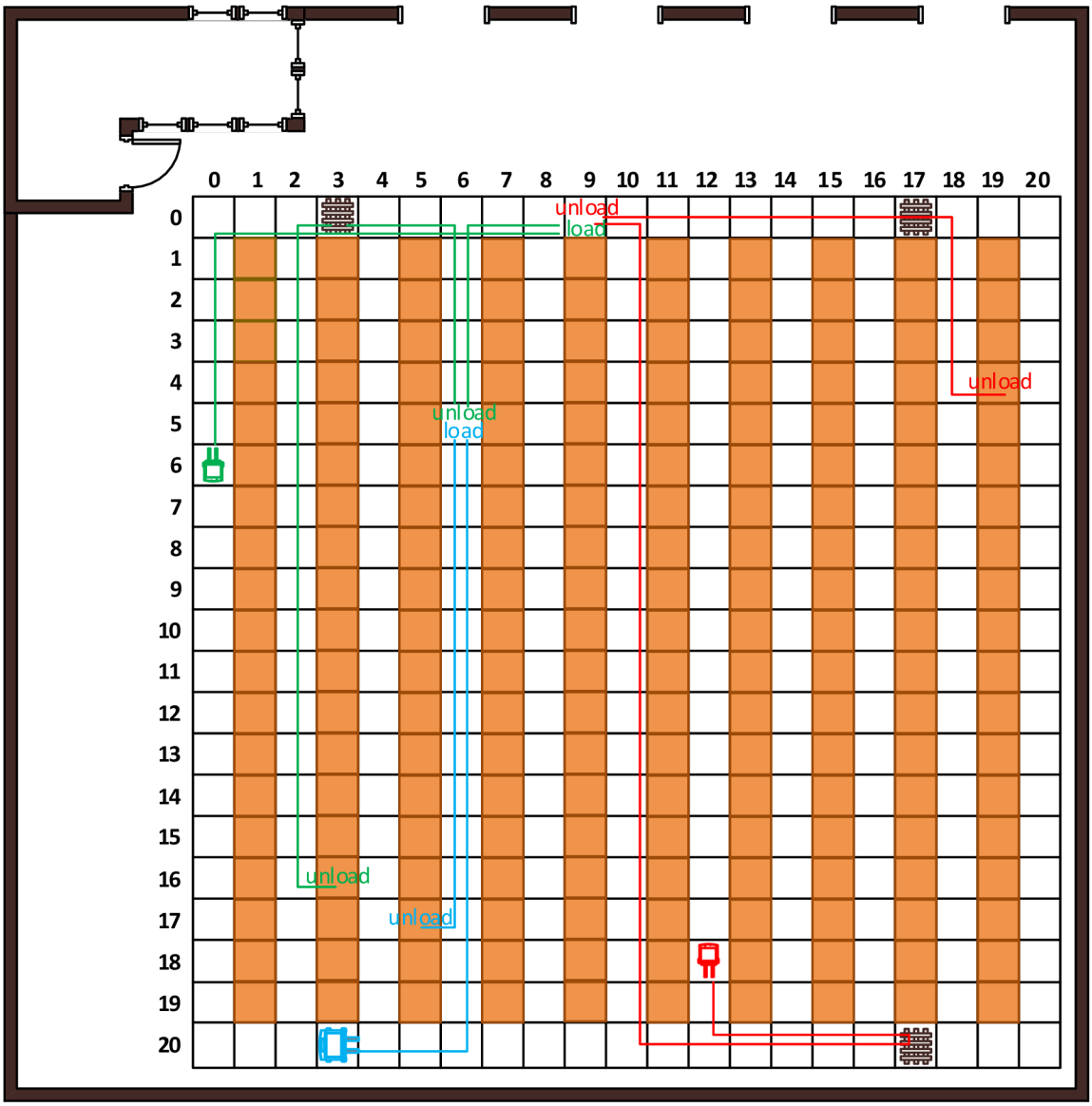

Fig. 8.1: An example of the scenario no. 54 from the fifth set.

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. 8.1 and described in Tab. 8.4.

Tab. 8.4: 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.

8.3 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. 8.2.

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. 8.5. 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

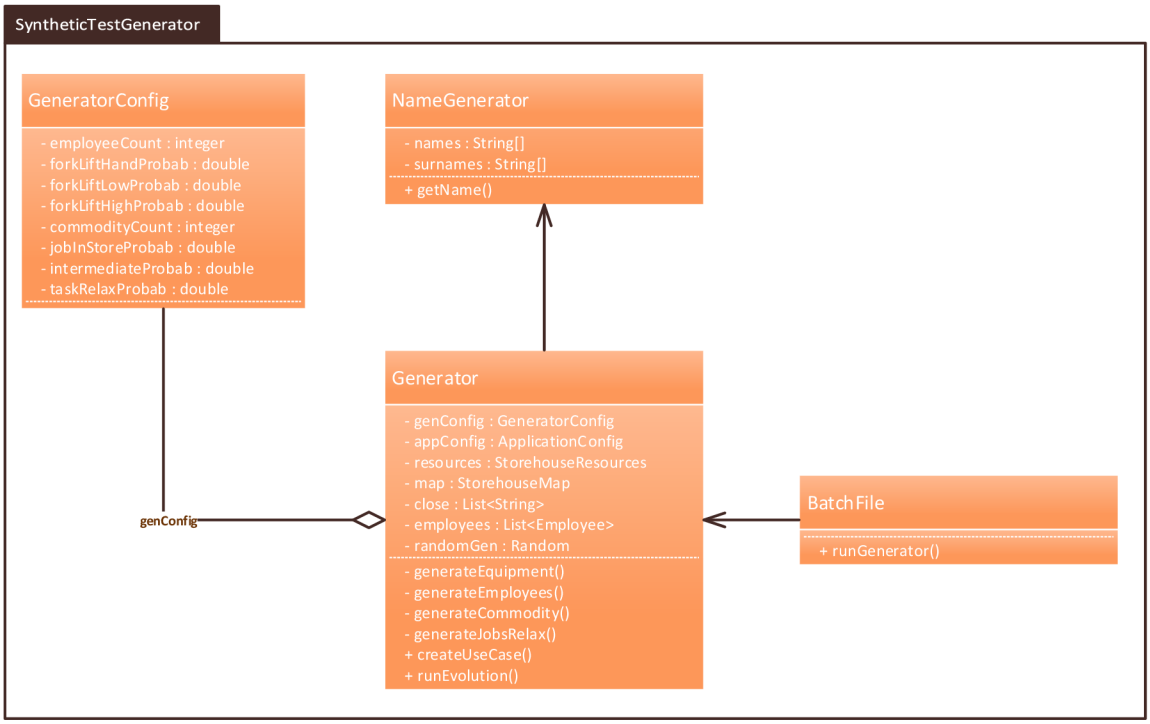

Fig. 8.2: The class diagram of the synthetic test set generator.

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. 8.5. 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. 8.5: 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

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 (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.

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.

The results of first ten scenarios 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
...	...	...	...	...	...	...	...	...	...	...

The results of second ten of scenarios 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
...	...	...	...	...	...	...	...	...	...	...

The results of third ten of scenarios 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
...	...	...	...	...	...	...	...	...	...	...

The results of fourth ten of scenarios 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,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

The last set of scenarios 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,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,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

9.1 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). In this shortened version of ph.d. thesis only the second case is presented.

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 are in full version of thesis. The results of the example of 20 employees and 100 jobs are given in Tab. 9.9. 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. 9.8: 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 = 20/80 %
JO Mutation Rate	Tab. 9.9 = 20/80 %
SJ Mutation Rate	Tab. 9.9 = 20/80 %
SW Mutation Rate	Tab. 9.9 = 20/80 %
SP Mutation Rate	Tab. 9.9 = 20/80 %

Tab. 9.9: 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

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 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 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.

10.1 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 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. 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 which can occure during the work-flow processing (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.

10.2 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.

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

[2] Michael T. Hompel and Thorsten Schmidt, *Warehouse Managmenet - Automation and Organisation of Warehouse and Order Picking Systems*, Springer-Verlag Berlin Heidelberg, 2007.

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

[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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*, K.P. Ellis, K. Gue, d. R. Koster, R. Meller, B. Montreuil, and M. Oglep, Eds., 2010.

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

[14] 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 Op. Res.*, vol. 221, no. 3, pp. 557–570, 2012.

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

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

[17] 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.

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

[19] 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.

[20] 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.*, 2013.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] Marvin L. Goldberger and Kenneth M. Watson, *Collision Theory*, Dover Publications Inc., Sep. 2004.

[33] 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.

[34] Marc García-Arnou, Daniel Manrique, Juan Rios, and Alfonso Rodríguez-Patón, "Initialization method for grammar-guided genetic programming," *Knowledge-Based Syst.*, vol. 20, no. 2, pp. 127–133, 2007.

[35] Jorge Couchet, Daniel Manrique, Juan Rios, and Alfonso RodríguePatón, "Crossover and mutation operators for grammar-guided genetic programming," *Soft Comp.*, vol. 11, no. 10, pp. 943–955, 2007.

[36] 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

[37] Jan Karásek and Lubomír Cvrk, "Stav vědy a techniky v oblasti genetického programování," *Elektrorevue*, vol. 15, no. 2, pp. 147–155, 2013.

[38] 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. ?–?

[39] 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.

[40] 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.

[41] Radim Burget, Jan Karasek, and Zdeněk Smékal, "Recognition of emotions in czech newspaper headlines," *Radioengineering*, vol. 20, no. 1, pp. 31–39, April 2011.

[42] 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.

[43] Radim Burget, Jan Karásek, Václav Uher, and Martin Zukal, "Semi–automatic image data analysis," *Elektrorevue*, vol. 2010, no. 1, pp. 61–67, 2010.

[44] 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.

[45] 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.

[46] Jan Karásek, Radim Burget, and Ondřej Morský, "Towards an automatic design of non-cryptographic hash function," in *34th Int. Conf. on Telecommunications and Signal Processing*, 2011, pp. 19–23.

[47] 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.

[48] 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*, December 2010, pp. 205–210.

[49] 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.

[50] Jan Karásek, Radim Burget, and Lukáš Povoda, "Logistic warehouse process optimization through genetic programming algorithm," in *Advances in Intelligent Systems and Computing*, April 2014.

[51] 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.

[52] Jan Karásek, Radim Burget, and Václav Uher, "Optimization of warehouse processes – benchmark definition," in *MENDEL 2013 19th International Conf. on Soft Computing*, 2013, vol. 1, pp. 45–50.

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.

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.